KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma / Tietotekniikka

Pavel Savushkin

JULKAISUJÄRJESTELMÄN HAKUTOIMINNON KEHITTÄMINEN

Opinnäytetyö 2013

TIIVISTELMÄ

KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikka

Savushkin Pavel	Julkaisujärjestelmän hakutoiminnon kehittäminen
Opinnäytetyö	33 sivua + 1 liitesivua
Työn ohjaaja	Yliopettaja Paula Posio
Toimeksiantaja	Nitro ID / ID Partners Oy
Huhtikuu 2013	
Avainsanat	Hakukone, stemmaus, tietokanta

Tietojen haku Internetistä tulee päivä päivältä tärkeämmäksi. Internetiin tulee teratavuja tietoa joka päivä ja niiden tehokas hakumahdollisuus on erittäin tärkeää. On olemassa hakukoneita, jotka käyvät koko verkon läpi ja tallentavat tietoa sivujen sisällöistä tietokantaansa. Sitä kutsutaan indeksoinniksi. Lisäksi on sellaisia hakukoneita, jotka toimivat ja tekevät hakuja sivuston sisältä. Nykyaikana melkein jokaisella yhtiöllä on oma sivustonsa, josta löytyy erilaisia tietoja. Sivusto ei voi olla täydellinen, jos sieltä puuttuu hakukone.

Työn tavoitteena oli kehittää ID Partners OY yhtiölle, joka myy web-sivustoja, oma hakukone, joka sitten integroidaan lopputuotteeseen. Sovellus toteutettiin PHP-ohjelmointikielellä Eclipse-ohjelmointiympäristössä. Sanojen käsittelemiseen käytettiin Porterin stem -algoritmia. Projektiin suunniteltiin luokat ja metodit ArgoUML-ohjelmalla. Tietokantojen suunnittelu tehtiin MySQL-WorkBench työkalulla ja niiden täyttäminen MySQL-relaatiotietokantaohjelmistolla. Indeksointialgoritmi pohjautuu Appache Lucene -tekstihaun ilmaiseen kirjastoon.

Lopputuloksena saatiin toimiva sovellus, joka käsittelee asiakirjan tekstiä ja indeksoi tekstin tallentamalla sanoja tietokantoihin. Varsinainen tietojen haku tapahtuu kyseisten tietokantojen kautta. Sovellus integroidaan osaksi web-sivustoa.

ABSTRACT

KYMENLAAKSON AMMATTIKORKEAKOULU
University of Applied Sciences
Information Technology

Savushkin Pavel	Search Engine of Content Management System
Bachelor's Thesis	33 pages + 1 appendix page
Supervisor	Paula Posio, Senior Lecturer
Commissioned by	Nitro ID / ID Partners Oy
March 2013	
Keywords	Search engine, stem process, database

Finding information on the Internet comes more important day by day. Terabytes of information are uploaded to Internet every day and search capability of this information is very important. There are search engines which go through network and store contained information in its databases. It is called crawling. Also there are such search engines that operate in a certain web-site. At present time almost every company has its own site, which consists of variety of information. This web-site cannot be complete if search engine is absent.

The goal of the thesis was to develop for ID Partners Oy company, which sells web-sites, its own search engine, which then would be integrated to the final product. This application was made by using PHP programming language and Eclipse programming environment. Porter stem algorithm was used to remove the more common morphological and inflexional endings from words. Classes and methods were planned by ArgoUML program. Database design was made by using MySQL-WorkBench software and databases were filled in by using MySQL programming language. Indexer algorithm was established with a free text search Appache Lucene library.

The final result of the empirical part of the thesis work is the implementation of true application. This application handles text in a document, indexes it and saves words in a database. Data search happens from this database. The application will be integrated to be part of web-site.

SISÄLLYS

TERMIT JA LYHENTEET

API	Akronyymi sanoista ***A**pplication **P**rogramming **I**nterface.* Se tarkoittaa määritelmää, jonka mukaan ohjelman eri osat voivat tehdä pyyntöjä ja vaihtaa tietoja eli keskustella keskenään.
BCPL	Akronyymi sanoista ***B**asic **C**ombined **P**rogramming **L**anguage.* Se on vanha tietokoneiden ohjelmointikieli.
C++	Yksi tärkeimmistä kaupallisessa ohjelmistokehityksessä käytettävistä ohjelmointikielistä.
CGI	Akronyymi sanoista ***C**ommon **G**ateway **I**nterface.* Se on tärkeä Web-ympäristön tekniikka, jonka avulla selain voi välittää dataa palvelimella suoritettavalle ohjelmalle.
Cross-platform	Tarkoittaa ohjelmointikieltä tai sovellusta, joka ei ole sidoksissa tiettyyn laitteistoalustaan tai käyttöjärjestelmään.
DOC	Lyhenne englannin sanasta ***doc**ument.* Se on tekstinkäsittelyohjelmissa käytetty tiedostopääte.
Fuzzy-search	Hakukonetyyppi, joka löytää väärin kirjoitettuja tai osittain syötettyjä sanoja.
GPL	Lyhenne sanoista ***G**eneral **P**ublic **L**icense.* Se on käyttöoikeuslisenssi ohjelmien julkaisemista varten. Lisenssi antaa laajat oikeudet muuttaa ja jakaa alkuperäistä sekä muutettua lähdekoodia sekä toteuttaa sovelluksia kyseisillä lähdekoodeilla.
HTTP	Lyhenne sanoista ***H**ypertext **T**ransfer **P**rotocol.* Se on protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon.

Java	Laaja teknologiaperhe ja ohjelmistoalusta, johon kuuluu muun muassa laitteistoriippumaton oliopohjainen ohjelmointikieli sekä ajonaikainen ympäristö virtuaalikoneineen ja luokkakirjastoineen.
JDBC	Lyhenne sanoista ***J**ava **D**atabase **C**onnectivity.* Se on Java-ohjelmointikielen rajapinta, joka määrittelee standardin tavan, jolla asiakassovellus voi käyttää tietokantaa.
Jini	Java-teknologia, jotka käytetään hajautettujen järjestelmien rakentamiseen.
Klusteri	Yleisesti klusterilla tarkoitetaan yhteenkuuluvaa joukkoa jotain asioita, joilla on taipumuksena pienetä tai kasvaa yhdessä.
MySQL	Relaatiotietokantaohjelmisto.
PDF	Lyhenne sanoista ***P**ortable **D**ocument **F**ormat.* Se on PostScript-kieleen pohjautuva ohjelmistoriippumaton, siirrettävä tiedostomuoto.
Perl	Lyhenne sanoista ***P**ractical **E**xtraction and **R**eport **L**anguage.* Se on tulkattava proseduraalinen skriptimäinen ohjelmointikieli.
PHP	Lyhenne sanoista ***P**ersonal **H**ome **P**ages.* Se on nykyisin Hypertext Preprocessor ohjelmointikieli, jota käytetään erityisesti Web-palvelinympäristöissä dynaamisten websivujen luonnissa.
PostgreSQL	Avoimena lähdekoodina jaettava olio-relaatiotietokantapalvelin, tietokannan hallintajärjestelmä.
Python	Monipuolinen, tulkattava ohjelmointikieli.

Replikointi	Se tarkoittaa sitä, että sama tieto on ainakin kahdessa eri paikassa P1 ja P2. P1:stä voidaan pitää tiedon lähteenä ja P2:sta tiedon kopiona. Kun tietoja päivitetään automaattisesti, vain P1:n muuttuneet tiedot kopioidaan P2:lle.
Ruby	Tulkattava, dynaaminen ja dynaamisesti tyypittävä oliopohjainen ohjelmointikieli.
SOAP	Akronyymi sanoista ***S**imple **O**bject **A**ccess **P**rotocol.* Se on tietoliikenneprotokolla, jonka pääasiallisena tehtävänä on mahdollistaa proseduurien etäkutsu.
Soundex	Foneettinen algoritmi, joka vertailee kahden nimen samankaltaisuutta, kun ne lausutaan englannin kielellä.
TCL	Lyhenne sanoista ***T**ool **C**ommand **L**anguage.* Se on luoma tulkattava ohjelmointikieli
XML	Lyhenne sanoista *EX**T**ensible **M**arkup **L**anguage*. Se on standardisoitunut merkintäkieli, jota käytetään usein alustariippumattomaan tiedonvälitykseen ja puurakenteellisen tiedon kuvaamiseen.

1 JOHDANTO

Yritys Nitro ID on Kotkassa, Kouvolassa ja Lappeenrannassa toimiva kokonaismarkkinointiviestinnän toimisto. Se on ollut Nitro FX Oy:n osa, mutta vuonna 2008 se yhtiöitetty ID Partners Oy:ksi, jonka markkinointinimenä on Nitro ID. Nitro ID:n missiona on auttaa asiakkaita saavuttamaan strategisia tavoitteita markkinointiviestinnän keinoin. Nitro ID tarjoa seuraavia palveluita:

- kanavavalinnat ja seuranta,
- brändi-identiteetti markkinointiviestintä,
- konsultointipalvelut,
- filmi- ja äänituotannot,
- Internet- ja multimediatuotanto.

Internet palveluille kuuluu julkaisujärjestelmät, eli Nitro ID tarjoaa asiakkaille kätevän työkalun web-sivuston kehittämiseen.

Nykyaikana jokaisen kehittäjän, joka tekee web-sovelluksia, täytyy toteuttaa sisällön hakutoiminto omassa projektissaan. Nitro ID:n tarjoamassa julkaistujärjestelmässä haku on realisoitu MySQL:n LIKE-kyselyillä. MATCH-kysely olisi teoriassa parempi, mutta se asettaa rajoitteita tietokannan taulun tyypille. Kumpikaan ei pysty tuottamaan kunnollisia hakutuloksia suomenkieliseen sisältöön, jossa sanat ovat taipuneet.

Opinnäytetyöni aiheeksi tuli yrityksen julkaisujärjestelmän hakutoiminnan kehittäminen ja parantaminen. Ohjelmointikieleksi tuli PHP, koska Nitro ID:n järjestelmä on kehitetty ja realisoitu PHP-ohjelmointikielellä.

Tässä raportissa luvussa 2 käydään läpi sanojen käsittely Porterin stemmausalgoritmilla. Luvussa 3 kerrotaan yleisesti tekstin hausta web-projekteissa - mitä työkaluja on nykyaikana olemassa - ja perustellaan omaa valintaa. Luvussa 4 kerrotaan tietokannasta - miten ja millä se on toteutettu. Luvussa 5 käsitellään hakutoimintoa - miten se on toteutettu, sekä eri vaihtoehtoja. Ja luvussa 6 on koko työn yhteenvetoa ja oma mielipide työstä.

2 STEMMAUS

Luonnollisissa kielissä sana koostuu vartalosta ja affikseista. Vartalo viittaa sanoman keskeiseen ideaan tai merkityksen. Affiksi on lisätty muokkaamaan sanan merkitystä ja tuottaa sanasta erilaisia sekä taivutusmuotoja että morfologisia variantteja. Sellaisilla morfologisilla varianteilla, joilla on sama vartalo, niillä on samantapainen perusmerkitys ja tämä vartalo voidaan pitää samanarvoisina tiedonhakusovellusten kannalta. Tiedonhaussa se on hyödyllisempi jos tällaiset sanat voidaan yhdistää. Stemmaus on sanojen katkaisun ja morfologisen analyysin eräänlainen välimuoto. Stemmaus ei ole kehitetty niin hyvä kuin morfologinen analyysi, mutta toimii parempi kuin sanojen katkaisu.(1.)

Englannin kielessä ”stem” tarkoittaa sanan vartaloa. Stemmauksen yhteydessä ”stem” kuvaa sananmuodon alkuosa, katkaistu sana. Suomen kielissä sanalla stemmaus käytetään nimitystä typistys tai typistäminen. Stemmauksen tarkoitus on typistää sana olennaiseen merkitykseensä poistamalla affiksit sanasta. Sanamuodon muutos tyvimuodoksi tapahtuu stemmauksessa suffiksisääntöjen perusteella. Sääntöihin liittyy ehtoja, jotka määrittelevät, missä tilanteissa suffiksit poistetaan. Eli stemmaus käsittelee sanoja ilman sanakirjaa. Lopputulokseksi saatu muoto ei välttämättä ole mikään kielitieteellinen sanavartalo tai sananmuoto.(1.)

Tiedonhaussa sanat, joilla on sama perusmerkitys, yhdistetään samaan muotoon ja sanat eri merkityksessä pidetään erikseen. Mutta ei ole eroa, ovatko vartalot oikeita sanoja vai eivät. Tiedonhaussa stemmaus poistaa pääteainekset sekä hakusanoista että hakemistotermeistä. Seuraavassa on esitettynä tiedonhaun kannalta stemmauksen etuja ja haitoja.(1.)

Etuja:

- Tiedonhakija voi esittää sanan, missä muodossa tahansa, stemmaus etsii merkityksen.
- Tunnettujen sanojen talletukseen tarvittavan hakemiston koko pienenee.
- Voidaan saada parempia hakutuloksia.

Haittoja:

- Hakutulokset voivat myös huonontua, koska stemmaus voi saada aikaan eri merkityksellisistä sanoista samat muodot.

Käytännössä stemmausta suoritetaan stemmereiden eli karsintaohjelmien avulla, algoritmeja tai sanakirjaa hyödyntäen (2). Koska stemmaukseen ei ole olemassa yhtä ainoaa oikea tapaa, myös karsintaohjelmia on useita erilaisia. Luonnollisissa kielissä on aina epäsäännöllisyyttä, jolloin mikään algoritmi ei pysty ottamaan huomioon kaikkia kielen omituisuuksia, siis stemmerit eivät ole täydellisiä. Tavallisesti stemmereitä on tehty eri tarkoituksiin, jolloin niissä on erilaiset lähestymistavat, riippuen siitä, mitä haetaan. (1.)

Porter stemmer on stemmausalgoritmi, jonka Martin Porter julkaisi vuonna 1980. Alkuperäinen versio oli tehty englannin kielelle ja toteutettu BCPL-ohjelmointikielellä. Myöhemmin Porter loi Snowball-nimisen ympäristön, jossa hän käytti oman algoritminsa pääideaa ja teki karsintaohjelmia muille suosituille luonnolliselle kielille. Algoritmi ei käsittele sanojen vartaloita lainkaan. Sen sijaan se leikkaa sanojen loppupäätteitä. Tästä syystä algoritmi toimii nopeasti, mutta ei aina tuota oikeaa lopputulosta.

Koska Nitro ID:llä on monikielisiä asiakkaita, yritys kehittää sivustoja eri kielillä. Suurin osa on suomea, mutta ruotsia, englantia ja venäjää kuitenkin löytyy. Siksi työn suunnittelussa sovittiin, että alussa kehitetään suomen- ja venäjänkielen typistysalgoritmeja, ja jos jää aikaa, englannin- ja ruotsinkielisiä.

2.1 Suomen kielen typistysalgoritmi

Suomen kieli ei ole indoeurooppalainen kieli. Se kuuluu suomalais-ugrilaisten kielten ryhmään, joka on osa uralilaisten kielten ryhmää. Sanojen päätteiden järjestelmä on äärimmäisen mutkikas, mutta tarkasti määritelty. Järjestelmää käytetään kaikissa substantiiveissa, adjektiiveissa ja pronomineissa. (3.)

Suomen kieli sisältää aksenttimerkkien muodot "ä" ja "ö".

Vokaalit merkitään kirjaimilla a, e, i, o, u, y, ä, ö. Loput merkit ovat konsonantteja.

Alussa määritellään R1 ja R2 niin, että

- R1 on sellainen sanan osa, joka alkaa ensimmäisen vokaalia seuraava konsonantin jälkeen tai on tyhjä, jos sellaista konsonanttia ei löydy.
- R2 on sellainen R1:n osa, joka alkaa ensimmäisen vokaalia seuraavan konsonantin jälkeen tai on tyhjä, jos konsonanttia ei löydy.

Esimerkiksi sanassa ”tietotekniikka” R1 on ”otekniikka” ja R2 on ”ekniikka”. Sanassa ”ohjelmoija” R1 on ”jelmoija” ja R2 on ”moija”.

Muunnos tapahtuu alla olevan kuvauksen mukaan kuusivaiheisesti.

1. Partikkelit.

   Etsitään pisin seuraavista päätteistä R1:ssä ja tehdään seuraavat vaiheet:

   - ”kin”, ”kaan”, ”kään”, ”ko”, ”kö”, ”han”, ”hän”, ”pa”, ”pä”, poistetaan jos sitä edeltää kirjaimet "n", "t" tai vokaali, esimerkiksi sanassa ”minäkin”;
   - ”sti”, poistetaan jos sisältyy R2:een, esimerkiksi sanassa ”ahkerasti”.

   Huomio: kirjaimille ”n”, ”t” tai vokaalille ei ole pakkoa olla R1:ssä, vaan poistettava pääte pitää olla R1:ssä. Ja samaa alhaalla.

2. Possessiivit.

   Etsitään pisin seuraavista päätteistä R1:ssä ja tehdään seuraavat vaiheet:

   - "si", poistetaan jos sitä edeltää "k", esimerkiksi ”esimerkiksi”;
   - "ni", poistetaan jos sitä edeltää "kse", vaihdetaan "ksi":ksi, esimerkiksi sanasta ”ehdotukseni” tulee ”ehdotuksi”;
   - "nsa", "nsä", "mme", "nne", poistetaan;
   - "an", poistetaan jos sitä edeltää "ta" tai "ssa" tai "sta" tai "lla" tai "lta" tai "na", esimerkiksi sanassa ”innostustaan”;
   - "än", poistetaan jos sitä edeltää "tä" tai "ssä" tai "stä" tai "llä" tai "ltä" tai "nä"
   - "en", poistetaan jos sitä edeltää "lle" tai "ine", esimerkiksi sanoissa ”ajatelleen” tai ”ajatuksineen”.

Seuraavat askelet tarvitsevat muutama määrittelyä:

- Määritellään ”v” (vokaali) on yksi seuraavista "a", "e", "i", "o", "u", "y", "ä", "ö";
- Määritellään ”V” (rajoitettu vokaali) on yksi seuraavista "a", "e", "i", "o", "u", "ä", "ö". ”Vi” tarkoittaa, että vokaalin jälkeen tulee "i"-kirjain.
- Määritellään ”LV” (pitkä vokaali) on yksi seuraavista: "aa", "ee", "ii", "oo", "uu", "ää", "öö";
- Määritellään ”c” (konsonantti), se on kirjain joka ei sisällä aikaisempi määritelty v:tä. ”cv” tarkoittaa, että konsonantin jälkeen tulee vokaali.

3. Sijamuodot.

   Etsitään pisin seuraavista pääteistä R1:ssä ja tehdään seuraavat vaiheet:

   - "hXn", poistetaan jos sitä edeltää X, missä X on V mutta X ei ole "u" (esimerkiksi ahan, ehen tai ihin);
   - "siin", "den", "tten", poistetaan jos sitä edeltää Vi, esimerkiksi sanassa ”halukkaiden”;
   - "seen", poistetaan jos sitä edeltää LV, esimerkiksi sanassa ”huoneeseen”;
   - "a", "ä", poistetaan jos sitä edeltää cv, esimerkiksi sanassa ”aitolahtea”;
   - "tta", "ttä", poistetaan jos sitä edeltää "e";
   - "ta", "tä", "ssa", "ssä", "sta", "stä", "lla", "llä", "lta", "ltä", "lle", "na", "nä", "ksi", "ine", poistetaan;
   - "n", poistetaan. Jos sitä edeltää LV tai "ie", poistetaan viimeinen vokaali, esimerkiksi sanastaa ”ohjelmien” tulee ”ohjelmi”.

4. Muut päätteet.

   Etsitään pisin seuraavista päätteistä R2:ssä ja suorittaa ilmoitettu toimintaa:

   - "mpi", "mpa", "mpä", "mmi", "mma", "mmä", poistetaan jos sitä ei edellä "po";

- "impi", "impa", "impä", "immi", "imma", "immä", "eja", "ejä", poistetaan.

5. Taivutusmuodot.

   Jos 3. askelessa pääte oli jo poistettu:

   - jos R1 lopussa on "i" tai "j", poistetaan sen.

   Jos 3. askelessa päätettä ei oltu poistettu

   - jos R1 lopussa on "t" ja sitä edeltää vokaali, poistetaan "t"
   - jos "t" oli poistettu, poistetaan "mma" tai "imma" R2:n lopussa, jos niitä ei edellä "po".

6. Siistiminen.

   - Jos R1:n lopussa on pitkä vokaali, poistetaan viimeinen kirjain;
   - Jos R1:n lopussa on cX, c on konsonantti ja X on "a" tai "ä" tai "e" tai "i", poistetaan viimeinen kirjain;
   - Jos R1:n lopussa on "oj" tai "uj", poistetaan viimeinen kirjain;
   - Jos R1:n lopussa on "jo", poistetaan viimeinen kirjain;
   - Jos sanan lopussa on kaksoiskonsonantti ja sitä seuraa vokaali tai tyhjä, poistetaan viimeinen konsonantti.

2.2 Venäjän kielen typistysalgoritmi

Venäjän kielellä epäsäännölliset muodot tuottavat kaksi tai useampia stemejä. Stemit venäjän kielessä voivat olla hyvin lyhyitä ja monet päätteet ovat myös osasia joista luodaan stopwords. Tietokoneen hakukoneessa, "stopword" tarkoittaa sanaa, joka esiintyy kielessä hyvin usein, kuten "on". Hakukone ohittaa nämä sanoja indeksoinnin yhteydessä. (4.)

Venäjän aakkosto sisältää seuraavat 33 kirjainta:

”а”, ”б”, ”в”, ”г”, ”д”, ”е”, ”ё”, ”ж”, ”з”, ”и”, ”й”, ”к”, ”л”, ”м”, ”н”, ”о”, ”п”, ”р”, ”с”, ”т”, ”у”, ”ф”, ”х”, ”ц”, ”ч”, ”ш”, ”щ”, ”ъ”, ”ы”, ”ь”, ”э”, ”ю”, ”я”.

Kirjainta ”ё” käytetään hyvin harvoin: sen sijalla käytetään usein ”е”-kirjainta.

Vokaalit merkitään kirjaimilla а, е, и, о, у, ы, э, ю, я.

Jokaisessa sanassa RV on sellainen sanan osa, joka alkaa ensimmäisen vokaalin jälkeen tai sanan loppuosa, jos se ei sisällä vokaalia.

R1 on sellainen sanan osa, joka alkaa ensimmäisen konsonantin seuraavan vokaalin jälkeen tai sanan loppuosa, jos konsonanttia ei löydy.

R2 on sellainen R1:n osa, joka alkaa ensimmäisen konsonantin seuraavan vokaalin jälkeen tai sanan loppuosa, jos sellaista konsonanttia ei löydy.

Nyt määritellään seuraavat päätteiden luokat:

- Perfektiivinen gerundi:
  - ryhmä 1: ”в”, ”вши”, ”вшись”;
  - ryhmä 2: ”ив”, ”ивши”, ”ившись”, ”ыв”, ”ывши”, ”ывшись”.

  Ryhmän 1 päätettä käytetään aina ”а” tai ”я” kirjaimen jälkeen.

- Adjektiivi:

  “ее”, ”ые”, ”ими”, ”ыми”, ”ей”, ”ий”, ”ый”, ”ой”, ”ем”, ”им”, ”ым”, ”ом”, ”его”, ”ого”, ”ему”, ”ому”, ”их”, ”ых”, ”ую”, ”юю”, ”ая”, ”яя”, ”ою”, ”ею”;

- Partisiippi:
  - ryhmä 1: “ем”, ”нн”, ”вш”, ”ющ”, ”щ”;
  - ryhmä 2: ”ивш”, ”ывш”, ”ующ”.

  Ensimmäisen ryhmän päätettä käytetään aina ”а” tai ”я” kirjaimen jälkeen.

- Refleksiivi:

  ”ся”, ”сь”.

- Verbi:
  - ryhmä 1: ”ла”, ”на”, ”ете”, ”йте”, ”ли”, ”й”, ”л”, ”ем”, ”н”, ”ло”, ”но”, ”ет”, ”ют”, ”ны”, ”ть”, ”ешь”, ”нно”;
  - ryhmä 2: ”ила”, ”ыла”, ”ена”, ”ейте”, ”уйте”, ”ите”, ”или”, ”ыли”, ”ей”, ”уй”, ”ил”, ”ыл”, ”им”, ”ым”, ”ен”, ”ило”, ”ыло”, ”ено”, ”ят”, ”ует”, ”уют”, ”ит”, ”ыт”, ”ены”, ”ить”, ”ыть”, ”ишь”, ”ую”, ”ю”.

Ensimmäisen ryhmän päätettä käytetään aina ”а” tai ”я” kirjaimen jälkeen.

- Substantiivi:

  ”а”, ”ев”, ”ов”, ”ие”, ”ье”, ”е”, ”иями”, ”ями”, ”ами”, ”еи”, ”ии”, ”и”, ”ией”, ”ей”, ”ой”, ”ий”, ”й”, ”иям”, ”ям”, ”ием”, ”ем”, ”ам”, ”ом”, ”о”, ”у”, ”ах”, ”иях”, ”ях”, ”ы”, ”ь”, ”ию”, ”ью”, ”ю”, ”ия”, ”ья”, ”я”.

- Superlatiivi:

  ”ейш”, ”ейше”.

Edellä kuvatut olivat *i*-päätteitä. *d*-päätteiden lista on lyhyt. Muodostettu sana on usein eri kieliopin luokasta tai sisältää eri merkityksen. d-päätteen liittämisen mahdollisuus voidaan havaita vain sanakirjaan viittaamalla eikä kieliopista.

- Derivational:

  ”ост”, ”ость”.

Määritellään adjektiiviseksi päätteeksi partisiippiin päätteen seuraava adjektiivi.

Joka luokan päätteen etsimisessä pitää aina alussa käsitellä pisin pääte.

Seuraavaksi kuvataan stemmauksen säännöt:

Kaikki testit pitää suorittaa RV-sanan osalta. Perfektiivisen gerundin testauksessa ”а” ja ”я” kirjaimet mikä tulevat ennen ryhmän 1 päätteestä pitävät itse olla RV-osalla. Muissa tapauksissa kirjaimet ennen RV-osaa ovat tarkistamattomia.

Seuraavaksi suoritetaan nelivaiheinen muunnos:

1. Etsitään perfektiivisen gerundin pääte. Jos löytyy, siirrytään seuraavaan kohtaan. Muussa tapauksessa poistetaan refleksiivinen pääte ja sitten etsitään adjektiivin, verbin tai substantiivin pääte. Jos yksi päätteistä löytyy, poistetaan se ja siirrytään seuraavaan kohtaan.
2. Jos sana loppuu “и” kirjaimella, poistetaan se.
3. Etsitään derivational pääte R2:ssa ja jos löytyy, poistetaan se.
4. Jos sanalla on superlatiivinen pääte, poistetaan se tai jos sanan lopussa on kaksois-“н”, poistetaan toinen niistä tai jos sanan lopussa on ”ь”, poistetaan se.

2.3 Karsintaohjelman testaus

Ohjelmistotestaus kuuluu olennaisena osana ohjelmistoprosessiin. Testaamisen päätavoitteena on havaita ohjelmistossa ilmenevät häiriöt, jotta viat voidaan paljastaa ja korjata. Testaaminen ei voi vahvistaa, että ohjelmisto toimii oikein kaikissa tilanteissa tai olosuhteissa, vaan se osoittaa pelkästään, että se toimii oikein tietynlaisessa ympäristössä. (5, 480.)

Karsintaohjelman toteuttamisen jälkeen piti varmistaa, että se toimii oikein. ”Snowball”-sivustolta löytyi molemmille kielille tiedosto, jossa oli annettu lähes 50 000 sanaa ja niille vastaavia typistyssanoja riveittäin. Toteutettu testausohjelma irrottaa jokaisesta rivistä testaussanan ja sen typistysvariantin. Sen jälkeen ajetaan testaussana karsintaohjelman läpi ja vertaillaan lopputulosta sanalistassa annetun stemin kanssa: niiden täytyy täsmätä.

Testauksen alussa havaittiin muutamia muunnosvirheitä. Suurin osa näistä johtui siitä, että käytössä olevan ”Snowball”-algoritmin sivustolla ei ollut selkeää kuvausta algoritmin toiminnasta, ja sivusto sisältää muitakin epätarkkuuksia.

3 TEKSTIN HAKU WEB-PROJEKTEISSA

Mitä tahansa modernia web-projektia on vaikea kuvitella ilman sisältöä. Nykyaikana sisältö eri muodoissa on hyvin tärkeä erilaisissa projekteissa. Tieto on melkein kaikkien projektien perusta, ja sisällön kasvaessa tietojen hakumahdollisuus tulee aina vain tärkeämmäksi. Jokaisen kehittäjän, joka tekee web-sovelluksia, tarvitsee toteuttaa sisällön hakua omassa projektissaan. Hakukoneiden vaatimukset ovat nykyään korke-

ampia kuin vuosi sitten. Toki joissakin hankkeissa on mahdollista käyttää esimerkiksi Googlen tarjoamaa hakukonetta. Jos sisältöä pitää hakea erilaisilla hauilla ja esittää monissa eri muodoissa, tarvitaan oma hakukone. Lisäksi mitä monimutkaisempi sovellus on ja mitä monimutkaisempi sisällön rakenne on, sitä useammin tarvitaan oma hakukone. Se on oma järjestelmänsä, joka sisältää hakukonesovelluksen ja palvelimen. Palvelin voi olla hankittu kolmansilta osapuolilta, jos tarjonta on tarpeeksi joustavaa ja muokattavissa. Tässä kappaleessa tutustutaan eri vaihtoehtoihin ja markkinoilla tarjolla oleviin hakukoneratkaisuihin.

3.1 Yleinen arkkitehtuuri ja terminologia

Hakupalvelin on kirjasto tai komponentti eli ohjelmistoratkaisu, joka itse ylläpitää omaa asiakirjojen tietokantaa, jossa hakua tapahtuu. Hakupalvelu antaa muille sovelluksille mahdollisuuksia lisätä, poistaa ja päivittää asiakirjoja. Tätä prosessia kutsutaan indeksoinniksi. Toinen komponentti on hakukone, joka vastaanottaa hakupyyntöjä, käsittelee indeksoinnissa tehtyä tietokantaa ja palauttaa tiedot, jotka vastaavat kyselyä. Nämä ovat tärkeimmät haun osiot ja ne on toteutettu samassa kirjastossa tai ne ovat erillisiä palvelimia.(6.)

Hakuun liittyviä tärkeitä parametreja.

- Indeksoinnin nopeus. Kuinka nopeasti hakukone käsittelee asiakirjoja ja tallentaa niitä omaan indeksiinsä. Yleensä tämä nopeus mitattaan megatavuina sekunnissa.
- Uudelleen indeksoinnin nopeus. Kun asiakirjoja muutetaan tai lisätään uusia, ne täytyy indeksoida uudelleen. Jos palvelin tukee inkrementaalista indeksointia, käsitellään vain uusia asiakirjoja ja koko indeksin päivittäminen tapahtuu myöhemmin tai ei ollenkaan. Toiset palvelimet vaativat indeksin kokonaista uudelleenindeksointia tietojen lisäämisen yhteydessä tai käyttävät muita indeksejä (delta-indeksejä), joihin lisätään vain uusia tietoja.
- Tuetut API:t. Useimmilla hakukoneilla on toteutettu API kaikille suosituimmille ohjelmointialustoille, kuten Java, PHP, Ruby ja Python.
- Tuetut protokollat. API:n lisäksi pääsyprotokollat ovat tärkeitä, jos järjestelmää halutaan käyttää ulkopuolisilla sovelluksilla, joilla ei ole pääsyä

alkuperäiseen API:iin. Yleensä tuetut protokollat ovat XML-RPC, SOAP tai pääsy HTTP:n kautta.

- Tietokannan koko ja hakunopeus. Kun tietokannassa on muutamia tuhansia asiakirjoja, kaikki hakukoneet toimivat lähes samanlaisesti. Jos tietokannassa on miljoona tai enemmän asiakirjoja, pitää valita joku käytetyn alustan tuttu realisaatio. Suuren tietomäärän kanssa tulee helposti ennalta arvaamattomia ongelmia, jolloin valmiiksi tutusta realisaatiosta on apua. Tämä parametri ei aina saa arvoa. Pitää tutkia jokaisen systeemin ominaisuuksia ja hakukoneen algoritmeja ja uudelleen indeksoinnin nopeutta.
- Tuetut dokumenttityypit. Jokainen palvelin tukee tavallista tekstiä, mutta jos on tarve indeksoida eri tiedostotyyppejä, esimerkiksi HTML, XML, DOC tai PDF, kannattaa valita toteutus, joka tukee näitä indeksointimuotoja. Tietenkin kaikki tämä voidaan toteuttaa omassa sovelluksessa, mutta valmis ratkaisu on parempi.
- Monikielisyys ja stemmaus. Oikean tuloksen saamiseksi erilaisilla kielillä tarvitaan sekä alkuperäisen koodauksen tukea, että työtä kielen ominaisuuksien kanssa, esimerkiksi stemmauksen kanssa.
- Alusta ja ohjelmointikieli ovat yhtä tärkeitä.

Toki on olemassa vielä paljon erilaisia tähän vaikuttavia parametreja ja itse tietojenhakualue voi olla monimutkaisempi. Tässä opinnäytetyössä tämä rajaus riittänee, koska hakukone on suhteellisen pieni.

Seuraavaksi kerrotaan lyhyesti eri hakukoneiden ratkaisuista.

3.2 Sphinx hakukone

Taulukko1. Sphinx hakukoneen ominaisuuksia.(6)

Tyyppi	erillinen palvelin, MySQL säilöntämoduuli
Alusta	C ++, alustariippumaton
Indeksi	yhtenäinen + delta-indeksi
Hakuominaisuudet	Boolean-haku, lauseiden haun tuloksen ryhmittelyn, sijoituksen ja lajittelun mahdollisuus
API ja protokollat	SQL DB; oma XML-rajapinta; PHP:lle, Rubylle, Pythonille, Javalle, Perlille sisäänrakennettu

	API
Tuetut kielet	Sisäänrakennettu englannin ja venäjän kielen stemmaus, Soundex morfologian toteutukseksi
Muita kenttiä	Kyllä, rajoittamaton
Asiakirjan tyypit	Vain teksti tai oma XML-muoto
Indeksin koko ja haun nopeus	Erittäin nopea indeksointi noin 10 Mt / s (riippuen CPU-nopeudesta), haku noin 0,1 sek / ~ 2-4 Gt indeksillä, tukee satoja GB indeksiä ja satoja miljoonia asiakirjoja
Lisenssi	Avoin lähdekoodin, GPL 2 tai kaupallisia
URL	http://sphinxsearch.com

Tässä työssä käsittelyistä moottoreista Sphinx on tehokkain ja nopein moottori. Sen erityinen etu on, että se on integroitu moneen suosittuun tietokantaan ja se tukee kehittyneitä hakuominaisuuksia. Se tukee myös ei-triviaaleja hakumahdollisuuksia kuten klusterointia, mutta sen paras ominaisuus on haun ja indeksoinnin erinomainen nopeus, sekä mahdollisuus rinnakkaisajolle ja näin ollen nykyaikaisen palvelimen resurssiseiden hyödyntämisellä. Suurten tietomäärien järjestelmille suositellaan Sphinxiä omaksi hakupalvelimeksi korkean vaatimustason hankkeisiin. Suosittujen tietokantojen, MySQL ja PostgreSQL, integraatio antaa mahdollisuuksia käyttää sitä tavallisessa web-kehitysympäristössä. Lisäksi API on olemassa monelle ohjelmointikielelle, pääasiassa PHP:lle. Itse hakukone on käännettävä lähdekoodista ja asennettava erikseen. Siksi se ei usein sovellu tavanomaiseen web-hotelliin.(6.)

3.3 Apache Lucene

Taulukko2. Lucene hakukoneen ominaisuuksia.(6)

Tyyppi	Yhden palvelimen tai servletin, upotettu kirjasto
Alusta	Java / Cross-platform
Indeksi	Inkrementaalinen indeksi, mutta vaatii segmenttien yhdistämistoimintaa
Hakuominaisuudet	Boolean-haku, lauseiden haun tuloksen ryhmit-

	telyn, sijoituksen ja lajittelun mahdollisuuksissa.
API ja protokollat	Java API
Tuetut kielet	morfologia puuttuu, on stemming (Snowball) ja muutamien kielien analysaattori
Muita kenttiä	Kyllä, rajoittamaton
Asiakirjan tyypit	Teksti, tietokantojen ineksoinnin mahdollisuus JDBC kautta
Indeksin koko ja haun nopeus	20 Mt / min, hakemistotiedostojen koko on vain 2 Gt (32-bittinen käyttöjärjestelmässä). Rinnakkainen haku on mahdollista useiden indeksien ja klusteroinin kautta (vaatii kolmannen osapuolen alustan)
Lisenssi	Avoin lähdekoodin, Apache License 2.0
URL	http://lucene.apache.org

Lucene on kuuluisin hakukone, joka on kehitetty alkuperin on kehitetty sovellusintegrointeihin. Se on laajasti käytetty esimerkiksi Eclipsissä. Hakukoneen positiivisia puolia on hyvä indeksin rakennus- ja varastointijärjestelmä, jossa voi samaan aikaan haun kanssa täydentää tai poistaa asiakirja sekä käyttää optimointia. Itse indeksi on rakennettu segmenteistä, mutta nopeuden parantamiseksi on suositeltavaa optimoida segmentit, mikä usein tarkoittaa lähes samaa kuin uudelleenindeksointi.

Negatiivisena puolena on kuitenkin hidas indeksointi (erityisesti, verrattuna Sphinxiin), tietokantojen monimutkaisuus ja APIn puutteellisuus (vain Java). Merkittävän suorituskyvyn parantamiseksi Lucene voi tallentaa indeksinsä hajautetuun tiedostojärjestelmään tai tietokantaan, mutta se edellyttää kolmansien osapuolien ratkaisuja, samaa kuin kaikille muillekin toiminnolle. Esimerkiksi alussa se voi indeksoida vain normaalia tekstiä. Kolmansien osapuolien sovellusten kanssa Lucene yltää muiden edelle. Millään muulla hakumoottorilla ei ole niin paljon käyttöliittymiä muihin ohjelmointikieliin ja käyttötapoihin. Yksi tärkeimmistä syistä on erittäin hyvä indeksitiedostojen muoto, jota kolmannen osapuolen ratkaisut käyttävät.(6.)

3.4 Xapian

Taulukko3. Xapian hakukoneen ominaisuuksia.(6)

Tyyppi	Sisäänrakennettuaan kirjasto
Alusta	C ++
Indeksi	Inkrementaalinen indeksi, läpinäkyvästi päivitetty haun rinnalla, toiminta useiden indeksien kanssa, in-memory indeksi pienimmille tietokannoille
Hakuominaisuudet	Boolean haku, lauseiden haku, maskin haku, synonyymihaku. Ryhmittelyn, sijoituksen ja lajittelun mahdollisuudet
API ja protokollat	C++, Perl API, Java JINI, Python, PHP, TCL, C# ja Ruby, CGI liitäntä XML/CSV formaatissa
Tuetut kielet	morfologia puuttuu, stemming useille kielille, oikeinkirjoituksen tarkastus hakukyselyissä
Muita kenttiä	Ei
Asiakirjan tyypit	Teksti, HTML, PHP, PDF, PostScript, OpenOffice / StarOffice, OpenDocument, Microsoft Word / Excel / Powerpoint / Works, Word Perfect, AbiWord, RTF, DVI, SQL tietokantojen indeksointi Perl tietokantojen liitännän kautta
Indeksin koko ja haun nopeus	Tunnetaan 1.5 teratavua asennuksia ja satoja miljoonia asiakirjojen määriä
Lisenssi	avoin lähdekoodin, GPL
URL	http://xapian.org

Tällä hetkellä Xapian on ainoa kilpailija Lucenelle ja Sphinxille. Xapian erottuu "elävänä" indeksinä eli se ei vaadi uudelleenindeksointia, kun asiakirjoja lisätään. Edukseen Xapianissa on erittäin voimakas kyselykieli, sisäänrakennettu stemming, oikeinkirjoituksen tarkastus ja synonyyminen tuki. Tämä kirjasto on paras vaihtoehto, jos in-

tegroitava sovellus on tehty Perl-ohjelmointikielellä tai tarvitaan indeksin rakentamiseen kehitettyjä mahdollisuuksia ja indeksin päivittäminen tapahtuu usein tai uudet vasta lisätyt asiakirjat on oltava välittömästi haettavissa. (6.)

En löytänyt mitään tietoa mahdollisuudesta lisätä asiakirjoihin lisäkenttiä ja saada niitä hakutuloksena, niinpä tämän hakukoneen ja oman hakunsysteemin yhteen nivominen voi olla vaikeaa. Pakkaus sisältää Omega-kirjaston yläosan, joka on valmis käytettäväksi itsenäisenä hakukoneena ja on nimenomaan vastuussa erilaisten asiakirjojen ja CGI-liitännän indeksoinnissa.

3.5 Tuloksia

Vaikka loppupäätös kysymykseen, sopiiko joku konkreettinen hakukone projektille vai ei, voidaan tehdä vasta tarkan tutkimuksen ja testausten jälkeen, muutamia johtopäätöksiä voi tehdä jo nyt. Sphinx sopii parhaiten, jos on tarve indeksoida isoja määriä tietoja MySQL tietokannassa ja haun sekä indeksoinnin nopeudet ovat tärkeitä. Se ei pysty tekemään erikoisia mahdollisuuksia kuten ”fuzzy search” ja Sphinxin valitessaan täytyy olla valmiita antamaan hakukoneelle erillinen palvelin tai vaikka klusteri.

Jos on tarvetta integroida hakukone omiin sovelluksiin, niin paras vaihtoehto on etsiä tarvittaville kielille Lucene-kirjastoja: kaikille suosituille kielille ne ovat olemassa, mutta voivat toimia suppeammin kuin originaali. Jos sovellus on Java-kielellä, niin Lucene on paras vaihtoehto. Mutta pitää ottaa huomioon, että Lucenessa on hidas indeksointi, indeksiä pitää usein optimoida ja Lucene on vaativa tietokoneen ja kovalevyn ominaisuuksista. PHP:lle Lucene on näköjään ainoa vaihtoehto ilman lisämoduuleja.

Xapian on hyvä ja hyvälaatuinen tuote, mutta ei ole niin suosittu ja joustava kuin muut. C++ kielisille sovelluksille se on paras vaihtoehto, mutta se vaati manuaalista viritystä, jos on tarvetta integroida se omaan koodiin tai käyttää sitä hakupalvelimena.

3.6 Oma hakukone

Monet mainituista hakukoneista vaativat sovelluksen asentamisen sivuston hosting-palvelimelle. Kuitenkin asiakas sijoittaa sivustonsa pääsääntöisesti shared hosting -ympäristöön, esimerkiksi Nebulalle tai Kymp/Optimille.

Shared hosting -ympäristöihin ei pääsääntöisesti ole mahdollista asentaa omia palvelinohjelmistoja, eikä niissä saa ajaa sellaisia ohjelmia, joita ei palvelimelta löydy ennestään. Tällöin olisi pakko rakentaa Nitro ID:n tuotteisiin kaksi erillistä hakuratkaisua, esimerkiksi Sphinx sekä PHP-pohjainen toteutus. Kuitenkin NitroID haluaa säilyttää vapauden hosting -palvelunsa valinnasta jatkossakin samalla pitäen ylläpidettävän ohjelmakoodin määrän pienenä.

Vaihtoehdoiksi näiden rajoitteiden vuoksi jäivät ZendFrameworkin Lucene käännös PHP:lle (https://github.com/zendframework/ZendSearch/), jonka kehitys näyttää olleen ainakin kuuden kuukauden ajan jäissä, tai oman ratkaisun toteuttaminen.

ZendFrameworkin Lucene-käännös käyttää alkuperäisen Java-toteutuksen tapaan hakuindeksin ylläpitoon sekä käsittelyyn tiedostoja. Uskoisin kuitenkin että tietokantapohjainen hakuindeksi olisi sekä joustavampi että indeksin eheyden säilyttämisen kannalta vakaampi ratkaisu.

Tiedostopohjaisessa ratkaisussa koen riskitekijöinä pääasiassa:

- vajavaisen tietojen kirjoituslukkojen tuen shared hosting -ympäristöissä
- mahdolliset tiedostopohjaisen indeksin hajoamiset

NFS-tiedostojärjestelmiä käytetään usein shared hostingeissa ja kaikki NFS-tiedostojärjestelmät eivät tue tiedostojen kirjoituslukkoja. Tällöin hakuindeksien yhtäaikainen (monen käyttäjän) käsittely saattaisi johtaa indeksin hajoamiseen. Todisteena tälläisistä ongelmatilanteista käy Googlehaku ’Lucene search "broken index"', tuloksenaan satoja hakutuloksia.

Lisäksi, koska shared hosting -palveluissa ylläpitäjät eivät voi tietää, mitä palvelun käyttäjillä kulloinkin on meneillään, on mahdollista, että esimerkiksi Lucenen indeksiä yhtenäistettäessä palvelin käynnistettäisiin uudelleen kesken kaiken, jolloin indeksi voisi rikkoutua.

Näiden tietojen pohjalta vaihtoehtona on joko tehdä oma PHP-pohjainen ratkaisu tai muuntaa ZendFrameworkin Lucene käännös käyttämään tietokantaa tietojen tallentamiseen tiedostojen sijaan. Näistä kahdesta vaihtoehdosta on mahdollista tehdä sovellettu oma PHP-pohjainen ratkaisu, jossa indeksointi tapahtuu Zendin-algoritmeillä.

4 TIETOKANTA

Nykypäivän www-palvelut tarjoavat käyttäjilleen erilaisia mahdollisuuksia ja ovat helposti yläpidettäviä. Myös tietokantojen käyttö hyödyntää niissä tarkkaan. PHP-ohjelmointikieli tarjoaa erittäin hyvän ja vakaan pohjan nykyaikaisten palvelujen rakentamiselle. Niitä on mahdollista ajaa sekä Linux- että Windows-alustalla. (7,9.)

Tietokantojen suunnittelu on tosi tärkeää monesta syystä. Ensiksi, suurin osa tiedoista tallennetaan tietokantoihin. Toiseksi, tarvittavat tiedot helposti löytyvät tietokannoista ja niitä voidaan nopeasti esittää tiedon hakijalle. Kolmas seikka on, että tietokannan rakennetta joskus muutetaan erillisestä syystä. Tällöin hyvä tietokannan suunnittelu säästää sekä aikaa että rahaa.

Tietokannan suunnittelussa on seuraavia vaiheita:

- käsiteanalyysi: tehdään karkeat piirustukset tietokannasta
- tarveanalyysi: täydennetään ja testataan tehtyä käsitemallia
- normalisointi: tarkistetaan, että malli on niin sanotussa kolmannessa normaalimuodossa
- taulujen muodostaminen: muunnetaan käsitemalli relaatiokannan taulurakenteiksi
- suorituskyvyn tarkistus: lisätään tarpeelliset indeksit ja mahdollisesti muutenkin viritetään kantaa. (9,10.)

Tavoitteita tietokannan rakenteelle ovat: (8,21)

- kattavuus: sisältää kaikki järjestelmissä tai kyselyissä tarvittavat tiedot ja yhteydet
- selkeys ja ymmärrettävyys: yksinkertainen rakenne, ilmaisuvoima; helppo kysellä
- muutosjoustavuus: laajennettavuus minimoiden nykyisten ohjelmien muutokset
- yleiskäyttöisyys: soveltuvuus erilaisiin ympäristöihin ja eri asiakkaille tarvitsematta muuttaa tietokannan rakennetta
- eheys: toisteisuuden välttäminen; oikeellisuus, sisäinen ristiriidattomuus
- ohjelmointimukavuus: selkeät tietorakenteet, sarakkeilla kiinteä merkitys

- suorituskyky eli tehokkuus: riittävä vastausaika tapahtumille ja riittävän tehokkaat eräajot.

Käsiteanalyysi on ensimmäinen vaihe tietokannan suunnittelussa. Sen aikana suunnitellaan tietokantaa loogisella tasolla ja muodostetaan käsitemalli ensin melko karkealla tasolla ja sitten tarkentaen. Käsiteanalyysinvaiheessa saadaan käsitemallirunko, joka kuvaa kohdealueen käsitteet, mutta ei vielä kaikkia tietoja.(8,24.)

Opinnäytetyön tietokanta tulee Appache Lucenen pohjalle, mutta ei samanlaisena. Indekserin tarkoituksena on käsitellä käyttäjän tallentamia asiakirjoja ja lisätä käsiteltyjä sanoja tietokantaan. Sen vuoksi ensimmäisen taulun nimeksi tulee Search_Word, joka sisältää annettuja sanoja.

Nitro ID: n asiakkaat haluavat, että heidän sivustonsa olisivat muutamalla kielellä. Siksi pitää vielä erottaa, millä kielellä asiakirja on tehty. Toinen taulukko on Search_Locale.

Käsiteltävä teksti voi olla missä tahansa asiakirjan paikassa, esimerkiksi otsikossa tai suoraan tekstissä. Sen vuoksi tehdään taulukko, joka sisältää infoa siitä, missä asiakirjan paikassa sanaa sijaitsee. Kolmas taulukko on Search_Field.

Haun aikana, kun käyttäjä etsii jotain sanaa, vastauksena pitää tulla linkit asiakirjoihin, joista sanaa löytyy. Luodaan neljäs taulukko Search_Document, joka sisältää tietoja kaikista käsitellystä asiakirjoista.

On olemassa tietoja, joita ei tarvitse käsitellä vaan tallentaa, koska ne voivat olla tärkeitä haun aikana. Luodaan viides taulukko Search_StoredValues, johon tallennetaan sellaisia tietoja.

Lopuksi luodaan kuudes taulukko Search_Indexer. Se on päätaulukko, joka täydentää indekserin työn aikana ja sisältää vain numeroita, jotka ovat viittauksia taulukosta toiseen.

Käsiteanalyysin jälkeen tietotarveanalyysin käsitemallia tarkennetaan sovelluksen tietotarpeiden perusteella ja lisätään kaikki puuttuvat tiedot (8,24).

Normalisointi on menetelmä, jolla voidaan tiettyjä sääntöjä noudattaen tarkistaa, ettei mallissa ole turhaa toisteisuutta. Pyritään niin sanottuun kolmanteen normaalimuotoon. (8, 25). Kolmas normaalimuoto kieltää attribuuteilta, jotka eivät ole avaimia, eitriviaalit toiminnalliset riippuvuudet muihin kuin avainehdokkaiden superjoukkoon.

Tämä tarkoittaa sitä, että taulun ei-avainkenttien pitää riippua vain avainkentistä.(10.) Opinnäytetyön tietokanta on suunniteltu näin: se on kolmannessa normaalimuodossa.

Lopuksi viritetään suorituskyky kohdalleen, mikä tarkoittaa lähinnä indeksointia (8,25).

MySQL-tietokanta on kenties helpointa tehdä MySQL Workbench välija-ohjelman avulla (11). Workbench on työkalu tietokantojen luomiseen MySQL-tietokantoja varten. Workbenchillä voidaan luoda vaikkapa tauluja, rutiineita ja näkymiä. Tämän lisäksi ohjelmasta löytyy työkalut tietokannan luomiseen monin eri tavoin, vaikkapa suoraan sql scriptiin. Workbench käyttää hyväkseen näytöohjaimen OpenGL-rajapintaa, joten nykyaikainen näytöohjain kuuluu minimivaatimuksiin.(12.) Tämän työkalun avulla tietokannan suunnitteluprosessi ja siten taulukon ja indeksin luominen palvelimelle tehdään helposti.

Opinnäytetyön tietokannan loppuversio, joka on tehty MySQL Workbench avulla, koostuu kuudesta taulukosta. Tietokannan skeema löytyy liitteessä 1.

Taulukko4. Tietokannan Search_Locale taulukko

Kenttä	Kuvaus
id	Kielen tunniste
locale	Kielen nimi, esimerkiksi english, finnish
short	Kielen lyhyt nimi, esimerkiksi fi, ru, en

Taulukko5. Tietokannan Search_Word taulukko

Kenttä	Kuvaus
id	Sanan tunniste
Search_Locale_id	Kielen tunniste. Vastaa millä kielellä sana on
word	Sana

Taulukko6. Tietokannan Search_Field taulukko

Kenttä	Kuvaus
id	Kentän tunniste

field	Kentän nimi

Taulukko7. Tietokannan Search_Document taulukko

Kenttä	Kuvaus
id	Asiakirjan tunniste
docRef	Asiakirjan viite

Taulukko8. Tietokannan Search_StoredValues taulukko

Kenttä	Kuvaus
Search_Document_id	Asiakirjan tunniste
Search_Field_id	Kentän tunniste
value	Arvo

Taulukko9. Tietokannan Search_Indexer taulukko

Kenttä	Kuvaus
Search_Document_id	Asiakirjan tunniste
Search_Field_id	Kentän tunniste
Search_Word_id	Sanan tunniste
position	Vastaa mikä paikalla tekstissä sana seisoo. Tarvitse kuin on olemassa peräkkäisen sanojen haku
weight	Sanan paino. Painosta riippuu hakutuloksen asiakirjan sortaus
keyword	Vastaa oliko sana stemmattu tai ei, koska on olemassa sellaisia sanoja joka ei saa stemmata.
field_numb	Kentän numero. Voi olla että sama sana tavataan samassa asiakirjassa, samannimisessä kentässä. field_numb erottaa niitä.

5 HAKUTOIMINTO

Suurin osa käyttäjän ajasta netissä kuluu tietojen etsimiseen. On monia tapoja saada nämä tiedot: voidaan yrittää löytää vastausta online-tietosanakirjasta, voidaan tehdä sähköpostitilaus kiinnostavasta aiheesta tai voidaan kysyä foorumeilla ja saada vastaus ammattitaitoisilta ihmisiltä.

Mutta eri tavoille tehdyn haun tehokkuus vaihtelee. Joku löytää tiedot heti, toisella haku kestää hyvin pitkän aikaa ja kolmas ei ehkä löydä mitään. Näin käy, koska haku netistä on kuin kalastusta: täytyy tietää, mistä kalastaa ja miten.

Tietojen haussa riittää, että kirjoitat hakukenttään kysymyksen. Mutta jos haku on liian laaja tai toisaalta liian suppea, käyttäjä on joko tarkennettava hakuaan tai yleistettävä sitä. Haussa ei oteta huomioon, onko teksti kirjoitettu isoilla tai pienillä kirjaimilla. Myöskään välimerkkejä ja erikoismerkkejä ei huomioida eikä stopword-listan sanoja. Se, kuinka osuvat tulokset saadaan, riippuu siitä, miten hyvin hakukysely oli tehty.

Mitä parempi hakukysely on, sitä parempi tulos tulee. Hakukone ei ole henkilö, vaan ohjelma, joka vertaa hakukyselyn sanoja ja lauseita nettisivujen sanojen kanssa. Kyselyn jokaista sanaa käytetään tarkentamaan hakua.

Koska käytössä on kyselyn kaikki sanat, jokainen sanaa tekee lisää rajoituksia. Jos on liian paljon rajoituksia, oikeasti haluttu tieto voidaan ohittaa, joten haku on parasta aloittaa muutamista keskeisiä sanoista ja käyttää täsmällistä hakua, mikä lisää relevanttien tuloksien todennäköisyyttä.

Alussa opinnäytetyön vaatimuksissa sovittiin, että suunnitellaan vain yhden sanan haku. Mutta työn aikana tuli selväksi, että pitää laajentaa hakupiiriä. Tätä varten päätettiin ottaa käyttöön hakukoneen lisäkomentoja.

Hakukoneen lisäkomennot antavat mahdollisuuden saada parempia tuloksia. Niiden avulla voidaan rajata hakualuetta ja määrittä hakukoneelle, että ei tarvitse etsiä kaikilta sivuilta.

Lisäkomennot kirjoitetaan hakukyselyn kenttään. Käytettävissä oleville resursseille ei tietenkään ole mahdollista tehdä niin täydellisiä hakutoimintija kuten esimerkiksi Googlessa. Mutta peruskomennot on tehty.

Hakukoneessani on realisoituna neljä eri ryhmää. Ne ovat:

- Tavalliset sanat tekstissä
- Peräkkäiset sanat tekstissä
- Tietojen haku tunnetuista kentistä avainsanan avulla
- Tietojen haku tunnetuista kentistä

Tavalliseksi sanaksi ymmärretään sana ilman erikoismerkkejä joka on kirjoitettu hakukenttään. Esimerkiksi, ”ammattikorkeakoulu” tai ”Kotka” (lainausmerkkejä ei ole kirjoitettu hakukenttään). Jos kirjoittaa ”ammattikorkeakoulu Kotka” hakukone palauttaa kaikki ne asiakirjat, joissa on nämä sanat. Kaikki annetut sanat stemmataan ja tarkistetaan stopwordiksi ennen hakua.

Peräkkäisiksi sanoiksi ymmärretään sanat, jotka ovat peräkkäin ja lainausmerkkien sisällä. Myös heittomerkkiä on käytössä. Esimerkiksi hakukyselyssä ”Kotkan ammattikorkeakoulu” (lainausmerkit kirjoitettu hakukenttään) etsitään kaikki asiakirjat, joissa sanat Kotka ja ammattikorkeakoulu ovat peräkkäin. Tässä tapauksessa hakutekstin sanat stemmataan, tarkistetaan stopwordeiksi ja sitten etsitään tietokannassa vastaavat sanat, jotka ovat peräkkäin.

Tunnetun kentän nimen avulla avain sanan haku tapahtuu, kun käyttäjä käyttää merkkiä ”#”. Esimerkiksi hakukysely title#ammattikorkeakoulu tarkoittaa että hakukone etsi tietokannasta vain title-nimisestä kentästä sanaa ammattikorkeakoulu. Tässä tapauksessa sanoja ei stemmataan eikä tarkistetaan stopwordiksi.

Tunnetun kentän nimen avulla sanan haku tapahtuu, kun käyttäjä käyttää merkkiä ”:”. Esimerkiksi hakukysely text:ammattikorkeakoulu tarkoittaa, että hakukone etsi tietokannasta vain text-nimisestä kentästä sanaa ammattikorkeakoulu, mutta tämä sana stemmataan ja tarkistetaan stopwordiksi.

Hakukyselyn käsittelyn tuloksena palautetaan neljä taulukkoa, jotka sisältävät asiakirjojen tunnistenumeroita ja asiakirjojen painoja – kuinka monta kertaa hakusanat on

löydetty asiakirjasta tai tyhjä taulukko. Sitten tehdään näiden neljän taulukon yhdistäminen, jolloin saadaan yksi taulukko, joka sisältää näiden asiakirjojen tunnisteita, jotka tavataan jokaisessa taulukossa.

Esimerkiksi, hakukyselyn ’ ”Kotkan ammattikorkeakoulu” opiskelija ”tietotekniikan suuntaus” title#opinnäytetyö text:theseus’ tuloksena palautetaan:

- asiakirjojen tunnistenumeroita sisältävä taulukko niistä asiakirjoista, joissa on sana opiskelija;
- asiakirjojen tunnistenumeroita sisältävä taulukko niistä asiakirjoista, joissa on peräkkäisiä ”Kotkan ammattikorkeakoulu” ja peräkkäisiä ” tietotekniikan suuntaus”;
- asiakirjojen tunnistenumeroita sisältävä taulukko niistä asiakirjoista, joissa title- kentästä löytyy sana opinnäytetyö
- asiakirjojen tunnistenumeroita sisältävä taulukko niistä asiakirjoista, joissa text- kentästä löytyy sana theseus.

Taulukoiden yhdistäminen palauttaa yhden taulukon, joka on lajiteltu painon avulla. Tämä taulukko sisältää niiden asiakirjojen tunnistenumeroita, jotka ovat kaikissa neljässä taulukossa. Sitten näistä tunnistenumeroista tehdään viittauksia ja viittausten mukaiset asiakirjat palautetaan käyttäjälle.

6 YHTEENVETO

Tietojen haku web-sivustoista on tärkeä tehtävä. Hakukoneesta riippuu, miten nopeasti ja täydellisesti käyttäjä saa tarvitsemaansa tietoa. Siksi hakukoneen kehittäminen on vaikea tehtävä.

Opinnäytetyön teorian aihepiiri oli minulle osittain tuttu. Aikaisemmin olen kehittänyt tietokantajasovelluksia ja käyttänyt paljon aikaa tekstikentän käsittelyssä. Mutta toisaalta PHP-ohjelmointikielen alkeet olen opiskellut vastaa muutamaa viikkoa ennen työn aloittamista. Mutta se ei häiritse minua, koska elämän mottoni on: ”jos ohjelmoija osaa yhtä ohjelmointikieltä – hän oppii helposti muitakin”.

Sovelluksen palvelinympäristö oli määrätty ennalta, siksi tietokantojen palvelimen hallinta jäi opinnäytetyön ulkopuolelle. Käytetyistä tekniikoista minulla oli aiempaa kokemusta, mutta tarvittava dokumentaatio on todella huonoa. Eniten ongelmia tuli stemmerin kehittämisessä, koska Snowball-sivulla annetut algoritmit eivät olleet täy-

dellisiä. Algoritmien täydennystä pystyin tekemään vasta testauksen jälkeen, kun oli selvää, mistä mikäkin virhe riippui. Esimerkiksi ruotsin kielen stem algoritmin määrittelyssä ei ollut, että sanan käsittely pitää aloittaa kolmannesta merkkistä.

Sovellus vastaa sille asetettuja vaatimuksia ja on kehitetty laajempana kuin oli alkuperin määritelty. Työn lopussa ehdin kehittää sekä englanninkielisen että ruotsinkielisen stem algoritmin. Opinnäytetyön tekeminen oli haastavaa ja itsenäistä. Tarvittavaa apua aina sain Nitron henkilökunnalta, erityisesti Matti Järviseltä. Yhteensä työ sisältää 5434 koodiriviä.

Toivon, että lähitulevaisuudessa opinnäytetyöni tulos otetaan käyttöön.

LÄHTEET

1. Silvanto, T. 2003. Tiedonhakumenetelmien klassikoita –seminaari. Helsingin yliopisto. Saatavissa: http://www.cs.helsinki.fi/u/tjvander/tihame/FinText/19.txt [Viitattu 16.12.2012]

2. Stemmaus. Vapaa tietosanakirja. Saatavissa: http://www.fi-free.info/?title=Stemmaus [Viitattu 21.2.2013]

3. The Finnish stemming algorithm. Porterin stemmausalgoritmin virallinen jakelusivusto. Saatavissa: http://snowball.tartarus.org/algorithms/finnish/stemmer.html [Viitattu 12.11.2012]

4. Russian stemming algorithm. Porterin stemmausalgoritmin virallinen jakelusivusto.Saatavissa: http://snowball.tartarus.org/algorithms/russian/stemmer.html [Viitattu 12.11.2012]

5. Kaner, C. - Falk, J. – Nguyen, H. 1999. Testing Computer Software. New York: John Wiley and Sons, Inc.

6. Лозовюк, А. 2008. Полнотекстовой поиск в веб-проектах: Sphinx, Apache Lucene, Xapian. Saatavissa: http://habrahabr.ru/post/30594/ [Viitattu 20.11.2012]

7. Heinisuo, R. – Rauta, I. 2007. PHP ja MySQL. Tietokantapohjaiset verkkopalvelut. Helsinki: Talentum.

8. Hovi, A. - Huotari, J – Lahdenmäki, T. 2005. Tietokantojen suunnitteli & indeksointi. Porvoo: WS Bookwell

9. Hovi, A. 2012. SQL-opas. Hansaprint Oy, Direct Vantaa

10. Manninen, J. 2011. Pilkkikisojen tuloslaskentaohjelma. Oulun seudun ammattikorkeakoulu. Saattavissa: http://publications.theseus.fi/bitstream/handle/10024/36283/Janne_Manninen.pdf?sequence=1 [Viitattu 7.2.2013]

11. Konttinen, J. 2012. MySQL-tietokanta MySQL Workbenchin avulla. Suomen liikemiesten kauppaopisto. Saattavissa:

http://opiskelu.businesscollege.fi/konju/dpoh432/category/mysql-workbench/ [Viitattu 7.3.2013]

12. Ahonen T. – Hamadi, S. 2010. MySQL Workbench –Asennus ja käyttöohje. Documents & Resources for Small Bussiness & Professionals. Saattavissa: http://www.docstoc.com/docs/23454573/MySQL-Workbench [Viitattu 7.3.2013]

Liite 1

TIETOKANNAN SKEEMA