KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ INŽINERIJOS KATEDRA

Povilas Balzaravičius

Metodas greitai duomenų paieškai duomenų bazėse

Magistro darbas

Darbo vadovas
dr. S. Drąsutis

Kaunas, 2010

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ INŽINERIJOS KATEDRA

Povilas Balzaravičius

Metodas greitai duomenų paieškai duomenų bazėse

Magistro darbas

Recenzentas
doc. dr. E. Karčiauskas
2010-05-31

Darbo vadovas
dr. S. Drąsutis
2010-05-31

Atliko
IFM-4/2 gr. Studentas
Povilas Balzaravičius
2010-05-31

Kaunas, 2010

Turinys

Summary

Full-text database search method

In this Master thesis the performance of full-text search is analyzed. Search engine is implemented by using PHP and MySQL also Memcached cache engine. Data for the research is gathered from RSS aggregator that periodically collects information from lithuanian blogs. The main attention is drawn on speed of both data indexing and search results gathering. While analyzing indexing process the speed of its performance was measured on different data sets. Indexing requires a lot of computer resources and can last long when having a lot of data. The duration of this process was measured with enabled and disabled Memcached service and list of ignored words. A test on how strongly query's number of elements influence the collection of results was performed. Performances were tested by using both InnoDB and MyISAM tables. Paper suggests solutions that are recommended for implementing and using similar search engines.

Santrauka

Magistro darbe nagrinėjama pilno teksto paieškos veikimo sparta. Paieškos sistema sukurta naudojantis PHP ir MySQL priemonėmis, taip pat panaudojant Memcached laikinosios atminties valdymo sistemą. Tyrimui atlikti reikalingi duomenys paimti iš RSS srautų agregatoriaus, periodiškai nuskaitančio informaciją iš lietuviškų tinklaraščių. Darbe didžiausias dėmesys skiriamas paieškos variklio indeksavimo ir rezultatų atrinkimo procesų vykdymo trukmei. Tiriant indeksavimo procesą, veikimo sparta išmatuota dirbant su skirtingais duomenų kiekiais. Turint daug duomenų, šis veiksmas reikalauja daug kompiuterio resursų ir gali trukti ilgai. Indeksuojant duomenis proceso trukmė buvo išmatuota naudojant ir nenaudojant Memcached ir ignoruojamų žodžių sąrašų. Paieškos tyrimo metu tirtas paieškos užklausos elementų kiekio įtaka rezultatų atrinkimui. Abiejų procesų veiklų matavimai atlikti naudojant InnoDB ir MyISAM lenteles duomenų saugojimui. Tyrimo metu rasti sprendimai, kuriuos rekomenduojama rinktis realizuojant arba naudojant panašiais principais paremtą paieškos sistemą.

1 Terminų ir santrumpų žodynas

Terminas	Paaiškinimas
DBVS	Duomenų bazių valdymo sistema.
InnoDB	MySQL DBVS duomenų saugojimo variklis. Palaiko transakcijas ir išorinius raktus.
Išorinis raktas (angl. *foreign key*)	Duomenų bazės lentelės laukas, apibrėžiantis ryšį tarp dviejų lentelių. Laukui priskiriama kitos lentelės pirminio rakto reikšmė.
Klasteris (angl. *cluster*)	Vienetas, sudarytas iš kelių vieno tipo elementų, kuris veikia kaip vieninga sistema, turinti tam tikrų savybių. Kompiuterių klasteris - spartaus ryšio kanalais sujungtų kompiuterių grupė, kuri vartotojo požiūriu veikia kaip vieningas įrenginys.
Leksinė analizė	Procesas, kurio tikslas iš simbolių sekos atpažinti atskirus žodžius.
MyISAM	MySQL DBVS duomenų saugojimo variklis, sukurtas pagal ISAM variklį. Naudojamas pagal nutylėjimą. Nepalaiko transakcijų. Palaiko pilno teksto indeksavimą.
Pilno teksto paieška (angl. *fulltext search*)	Technika žodžių paieškai atlikti indeksuotoje informacijoje. Paieškos metu ieškomi visi užklausoje pateikti ieškomi žodžiai ir nagrinėjama jų tarpusavio sąveika dokumentuose.
RSS (*Really Simple Syndication*)	XML failų formatų šeima internetiniam duomenų rinkimui iš interneto svetainių.
Tinklaraštis (angl. *blog*)	Internetinio puslapio forma. Dažniausiai tinklaraštyje periodiškai skelbiami autoriaus įrašai ir suteikiama galimybė apie juos palikti atsiliepimus.

2 Įvadas

2.1 Paieškos sistemų poreikis

Šiais laikais kompiuteriuose, prijungtuose prie interneto, saugomas neišmatuojamas kiekis informacijos. Kuriamos internetinės enciklopedijos, knygos ir žodynai. Net ir bibliotekose saugomos knygos dabar perkeliamos į skaitmeninį formatą. O eiliniai interneto vartotojai patys kasdien tinklą papildo nauja informacija. Šis procesas įgyja vis didesnį pagreitį. Vienas iš galimų pavyzdžių – didžiausia pasaulio enciklopedija Wikipedia. Pagal jos statistikos puslapį [15], šiuo metu saugoma 3,26 milijono straipsnių anglų kalba, o bendras straipsnių kiekis jau viršija 20 milijonų. Tai tik vienas iš pavyzdžių, parodantis, kokie duomenų kiekiai šiuo metu yra pasiekiami interneto vartotojams. Visa ši informacija liktų bevertė, jei nebūtų lengvai pasiekiama. Vartotojai negalėtų naudotis informacija neturėdami galimybės greitai rasti rezultatų, kurių jie tikisi.

Veiksmingos paieškos metodai gali būti taikomi ir mažesnėms sistemoms. Kuo mažesnis duomenų kiekis, tuo tikslesnius rezultatus galima pateikti (žinoma, tai priklauso nuo saugomos informacijos tipo). Dėl šios priežasties tikslinga diegti tobulesnius paieškos sprendimus ir į nedideles internetines parduotuves, bei kitus mažesnius tinklapius, kaupiančius informacijos archyvą.

Rezultatų pateikimas turi būti greitas ir efektyvus. Ieškojimo įpročiai remiasi paieškos tikslinimo metodu: negavus norimų rezultatų, paieškos užklausa yra tikslinama arba performuojama ir paieška kartojama. Vėlavimas pateikiant rezultatus erzintų vartotoją. Rezultatų atrinkimo efektyvumą atspindi sistemos sugebėjimas rasti su užklausa susijusius įrašus. Svarbu, kaip greitai paieška gali aptikti į duomenų saugyklą naujai įtrauktą informaciją. Tačiau vien tik atrinkti įrašus nepakanka. Priklausomai nuo duomenų kiekio ar užklausos abstraktumo lygio, atrinktų rezultatų gali būti tiek daug, kad vartotojas pats nesugebės rasti tinkamiausio varianto. Paieškos sistema turi įvertinti atrinktus duomenis – kaip tiksliai jie atitinka ieškomus žodžius. Galima atsižvelgti ir į kitus faktorius, tokius kaip statistika (pvz. konkretaus straipsnio peržiūrų skaičius) ar vartotojų reakcija (reitingai, rekomendacijos kitiems vartotojams ir t.t.). Pateikiant rezultatus tokia tvarka, kad jie kuo geriau atitiktų ieškomą frazę, išauga tikimybė, kad atrinkta informacija bus ta, kurios tikėtasi.

2.2 Tikslas

Darbo tikslas – rasti priimtiną metodą greitai ir efektyviai duomenų paieškai duomenų bazėse. Tiriamas sprendimas orientuotas į duomenų paiešką nepaskirstytose duomenų bazėse, kurios saugomos vienoje sistemoje. Dažniausiai tokio tipo duomenų bazės yra naudojamos

daugelyje interneto tinklapių, elektroninių parduotuvių ar kitų sistemų. Priimtinas paieškos veikimo principas turi leisti greitai vykdyti pilno teksto paiešką iki 10000 įrašų saugančiose duomenų bazėse. Vartotojui nepakanka pateikti visus pilno teksto paieškos metu rastus įrašus, atitinkančius jo pateiktą užklausą. Gauti rezultatai turi būti surikiuojami pagal tai, kaip tiksliai jie atitiko vartotojo ieškomą frazę. Rezultatų atitikimas vertinamas pagal daugelį faktorių: ieškomų žodžių poziciją tekste, jų pasikartojimo kiekį ir t.t. Paieška kuriama naudojant PHP programavimo kalbą ir MySQL duomenų bazę.

2.3 Uždaviniai

- Išanalizuoti indeksavimo vykdymo trukmę. Išmatuoti kokią įtaką šio proceso spartai turi laikinosios atminties ir ignoruojamų žodžių sąrašo naudojimas ir nenaudojimas.
- Ištirti indeksuojamų žodžių sąrašą. Rasti kokio ilgio žodžius tikslinga ignoruoti indeksuojant duomenis. Išmatuoti kokią įtaką sąryšių lentelės dydžiui turi ignoruojamų žodžių sąrašo naudojimas.
- Ištirti paieškos proceso vykdymo trukmę. Išmatuoti paieškos frazės elementų kiekio įtaką šio proceso veikimo spartai.
- Tyrimo metu palyginti veiksmų su duomenų baze spartą, kai duomenims saugomi naudojamos InnoDB arba MyISAM tipų lentelės.
- Rasti sprendimus, leidžiančius optimizuoti panašiai veikiančių sistemų darbą.

2.4 Naudojami įrankiai

Organizacija TIOBE kas mėnesį vertina ir pateikia programavimo kalbų bendruomenių indeksą. Šis indeksas yra laikomas programavimo kalbų populiarumo sąrašu. Dauguma jų pasižymi didesniu efektyvumu konkrečioje srityje, pvz., interneto tinklapių, programų kūrime, multiprogramavime ir kitose srityse. Visos jos yra pernelyg skirtingos, tad lyginti pagal bendras charakteristikas (pvz. resursų sąnaudos, kodo eilučių kiekis užduočiai atlikti ar kt.) negalima. TIOBE indekse kalbos yra rikiuojamos pagal jų bendruomenių stiprumą. Bendras domėjimasis kokia nors kalba apskaičiuojamas atsižvelgiant į tai, kiek rinkoje yra darbo pasiūlymų konkrečios kalbos programuotojams, kiek parduota knygų, susijusių su ta kalba, taip pat analizuojami interneto paieškų rezultatai. Pagal TIOBE indeksą PHP programavimo kalba jau ilgą laiką yra įsitvirtinusi 4 vietoje [14]. Tai viena populiariausių programavimo kalbų, orientuotų į interneto tinklapių kūrimą. Vieni iš populiariausių su PHP kalba sukurtų sistemų yra Wikipedia ir Facebook. Tai patvirtina, kad šios kalbos galimybės yra didelės ir ji

yra tinkanti tiek mažiems asmeniniams ar reprezentaciniams tinklapiams, tiek ir didelėms sistemoms. Kartu su PHP dažnai yra naudojama nemokama atviro kodo duomenų bazių valdymo sistema (DBVS) MySQL. Jos populiarumas augo kartu su PHP. Dėl savo galimybių, spartos ir kainos šiuo metu MySQL yra viena populiariausių DBVS. PHP ir MySQL derinys yra dažniausiai sutinkamas interneto svetainių kūrime ne tik Lietuvoje, bet ir visame pasaulyje. Dėl tokio populiarumo, buvo pasirinktas sprendimas naudoti būtent šiuos įrankius kuriant paieškos sistemą. Įrankių populiarumas ne tik parodo jų galimybes, tačiau kartu reiškia ir didelę bendruomenę, bei greitesnę paiešką informacijos, susijusios su sprendimais naudojančiais šiuos įrankius.

3 Paieškos sistemų analizė ir palyginimas

3.1 Paieškos tipai

Paieškos sistemos skirstomos į tipus pagal keletą savybių. Šios savybės glaudžiai susijusios duomenų bazių struktūra, kuriose saugoma informacija.

Pagal duomenų saugyklų tipą:

- **Lokalios sistemos** – duomenys saugomi vienoje sistemoje, vienoje duomenų bazėje. Tokioje duomenų bazėje saugoma informacija visada bus vienodo tipo. Daugumoje sistemų veikia tokio tipo paieškos.
- **Paskirstytos sistemos** – duomenys saugomi skirtingose duomenų saugyklose. Tai gali būti skirtingos duomenų bazės, veikiančios tame pačiame ar skirtinguose serveriuose arba kompiuterių klasteryje. Realizuojant tokio tipo paieškos sistemą, reikia rūpintis duomenų gavimu iš skirtingų saugyklų, paskirstytomis indeksų lentelėmis.

Pagal realizavimo tipą:

- **Vidinės sistemos** – realizuotos panaudojant naudojamos duomenų bazės ir programavimo kalbos įrankius. Visi indeksavimo, paieškos ir rikiavimo veiksmai realizuojami su turimomis technologijomis. Dažnai naudojama MySQL duomenų bazės ir PHP programavimo kalbos kombinacija. Šio tipo paieškos sistemas nėra sudėtinga kurti. Žinoma, sudėtingumas priklauso nuo atliekamų veiksmų ir naudojamų algoritmų, tačiau paprasčiausių realizacijų kūrimas gali apsiriboti keliomis kodo eilutėmis. Kitas analogiškų paieškų privalumas – nekyla problemų jas diegiant bendro naudojimo serveriuose (angl. *shared hosting*), kur vartotojas negali įdiegti papildomų servisų ir konfigūruoti tarnybinės stoties.
- **Išorinės sistemos** – papildoma programinė įranga, skirta vykdyti paieškas duomenų bazėse. Vienos populiariausių MySQL duomenų bazę palaikančių išorinės paieškos sistemų yra Sphinx, Apache Lucene. Dažnai šio tipo servisai yra nepriklausomi nuo naudojamo duomenų saugojimo tipo ir gali dirbti su įvairiomis duomenų laikmenomis, tokiomis kaip MySQL, PostgreSQL, Oracle, XML failais ir t.t. Dauguma išorinių sistemų taip pat yra nepriklausomos nuo duomenų bazių saugojimo variklių (MyISAM, InnoDB, Falcon). Šios sistemos gali veikti gerokai greičiau nei vidinės paieškos algoritmai, todėl jas

rekomenduojama naudoti esant dideliam duomenų kiekiui arba siekiant sumažinti duomenų bazės apkrovą.

3.2 Veikimo principai

Dauguma paieškos sistemų yra paremtos šiais pagrindiniais principais:

- duomenų rinkimas;
- duomenų indeksavimas;
- paieška;
- rezultatų rikiavimas.

3.2.1 Duomenų rinkimas

Duomenų rinkimas nėra privalomas visoms paieškos sistemoms. Interneto svetainėse dažniausiai paieškos yra vykdomos jau paruoštose ir saugomose duomenų bibliotekose. Tačiau ieškant informacijos tarp tarpusavyje nuorodomis susietų dokumentų (pvz. interneto puslapių), gali prireikti surinkti duomenis pradedant nuo pradinio taško – kelių pradinių dokumentų. Vykdant duomenų rinkimą (angl. *crawling*), peržiūrimo dokumento informacija yra išsaugoma duomenų bazėje, atrenkamos į kitus dokumentus vedančios nuorodos ir duomenų rinkimas toliau vykdomas jau naujai atvertuose dokumentuose. Interneto puslapių duomenų rinkimo sistemos turi susidoroti su šiais iššūkiais [3]:

1. Duomenų bazėje turi būti saugomos šviežiausios duomenų kopijos, tad informacija turi būti perrenkama pakartotinai. Naujai atsiradusi informacija taip pat turi būti įtraukiama.
2. Rinkimas turi būti maksimaliai efektyvus išnaudojant turimus resursus (interneto srauto limitą), tačiau privalo neapkrauti duomenų šaltinių.
3. Renkama turi būti tik „gera“ informacija (tačiau ne visada aišku, kuri informacija yra gera).

Su šiomis problemomis daugiau ar mažiau susiduria dauguma duomenų rinkimo sistemų (ne tik interneto puslapių).

3.2.2 Indeksavimas

Informacijos indeksavimu vadinamas procesas, kurio tikslas sužymėti, kokia informacija yra laikoma saugomuose duomenyse. Vykdant indeksavimą, sudaromas dokumentuose esančių žodžių sąrašas. Indeksavimo tikslas – sužymėti informaciją taip, kad vykdant paiešką būtų galima daug greičiau gauti dokumentų sąrašą, kuriuose yra ieškomi žodžiai. Sudarinėjant žodžių sąrašą, kartu yra išsaugomos nuorodos į dokumentus, kuriuose

šie žodžiai egzistuoja. Vykdant paiešką, tereikia indekse surasti ieškomą žodį ir nuskaityti nuorodas į dokumentus, kuriuose šis žodis yra naudojamas. Neturint informacijos indekso, dokumentai būtų analizuojami pakartotinai kiekvieną kartą vykdant paiešką. Toks veikimas papildomai apkrautų sistemą ir pailgintų paieškos vykdymo trukmę.

Indeksas padeda greičiau gauti paieškos rezultatus, tačiau pats indeksavimo procesas gali trukti pakankamai ilgai. Indeksuojant duomenis reikia išanalizuoti visą informaciją, kurioje norima vykdyti paiešką. Jei yra kuriama specifinė paieškos sistema, galima teigti, kad saugomų duomenų kiekiai yra nemaži. Tokiu atveju ir indeksavimas bus vykdomas santykinai ilgai. Kad išvengtume pilno duomenų indeksavimo, galima naudoti gyvo indeksavimo (angl. *live indexing*) metodiką. Gyvo indeksavimo principas – į atskirą indeksą suindeksuoti tik naujausius duomenis, o pagrindiniame indekse laikyti informaciją apie senus įrašus arba pagrindinį indeksą papildyti suindeksuojant naujausiai pasirodžiusius įrašus. Pavyzdžiui, turint sistemą, kurioje yra saugoma virš milijono įrašų ir kasdien sukuriama tik 1000–2000 naujų įrašų, gali būti naudojama pagrindinio ir papildomo indekso sistema, dar kitaip vadinama *main+delta*. Pagrindinis indeksas saugotų informaciją apie senus įrašus, nes daroma prielaida, kad sena informacija kinta retai. Tuo tarpu naujausi įrašai būtų saugomi *delta* indekse. Šis indeksas būtų daug kartų mažesnis už *main* indeksą, todėl jo atnaujinimas truktų daug trumpiau ir jį vykdyti būtų galima dažniau. Taip sukurta sistema leistų vykdyti informacijos paiešką neseniai sukurtuose įrašuose, tačiau nenaudotų daug resursų. Realaus laiko indeksavimas taikomas rečiau, nes indeksų sužymėjimas įtakoja sistemos veikimo spartą.

Įprastai suindeksavus teksto žodžius, pateikus bendrą užklausą galima nesulaukti norimų rezultatų, nors duomenų bazėje tokie įrašai egzistuotų. Taip gali nutikti dėl įvairių žodžių linksnių – norint aptikti dokumentą rezultatų aibėje, pateikiami užklausos žodžiai turi būti užrašyti tiksliai taip, kokiais linksniais jie saugomi dokumente. Egzistuoja tikimybė, kad dokumentas nebus priskirtas rezultatams, nes jo turinyje esantis užklausos žodis bus pateiktas kitu linksniu. Norint į rezultatų sąrašą įtraukti visus dokumentus, atitinkančius ieškomą frazę, tačiau nekreipiant dėmesio į žodžių linksnius, reikalingas kitokio tipo indeksavimas, taikant morfologijos algoritmus. Tokiu atveju indekse saugomi žodžiai yra konvertuojami į jų kamieninę dalį, šaknį arba konkretų linksnį. Vykdant paiešką, užklausoje esantys raktiniai žodžiai taip pat konvertuojami į tokį patį tipą, koks saugomas indekse. Tokiu būdu panaikinama žodžių linksnių įtaka rezultatams.

Dar vienas dažnas indeksavimo realizacijos klausimas – kokiu metodu atskirti tekste esančius žodžius. Turinys elektroninėje formoje saugomas kaip simbolių seka. Leksinė analizė – tai procesas, paverčiantis simbolių srautą į žodžių arba žetonų (angl. *tokens*) rinkinį

[5]. Žodžiu arba žetonu laikoma simbolių eilutė, nuo kitų žetonų atskirta tarpais arba skiriamaisiais ženklais [8]. Tekste esantys žodžiai dažniausiai yra skiriami tarpais bei skyrybos ženklais. Einant paprasčiausiu keliu, gali pakakti išskaidyti tekstą atskiriant žodžius pagal bet kokius ne tekstinius ir ne skaitmeninius simbolius, tačiau tokiu atveju raktinis žodis „C++“ nebūtų suindeksuotas. Kita vertus, skaidant žodžius pagal tarpus ar skyriklius, atskirais žodžiais gali būti palaikyti žetonai, esantys kabutėse (nors jas reikėtų pašalinti). Be to trupmeniniai skaičiai, atskirti kableliu arba tašku, bus išskaidyti į du atskirus žodžius. Renkantis tinkamiausią variantą reikia išanalizuoti nagrinėjamos kalbos ypatybes ir įvertinti sistemos poreikius.

3.2.3 Paieška ir jos rūšys

Paieškos vykdymo tikslas – atrinkti visus rezultatus, kurie atitinka pateiktą paieškos užklausą. Rezultatų atrinkimo būdas priklauso nuo paieškos tipo. Dažniausiai naudojamos tokios paieškos rūšys:

- **Frazės paieška** – Dokumentuose ieškoma tikslaus įvestos frazės atitikimo. Jei įvedami kelis žodžiai, tai ieškant yra svarbus jų eiliškumas. Tokio tipo paieška dažnai labai paprastai realizuojama su SQL užklausų operatoriumi „LIKE“.
- **Visų atitikmenų paieška** – Ieškoma dokumentų, kuriuose egzistuoja visi į užklausą įvesti ieškomi žodžiai. Jei bent vienas žodis dokumente nerandamas, dokumentas nebus priskirtas rezultatų aibei.
- **Bent vieno atitikmens paieška** – Rezultatų aibei priskiriami dokumentai, kuriuose randamas bent vienas užklausoje esantis žodis.
- **Loginė paieška** – Rezultatai atrenkami interpretuojant užduotą paieškos užklausą kaip loginę išraišką. Loginė išraiška gali būti aprašyta naudojant grupavimą, loginius „IR“, „ARBA“, „NE“ operatorius. Realizavus loginių užklausų vykdymą, galima vykdyti visus aukščiau aprašytus paieškos tipus.

3.2.4 Rezultatų rikiavimas

Nepakanka tik atrinkti rezultatus pagal duotą užklausą, ypač kai rezultatų kiekis didelis. Vartotojui reikia pateikti labiausiai jo užklausą atitinkančius dokumentus, kitaip jie bus pateikiami tokia tvarka, kokia yra surašyti duomenų bazėje. Kaip gerai dokumentas atitinka užklausą, galima įvertinti taškais, kurie skaičiuojami pagal įvairius konkretaus rezultato parametrus. Pagal tokį principą, daugiausiai taškų surinkę dokumentai vartotojui turėtų būti pateikiami pirmi. Keletas populiariausių taškų skaičiavimo metodų, taikomų tekstinių duomenų paieškai:

- **Raktinių žodžių pasikartojimas** – Įvertinama, kiek kartų dokumente pasikartoja ieškomas žodis. Laikoma, kad kuo daugiau pasikartojimų – tuo labiau turinys atitinka užklausą.
- **Žodžių pozicija dokumente** – Laikoma, kad dokumentai, kuriuose raktiniai žodžiai randami jų pradžioje, yra vertingesni nei kiti dokumentai. Labiausiai vertinami įrašai, kuriuose raktiniai žodžiai yra randami antraštėse.
- **Raktinių žodžių atstumas** – Kuo dokumente esantys raktiniai žodžiai yra arčiau vienas kito, tuo dokumentas labiau atitinka pateiktą užklausą.

Be šių dokumentų vertės skaičiavimo metodų galima prigalvoti ir daugiau: vertinti vartotojų elgseną (kuriuos rezultatus prie konkrečios užklausos jie pasirinko), atižvelgti į duomenų reitingą ir t.t. Kiekvienas iš šių metodų nebūtinai turi būti lygiavertis su likusiais. Rekomenduojama taškų skaičiavimo metodus įvertinti fiksuotais koeficientais. Suteikiami koeficientai leidžia vienų metodų rezultatus laikyti svarbesniais nei kitų. Šių koeficientų reikšmės priklauso nuo sistemos paskirties, naudojamų vertinimo algoritmų ir saugomų duomenų tipų. Tad kiekvienam projektui galima naudoti skirtingus koeficientus, parenkant geriausią derinį. Suskaičiavus bendrą taškų vertę visi gauti rezultatai surikiuojami mažėjimo tvarka ir pateikiami vartotojui.

3.3 Galimi sprendimai

3.3.1 Vienos SQL užklausos paieška

Pati paprasčiausia paieška SQL duomenų bazėse gali būti realizuojama vieninteliu operatoriumi LIKE. Šis operatorius leidžia ieškoti teksto šablono nurodytame duomenų lentelės lauke. Pavyzdžiui:

```
SELECT * FROM parts WHERE title LIKE '%memory%';
```

Įvykdžius aukščiau pateiktą užklausą, iš duomenų lentelės *„parts“* bus atrinkti visi įrašai, kurių pavadinime (laukas *„title“*) bus rastas žodis *„memory“*. Ieškant frazės iš daugiau nei vieno žodžio, užklausa būtų tokia (ieškoma frazė *„4gb memory“*):

```
SELECT * FROM parts WHERE title LIKE '%4gb memory%';
```

Tačiau tokiu atveju bus rasti tik tie įrašai, kurių pavadinime ieškoma frazė yra pateikta būtent ta tvarka, kokia ji yra nurodyta. Tokią problemą galima išspręsti išskaidant ieškomos frazės žodžius. Šiuo atveju žodžių tvarka nebesvarbi:

```
SELECT * FROM parts WHERE title LIKE '%4gb%'
     AND title LIKE '%memory%';
```

Norint vykdyti paiešką keliuose duomenų bazės laukuose, reikia kiekvienam papildomam laukui taip pat nurodyti paiešką su LIKE operatoriumi. Galutinis šio operatoriaus panaudojimo skaičius bus $n \times m$, kur n – paieškos frazės žodžių skaičius, o m – duomenų lentelės laukų, kuriuose vykdoma paieška, skaičius. Ieškant frazės *„4gb memory"* dviejuose lentelės laukuose *„title"* ir *„desc"*, SQL užklausa atrodytų taip:

```
SELECT * FROM parts WHERE
     (title LIKE '%4gb%' AND title LIKE '%memory%')
     OR (desc LIKE '%4gb%' AND desc LIKE '%memory%');
```

Naudojant šį paieškos būdą, nėra galimybės rikiuoti rezultatų pagal jų atitikimą užklausai, nebent atitikimas būtų skaičiuojamas realiu laiku, panaudojant vidines duomenų bazės valdymo sistemos (DBVS) galimybes. Tačiau yra galimybė rikiuoti rezultatus tokiu būdu: pirmiausiai pateikiami tie, kuriuose ieškomi žodžiai rasti, pvz., pavadinimo lauke, o tik po to tie, kuriuose žodžiai rasti aprašyme. Tokiu atveju naudojamos kelios analogiškos užklausos jas apjungiant operatoriumi UNION:

```
SELECT * FROM parts
     WHERE (title LIKE '%4gb%' AND title LIKE '%memory%')
UNION ALL
SELECT * FROM parts
     WHERE (desc LIKE '%4gb%' AND desc LIKE '%memory%');
```

Gerai išmanant naudojamos DBVS galimybes ir SQL kalbą, galima gauti paieškos rezultatus, kurie tenkintų nedidelius bazinius reikalavimus, tačiau didėjant duomenų kiekiui ir užklausos sudėtingumui, paieškos efektyvumas mažėtų.

Aprašytos paieškos metodikos privalumai:

- Nereikia papildomo programavimo kuriant paieškos sistemą, realizuojant indeksavimą.
- Nenaudojami resursai ir laikas indeksavimui.
- Realaus laiko paieška.

Trūkumai:

- Nenaudojamas indeksavimas – žodžiai ieškomi tiesiogiai nurodytuose duomenų laukuose, todėl indeksavimo privalumai negali būti išnaudoti.
- Lėtas paieškos būdas. Tai ypač pasireiškia vykdant sudėtingesnes užklausas ir ieškant informacijos, kai turime daug duomenų.
- Negalima tinkamai įvertinti rezultatų atitikimo užklausai.

- Neįmanoma tiksliai apibrėžti, kad ieškoma žodžio, o ne jo dalies. Taip yra todėl, kad tekstas nėra išskaidytas į atskiras dalis. Dažnai negana tiesiog pridėti tarpus iš abiejų ieškomos frazės pusių, nes žodžių skyrikliais gali būti ir kiti simboliai (pvz. taškas sakinio gale).

3.3.2 MySQL FULLTEXT indeksavimas

MySQL duomenų bazių valdymo sistema nuo 4.0 versijos turi galimybę MyISAM tipo lentelėse indeksuoti saugomą tekstinį turinį. Tam naudojamas FULLTEXT indeksavimas. Indeksuoti galima CHAR, VARCHAR ir TEXT tipų laukus. Žemiau pateikta užklausa „title“ ir „desc“ laukuose ieškos frazės „memory“.

```
SELECT * FROM parts
     WHERE MATCH (title,desc) AGAINST('memory');
```

Vienas iš FULLTEXT paieškos privalumų – loginės paieškos galimybė – palaikomi visi pagrindiniai loginiai operatoriai. Įvykdžius žemiau pateiktą užklausą, bus atrinkti rezultatai turintys raktinį žodį „memory“, tačiau būtinai neturintys žodžio „toshiba“.

```
SELECT * FROM parts
     WHERE MATCH (title,desc)
     AGAINST('+memory -toshiba' IN BOOLEAN MODE);
```

Raktinių žodžių nuspėjimo galimybė leidžia nuspėti susijusius rezultatus, nebūtinai turinčius ieškomus žodžius. Šis metodas veikia vykdant analogišką paiešką kelis kartus, tačiau antrą kartą nurodžius „WITH QUERY EXPANSION“ parametrą:

```
SELECT * FROM articles
     WHERE MATCH (title,desc) AGAINST ('database');
SELECT * FROM articles
     WHERE MATCH (title,desc)
     AGAINST ('database' WITH QUERY EXPANSION);
```

Po pirmos užklausos įvykdymo bus atrinkti įrašai, turintys žodį „database“. Pakartojus užklausą su QUERY EXPANSION parametru, bus atrinkti galimai susiję įrašai. Įrašų susiejimas matuojamas ieškant bendrų žodžių. Pavyzdžiui, jei tarp po pirmos užklausos vykdymo gautų rezultatų dažnai buvo rastas žodis „MySQL“, tai po antros užklausos vykdymo bus pateikti rezultatai turintys šį raktinį žodį, tačiau nebūtinai turintys žodį „database“.

Metodo privalumai:

- Nereikia papildomo programavimo kuriant paieškos sistemą, realizuojant indeksavimą.

- Realaus laiko paieška (įrašai indeksuojami juos įterpiant).
- Loginių užklausų palaikymas.
- Panašių rezultatų atrinkimas (QUERY EXPANSION).

Trūkumai:

- FULLTEXT indeksavimas veikia tik MyISAM tipo lentelėse.
- Visi FULLTEXT indekso stulpeliai turi būti tos pačios rikiuotės (collation) ir tos pačios koduotės (character set).
- Paieška neduos rezultatų jeigu užklausą atitiko daugiau nei pusė įrašų (50% taisyklė, pagal kurią tokie žodžiai laikomi per daug bendri).

3.3.3 Sphinx

Sphinx yra nemokamas atviro kodo pilno teksto paieškos variklis. Apžvalgai buvo naudojama 0.9.9 versija. Ši paieškos sistema veikia kaip atskiras servisas, leidžiantis kitoms programoms vykdyti pilno teksto paiešką duomenų bazėse. Sphinx yra integruojamas su SQL duomenų bazėmis. Šiuo metu oficialiai yra palaikomos MySQL ir PostgreSQL duomenų bazių valdymo sistemos. Taip pat galima naudoti duomenis, saugomus XML failuose [13]. Kitų duomenų bazių pajungimui yra galimybė naudoti pačių pasirašytas tvarkykles. Šio paieškos variklio naudojimui, oficialiai yra palaikomos PHP, Python, Perl, Ruby ir Java programavimo kalbos. Šiuo metu Sphinx galima naudoti daugelyje populiariausių operacinių sistemų: Linux (2.4.x, 2.6.x), Windows (2000/XP), FreeBSD, NetBSD, Solaris, Mac OS X.

Standartinių MySQL galimybių nepakanka norint vykdyti greitą ir efektyvią paiešką, kai duomenų kiekiai tampa dideli. Sphinx projekto tikslas – sukurti ypač efektyvią ir greitą paieškos sistemą, kuri galėtų dirbti su dideliais duomenų kiekiais saugomais MySQL/PostgreSQL duomenų bazėse. Galutinis rezultatas – itin spartus (paieškos ir indeksavimo prasme) ir lankstus paieškos variklis.

Šioje sistemoje indeksuojama gali būti bet kuri informacija, kuri gali būti saugoma kaip tekstinė eilutė (angl. *string*). Tie patys duomenys vienu metu gali būti indeksuojami skirtingais būdais. Vykdant paiešką tai suteikia galimybę parinkti tinkamesnę indeksų lentelę ir vartotojui pateikti geresnius rezultatus. Sphinx leidžia duomenis susieti su jiems priklausančiais atributais: įrašų sukūrimo/redagavimo laikai, dokumentų grupės ir t.t. Pagal šiuos atributus taip pat galima filtruoti reikiamus rezultatus. Vykdant indeksavimą, leidžiama taikyti morfologijos algoritmus, siekiant indekse esančius žodžius saugoti vienoje formoje (angl. *stemming*). Pagal nutylėjimą yra palaikomos anglų ir rusų kalbos. Norint įdiegti kitų kalbų palaikymą, galima naudoti Snowball biblioteką. Snowball bibliotekos tikslas – įvairių

kalbų (ne tik anglų) morfologinio žodžių nagrinėjimo algoritmų apjungimas ir nuolatinis palaikymas [12]. Be šio tipo algoritmų taikymo, Sphinx dar leidžia naudoti *wordforms* žodyno funkciją, kuri skirta atpažinti neteisingai užrašytus žodžius ir juos paversti į reikiamą formą. Tai paprasčiausias tekstinis žodžių sąrašas, kuriame saugomi žodžiai nurodyti neteisingose ir teisingose formose. Žodžiams, esantiems *wordforms* sąrašuose, aukščiau aprašytos morfologinės šaknies atskyrimo funkcijos nėra taikomos. Dažnai *wordforms* sąrašas yra naudojamas siekiant sugrupuoti pavadinimus, kurie gali būti užrašomi skirtingai pvz. „C++ > cplusplus“. Palaikomas „gyvo“ indeksavimo rėžimas, kurio metu perindeksuojama tik naujausia informacija.

Vykdyti paiešką galima įvairiais rėžimais. Dauguma jų analogiški MySQL FULLTEXT rėžimams: loginė, išplėstinė, frazės paieškos. Tinkamiausiems rezultatams įvertinti, galima naudoti atrinkimo pagal svorį rėžimą. Skirtingiems paieškos tipams naudojamos skirtingos vertinimo funkcijos. Pagrindiniai vertinimo tipai:

- **Frazės atitikimo skaičiavimas** – kuo dokumente rasta frazė labiau atitinka ieškomą frazę, tuo aukštesnis atitikimo įvertinimas. Jei dokumente rasta frazė visiškai atitiks ieškomą frazę (atitiks visi žodžiai ir jų tvarka), toks dokumentas gaus maksimalų įvertinimą.
- **Statistikos skaičiavimas** – skaičiuojami ieškomos frazės žodžių pasikartojimai. Kuo daugiau pasikartojančių žodžių, tuo didesnis įvertinimas. Jei konkretus ieškomas žodis yra retas visoje duomenų bazėje (būtent duomenų bazėje, o ne dokumente), jo svorio vertė bus didesnė nei kitų žodžių. Vertinimui naudojama BM25 vertinimo funkcija [9].

Tai tik keli pagrindiniai Sphinx nustatymai ir galimybės. Konfigūracijos parametrų yra daug daugiau. Jie leidžia susitvarkyti paieškos sistemą pagal projekto poreikius ir pritaikyti ją naudojamai sistemos architektūrai.

Sphinx privalumai:

- Didelė indeksavimo sparta.
- Didelė paieškos sparta.
- Palaikomos kelios pagrindinės programavimo kalbos: PHP, Java, Ruby, Python, Perl.
- Komandinės eilutės priemonės (paieškos vykdymas, indeksavimas), leidžiančios naudotis sistema, naudojantis kitomis programavimo kalbomis.
- Paskirstytos paieškos sistemų organizavimo galimybės.

- Įvairių paieškos tipų palaikymas (loginė, išplėsta paieškos).
- Rezultatų reitingavimas ir rikiavimas.
- Morfologijos algoritmų taikymas žodžių indeksavimui.
- MySQL, PostgreSQL ir XML duomenų saugyklų palaikymas.
- Tiesioginis indeksavimas.
- Daugybė konfigūravimo galimybių.

Trūkumai:

- Veikia kaip atskira programa (galima naudoti ne visuose serveriuose).
- Reikalauja papildomo konfigūravimo.

3.3.4 Apache Lucene

Apache Lucene – didelės spartos, daug galimybių turinti pilno teksto paieškos biblioteka, parašyta Java programavimo kalba. Ši biblioteka gali būti naudojama bet kurios programinės įrangos, kuriai reikalinga *fulltext* paieška [1]. Apache Lucene yra atviro kodo projektas. Dažnai Lucene yra naudojamas su SOLR paieškos serveriu. Šis serveris, naudodamas šią paieškos biblioteką, leidžia vykdyti paiešką priimdamas užklausas ir pateikdamas rezultatus XML/HTTP ir JSON formatais. Tokiu būdu paieška naudotis gali daugeliu programavimo kalbų parašytos aplikacijos. Pati Lucene biblioteka gali būti naudojama naudojantis komandine eilute.

Apache Lucene neturi MySQL palaikymo pagal nutylėjimą, tačiau paiešką šioje duomenų bazėje galima nesunkiai susikonfigūruoti naudojant pagalbinius įrankius. Lucene sukurtas taip, kad galėtų indeksuoti bet kokio tipo tekstinę informaciją, kuri yra paduodama indekso generavimo skriptui. Todėl MySQL ir kitų duomenų bazių palaikymas tampa neprivalomas, nes pakanka duomenis užkrauti iš duomenų saugyklos panaudojant Java kalbos priemones. Dėl tokios prigimties ši biblioteka dažnai yra naudojama kartu su kitais įrankiais.

Apache Lucene privalumai:

- Rezultatų reitingavimas.
- Palaikomos įvairūs paieškų tipai: frazės, šablonai, loginė paieška, tikslumo, atstumo įvertinimas, sub-užklausos ir kt.
- Paieškos vykdymas tam tikruose laukuose.
- Rezultatų rikiavimas pagal bet kurį duomenų lauką.

- Kelių indeksavimo tipų palaikymas ir gautų rezultatų apjungimas.
- Tiesioginis indeksavimas.
- Realizacijos, skirtos kitoms programavimo kalboms [2].
- Lanksti struktūra leidžia plėsti paieškos variklį, suteikiant galimybę indeksuoti įvairaus tipo informaciją.

Trūkumai [7]:

- Reikia Java išmanymo norint naudotis.
- Nėra kitų kalbų palaikymo pirminėje distribucijoje.
- Reikalauja papildomo konfigūravimo.
- Nėra MySQL palaikymo pagal nutylėjimą.

3.3.4.1 Zend_Search_Lucene

Kaip buvo minėta, Lucene paieškos biblioteka turi ir kitų programavimo kalbų realizacijų. Viena iš jų – PHP kalbai skirta ir į „Zend Framework“ karkasą įtraukta Zend_Search_Lucene. Ši paieškos realizacija yra perėmusi visas pagrindines savo pirmtakės Apache Lucene savybes. Tai bendros paskirties tekstinės paieškos variklis, parašytas PHP 5 programavimo kalba. Indeksų lentelės yra saugomos tiesiogiai failų sistemoje ir nereikalauja duomenų bazės serverio. Ši paieška gali būti diegiama beveik kiekviename su PHP programavimo kalba sukurtame tinklapyje [16].

Zend Lucene realizacija turi visas „Zend Framework“ karkaso bibliotekoms būdingas savybes ir pilnai atitinka ZF standartus. Viena labiausiai pastebimų ZF ypatybių – objektiškumas, tad pati Lucene biblioteka yra klasių rinkinys. Kiekvieną veiksmą (indeksavimą, paiešką ir kitus) programuotojas gali keisti savo nuožiūra. Zend Lucene kiekvienas duomenų resursas taip pat yra aprašomas klasėmis. Dokumento klasėje nurodoma, kurie resurso laukai turi būti indeksuojami, transformuojami (pvz. tekstas išskaidomas į žodžius indeksuojant) ar ignoruojami. Ši biblioteka jau turi parašytas klases HTML, Word 2007, Powerpoint 2007 ir Excel 2007 dokumentams indeksuoti.

Kiekvienas paieškos rezultatas taip pat yra objektas, jis turi „id“ ir „score“ reikšmes, kurios atitinkamai saugo indekso identifikacinį numerį ir reitingavimo taškų kiekį. Visi rezultatai, kaip įprasta, surikiuoti taškų mažėjimo kryptimi. Tačiau programuotojui yra leidžiama pakeisti rikiavimo tipą kitu, pvz., nurodymu pradžioje rikiuoti pagal tam tikrą duomenų lauką, o tik po to pagal reitingą. Zend_Search_Lucene reitingavimui naudojami tokie patys algoritmai, kokie yra naudojami Apache Lucene bibliotekoje.

Paieškos vykdymui, kaip ir Apache Lucene, palaikomas didelis užklausų tipų kiekis. Programuotojas gali nesunkiai pakeisti užklausų sintaksę sukurdamas savo užklausų analizavimo klases. Į biblioteką nėra įtraukta jokia morfologinės teksto analizės klasė (vėlgi, suteikiama galimybė programuotojui pasirinkti norimą sprendimą). Biblioteka gali analizuoti tekstą atsižvelgdama arba ignoruodama didžiąsias ir mažąsias raides. Taip pat egzistuoja ignoruojamų žodžių sąrašo ir žodžių minimalaus ilgio filtravimo klasės. Nurodžius, kad indeksavimo mechanizmas naudotų šias klases, į indeksą nebus įtraukiami ignoruojami žodžiai ir žodžiai, kurie yra trumpesni nei nurodyta parametruose.

Zend Lucene privalumai:

- Rezultatų rikiavimas.
- Paieška pagal konkretų lauką (autorių, aprašymą, ...).
- Palaikomos įvairūs paieškų tipai: frazės, šablonai, loginė paieška, tikslumo ir atstumo įvertinimas bei kitos užklausos.
- Indeksų sąrašas suderinamas su Apache Lucene.

Trūkumai:

- Nepalaikomas dokumentų (Word, Excel, Powerpoint) indeksų atnaujinimas. Norint perindeksuoti – reikia dokumentą pašalinti iš indekso ir pridėti iš naujo.
- 32 bitų sistemose indekso dydis negali viršyti 2 Gb.
- Neveikia su tinklo failų sistemomis (NFS). Su dauguma Windows ir Unix failų sistemų turi veikti.
- PHP nustatymai (pavyzdžiui, maksimalus vykdymo atminties kiekis arba vykdymo laikas) gali riboti paieškos galimybes.
- Naudoja daug atminties, o tai gali būti svarbu vykdant PHP scenarijus.

4 Paieškos sistemos architektūra

Šiame darbe aprašoma paieškos sistema gali būti naudojama įvairiose interneto svetainėse, kuriose reikalingas paieškos sprendimas atrinkti rezultatus iš duomenų bazėje saugomų įrašų. Darbo metu sistema integruota į lietuviškų tinklaraščių (angl. *weblogs*) RSS srautų agregatorių. Paieškos variklis susideda iš trijų pagrindinių dalių:

1) informacijos rinkimas;
2) duomenų indeksavimas;
3) paieška.

4.1 RSS srautų agregatorius

Tinklapio paskirtis – periodiškai kaupti informaciją, pasirodančią lietuviškuose tinklaraščiuose. Surinkta informacija lankytojui pateikiama vienoje vietoje, todėl jam nereikia lankyti kelių svetainių. Tinklaraščiuose pasirodantys įrašai yra pateikiami RSS formatu. Agregatorius periodiškai nuskaito užregistruotų internetinių tinklapių RSS srautus ir į duomenų bazę įrašo naujai pasirodžiusius įrašus. Įrašai rikiuojami pagal datą, tad svetainėje apsilankiusiam lankytojui pateikiama „šviežiausia“ informacija. Agregatoriuje įdiegta paieška, leidžianti ieškoti informacijos duomenų bazėje saugomame įrašų archyve. Paieškos sistemai yra aktualūs šie svetainės ypatumai:

- **įrašų vertinimas** – lankytojui suteikiama galimybė vertinti įrašus teigiamai arba neigiamai. Vertinimų informacija saugoma duomenų bazėje;
- **paspaudimų skaičiavimas** – skaičiuojami paspaudimai, nukreipiantys vartotojus į originaliuose šaltiniuose esančius įrašus.

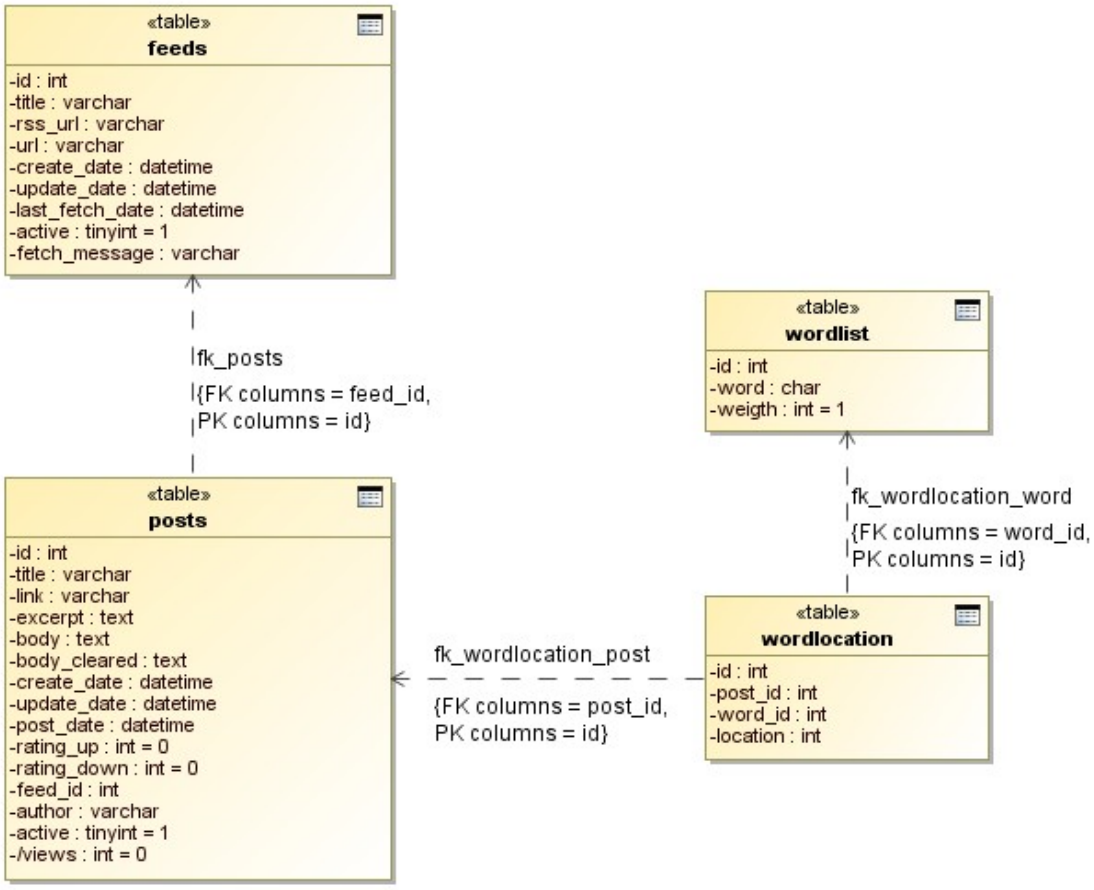
Atlikus paiešką, rezultatai gali būti pateikiami atsižvelgiant į aukščiau aprašytus lankytojų veiksmus.

4.2 Duomenų struktūra

1 pav. pateikiama pagrindinė RSS agregatoriaus duomenų struktūra, susijusi su paieškos varikliu. Diagramoje nėra pateikiamos duomenų bazės lentelės, egzistuojančios agregatoriaus tinklapyje, tačiau nesusijusios su paieškos sistema. Naudojamų lentelių aprašymai:

1 lentelė Duomenų bazės lentelių aprašymas

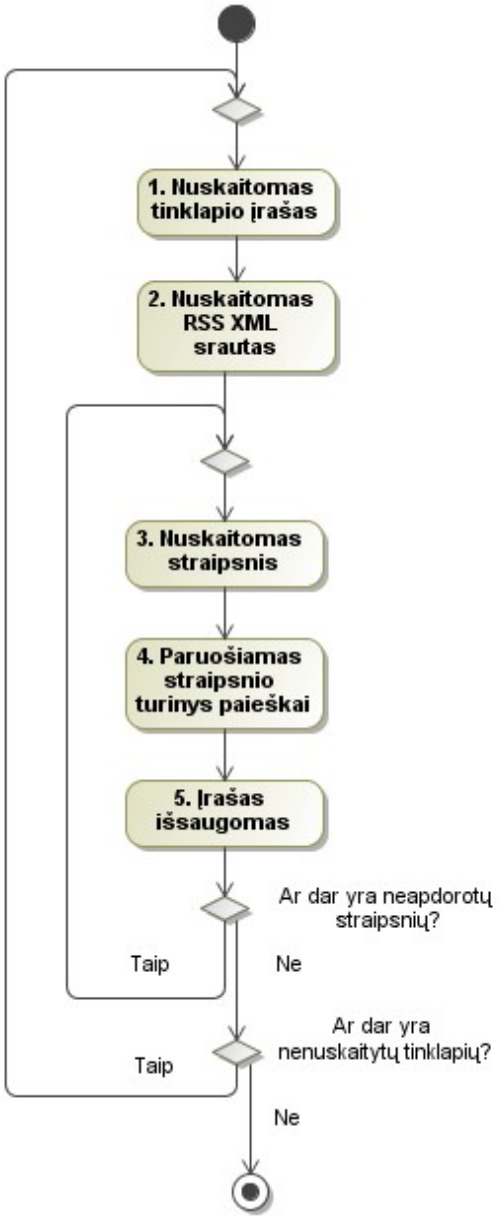
Lentelė	Aprašymas
Feeds	Saugomas stebimų internetinių puslapių sąrašas. Kiekvienas lentelės įrašas saugo tinklapio pavadinimą (laukas „title“) ir jo RSS srauto adresą (laukas „rss_url“).
Posts	Skirta saugoti iš RSS srautų nuskaitytai įrašų informacijai. Įrašai saugo nuskaitytą informaciją (laukas „body“), įrašo antraštę (laukas „title“), nuorodą į įrašą (laukas „link“), paspaudimų ant nuorodos kiekį (laukas „views“) ir vartotojų teigiamus bei neigiamus įvertinimus („rating_up“ ir „rating_down“ laukai). Kiekvienas šios lentelės įrašas priklauso konkrečiam tinklapiui iš lentelės „feeds“ ir su juo yra susietas išoriniu raktu (angl. *foreign key*). Paieška vykdoma įrašo turinio ir antraštės laukuose.
Wordlist	Skirta saugoti indeksavimo metu surinktiems žodžiams. Kiekvienas lentelės įrašas – atskiras žodis.
Wordlocation	Saugomi ryšiai tarp suindeksuotų žodžių ir įrašų. Vienas įrašas apibrėžia ryšį tarp konkretaus žodžio (susiejama su „worldlist“ lentele per išorinį raktą) ir konkretaus tinklapio įrašo (susiejama su „posts“ lentele per išorinį raktą). Įraše taip pat pateikiama žodžio pozicija tekste (laukas „location“).



1 pav. Duomenų bazės schema

4.3 Informacijos rinkimas

Informacija gali būti renkama iš įvairių šaltinių. Šiuo atveju paieškos sistema yra skirta RSS agregatoriui, tad duomenys nuskaitomi iš RSS srautų. Duomenų rinkimo schema pateikta 2 pav.



2 pav. Informacijos rinkimas iš RSS srautų į duomenų bazę

Informacijos rinkimas vykdomas 4 etapais:

1. Įš duomenų bazės lentelės „feeds“ (1 pav.) nuskaitomas internetinio tinklapio RSS srauto adresas. Šis veiksmas kartojamas visiems aktyviems lentelės įrašams.
2. Bandoma nuskaityti informaciją iš gauto RSS srauto adreso.
3. RSS srautas pateikia fiksuotą paskutinių tinklapyje paskelbtų įrašų kiekį. Nuskaitomas kiekvienas iš šių įrašų.
4. Nuskaitytas įrašas apdorojamas rašymui į duomenų bazę: atpažįstama įrašo paskelbimo data, pašalinami nereikalingi simboliai (teksto pradžioje ir pabaigoje esantys tarpai ir kt.).

5. Įrašas įrašomas į „posts“ lentelę (1 pav.). Jei lentelėje jau egzistuoja tokia naujiena, tai jos informacija atnaujinama, jei ne – sukuriamas naujas įrašas. Naujienos egzistavimas tikrinamas pagal RSS sraute pateiktą nuorodą į įrašą internetiniame tinklapyje.

4.4 Duomenų indeksavimas

Indeksavimo procesas išskaidytas į tris pagrindinius žingsnius (3 pav.):

1. Iš „posts“ lentelės (1 pav.) nuskaitomas įrašas, kurį ruošiamasi indeksuoti. Įrašas turi duomenų laukus, kuriuose saugomas indeksuojamas tekstas.
2. Tekstas išskaidomas į atskirus žodžius.
3. Išskaidyti žodžiai indeksuojami – žodžiai ir ryšiai tarp jų bei dokumentų surašomi į duomenų bazę.

3 pav. Indeksavimo procesas

Visa duomenų bazėje esanti informacija privalo būti suindeksuota, antraip vykdant paiešką nebus aptinkami į indeksą neįtraukti įrašai. Informacija atnaujinama nuolat, todėl ir indeksavimas turi būti vykdomas periodiškai. Kai įrašų kiekis didelis, indeksavimo laikas gali smarkiai išaugti. Kad sumažėtų resursų eikvojimas ir vykdymo trukmė, indeksuojami tik naujausi tam tikro periodo (pvz. 15 dienų) įrašai. Neigiamas šio pasirinkimo veiksnys – jei bus pakeista informacija sename įraše, duomenys indekse nebus atnaujinti. Tačiau resursų sąnaudų sumažinimas ir dažnesnis duomenų atnaujinimas yra svarbesni. Be to, RSS sraute

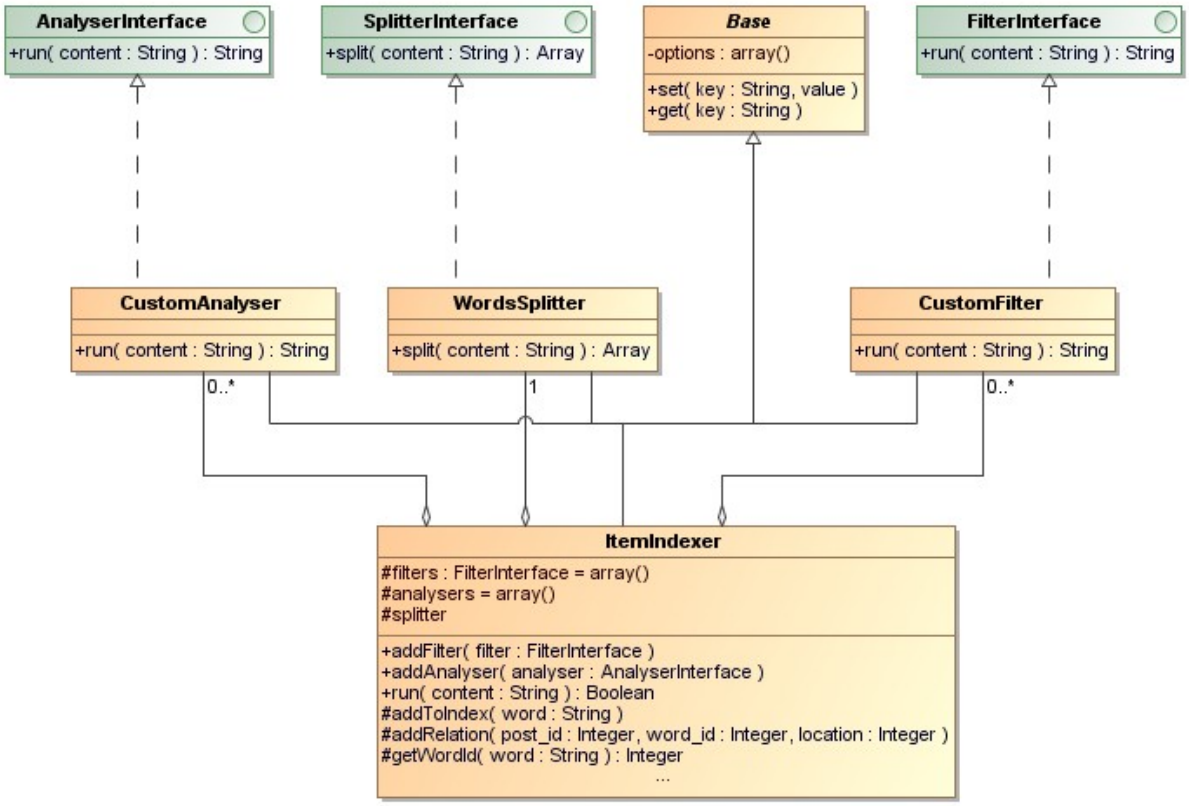
pateikiamas tik fiksuotas paskutinių įrašų kiekis (pvz., 10). Todėl būtų neefektyvu pakartotinai indeksuoti tuos įrašus, kurių informacijos jau negalima atnaujinti net jai ir pasikeitus (nebus nuskaityta iš RSS). Dar viena galimybė – gyvai indeksuoti įrašus nuskaitant juos iš informacijos šaltinių. Tačiau tokiu atveju smarkiai padidėtų informacijos nuskaitymo trukmė.

4.4.1 Indeksavimo modulio realizacija

Indeksavimo modulyje veiksmams su vienu duomenų įrašu yra realizuota klasė *ItemIndexer*. Jos veikimo diagrama pateikta 4 pav.

4 pav. ItemIndexer klasės veikimo diagrama

Indeksavimo modulio klasių diagramoje (5 pav.) pateikiama bendra jo struktūra. *CustomAnalyser*, *WordsSplitter* ir *CustomFilter* yra ne realios klasės, o jų pavyzdžiai ir pozicija hierarchijoje. Paieškos variklis leidžia modifikuoti indeksavimo procesą kuriant ir naudojant naujas filtrų, analizatorių ir teksto skaidymo klases, atitinkančias vartotojo poreikius. Pateiktas šio modulio elementų aprašymas (2 lentelė).



5 pav. Indeksavimo modulio klasių diagrama

2 lentelė Indeksavimo modulio struktūros klasės

Pavadinimas	Klasės tipas	Aprašymas
AnalyserInterface	Interfeisas	Apibrėžia analizatorių klasių bendrą struktūrą. Kiekvienas analizatorius privalo realizuoti šį interfeisą.
SplitterInterface	Interfeisas	Apibrėžia leksinės analizės klasių bendrą struktūrą. Kiekviena žodžių skaidymo klasė privalo realizuoti šį interfeisą.
FilterInterface	Interfeisas	Apibrėžia filtrų klasių bendrą struktūrą. Kiekviena žodžių skaidymo klasė privalo realizuoti šį interfeisą.
CustomAnalyser	Klasė	Analizatorių realizacija. Jie naudojami apdorojant atskirus žodžius prieš įrašant informaciją į duomenų bazę. Naudojami analizatoriai pateikti 4.4.4 sk.

Pavadinimas	Klasės tipas	Aprašymas
CustomFilter	Klasė	Filtrų realizacija. Filtrai taikomi įeinantiems duomenims apdoroti. Indeksavimui pateikiamą tekstą gali tekti reikiamai paruošti. Pavyzdžiui, iš RSS srautų ateinantys duomenys yra pateikiami HTML formatu. Prieš indeksuojant šią informaciją, reikia išvalyti turinį iš jo pašalinant. HTML žymas (žr. 4.4.2 sk.).
WordsSplitter	Klasė	Leksinės analizės realizacija. Šio pobūdžio klasės skirtos išskaidyti filtrais apdorotus duomenis į atskirus žodžius (žr. 4.4.3 sk.).
Base	Abstrakti klasė	Klasė, turinti parametrų nustatymo ir nuskaitymo galimybes. Naudojama klasėse, kurios gali būti konfigūruojamos nustatant parametrus.
ItemIndexer	Klasė	Indeksavimo veiksmus organizuojanti klasė. Ji gali turėti neribotą kiekį filtrų ir analizatorių objektų. Jie nustatomi iš išorės, konfigūruojant indeksavimo mechanizmą. Proceso metu, bus iškviečiamas kiekvienos iš šių klasių metodas *run()*. Objektai kviečiami ta tvarka, kuria jie buvo pridėti į indeksavimo klasę. Be jų, indeksavimo klasei privalo būti priskirtas teksto skaidymo į žodžius klasės objektas. Jis gali egzistuoti tik vienas, nes duomenys yra skaidomi tik vienu būdu.

Analizatoriai ir duomenų filtrai struktūriškai yra tos pačios klasės, tad vieni gali būti naudojami vietoje kitų. Filtrai daugiau skirti darbui su bendru turiniu (pvz. HTML kodo išvalymas iš eilutės), o analizatoriai – darbui su individualiais žodžiais (pvz. ignoruojamų žodžių pašalinimas). Tokie filtrai, kaip konvertavimas į didžiąsias ar mažąsias raides, gali būti naudojami abiem atvejais.

4.4.2 Indeksavimo sistemos duomenų filtrai

Realizuotoje sistemoje naudojami šių tipų duomenų filtrai:

1. HTML išvalymas. Iš RSS srautų gauta informacija yra HTML formato. Vykdant šį filtrą, pašalinamos HTML žymos ir paliekamas tik tekstinis turinys.

2. Konvertavimas į mažąsias raides. Pateiktas turinys konvertuojamas į mažąsias raides. Taip užtikrinama, kad skirtingais raidžių dydžiais parašyti tie patys žodžiai, bus traktuojami kaip tas pats žodis.

4.4.3 Leksinė analizė žodžių atskyrimui

Leksine analize vadinamas metodas, atpažįstantis atskirus žodžius pateiktame turinyje. Skaidant tekstą į atskirus žodžius, buvo keliami šie pagrindiniai reikalavimai:

- Žodžiai turi būti atskirti pagal visus tarpų simbolius (angl. *whitespace*).
- Turi būti pašalinti skiriamieji ženklai įskaitant brūkšnius, įvairaus tipo skliaustus, lietuviškas ir angliškas kabutes.
- Skaičiai atskirti kableliu arba tašku, neturi būti skaidomi į atskiras reikšmes, o turi būti laikomi kaip vienas žodis. Pavyzdžiui žodis „3.14“ neturėtų būti išskaidytas į žodžius „3“ ir „14“.

Skaidyti žodžius pagal šias taisykles yra naudojama žemiau pateikta reguliarioji išraiška:

```
/([\s\-_:;?!\/\(\)\[\]{}<>\r\n\"]|(?<!\d)[\.,](?!\d))/
```

Šios išraiškos trūkumas – iš anksto apibrėžti skyrikliai. Pasitaikius neapibrėžtai situacijai, žodžiai gali būti neatskirti arba nepašalinti nereikalingi simboliai. Tačiau tokių atvejų pasitaiko retai, ypač kai analizuojamuose tekstuose naudojama taisyklinga gramatika.

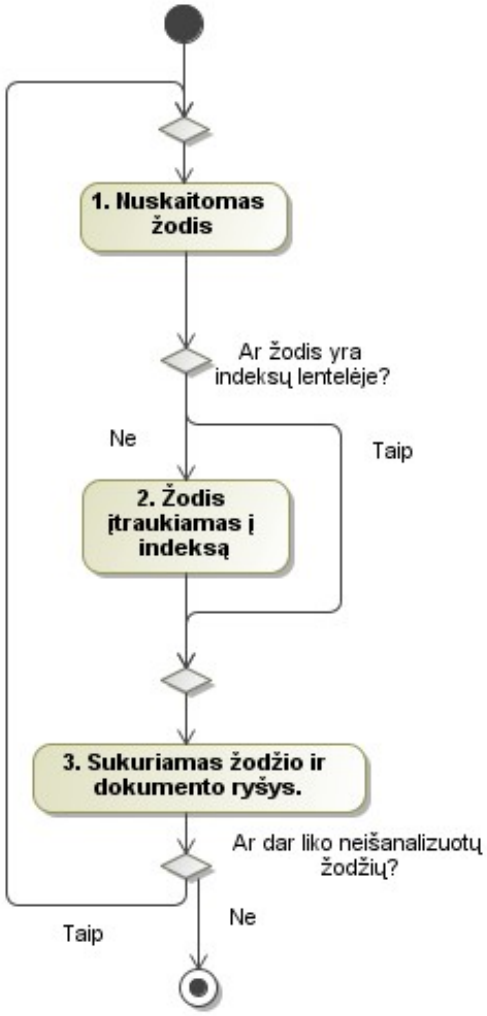
4.4.4 Indeksavimo sistemos analizatoriai

Indeksavimo metu naudojami šie analizatoriai individualiems žodžiams apdoroti ar atrinkti:

1. Ignoruojamų žodžių sąrašas (angl. s*toplist, stopwords).* Tai žodžiai, kurie neturi vertės indeksavimui. Ignoruojamų žodžių sąrašas arba kitaip – neigiamas žodynas (angl. *negative dictionary*) yra kompiuterio nuskaitomų žodžių sąrašas, kurie negali būti naudojami kaip indekso elementai [10]. Šiame sąraše esantys žodžiai yra ignoruojami indeksuojant duomenis. Dažniausiai į šio tipo sąrašus įtraukiami kalbos skyrikliai ir kiti bereikšmiai žodžiai, kurie nėra laikomi raktiniais žodžiais.
2. Minimalus indeksuojamų žodžių ilgis. Pro filtrą praleidžiami tik tie žodžiai, kurie yra nurodyto ilgio arba ilgesni. Taip galima atsikratyti dalies suindeksuotų žodžių, sutaupyti vietos ir resursų.

4.4.5 Žodžių įtraukimas į indeksą

Indeksuojamų žodžių įtraukimas į sąrašą pavaizduotas 6 pav. Pirmame žingsnyje iš indeksavimui pateikiamų žodžių nuskaitomas vienas elementas. Šie žodžiai jau yra atitinkamai apdoroti filtrais (žr. 4.4.2 sk.) ir analizatoriais (žr. 4.4.4 sk.), todėl papildomai jų paruošti nebereikia. Nuskaičius elementą, tikrinama, ar jis egzistuoja indeksų lentelėje. Jei egzistuoja – jo nereikia įtraukti pakartotinai, nes visi indekse esantys elementai privalo būti unikalūs. Jei žodžio nėra – jis įtraukiamas į indeksą. Galiausiai sukuriamas ryšys tarp dokumento ir žodžio, įtraukiant įrašą į ryšių lentelę „wordlocations“ (1 pav.).



6 pav. Žodžių įtraukimo į indeksą procesas

4.4.5.1 Indeksuojamų žodžių talpinimas laikinojoje atmintyje

Viena iš problematiškiausių vietų spartos atžvilgiu serverio pusės interneto aplikacijų programavime yra duomenų bazė. Tam, kad duomenų bazės apkrova sumažėtų, galima naudoti papildomą programinę įrangą ar technologijas [6]. Paieškos variklis realizuotas taip, kad būtų galima pasirinkti kokią laikinosios atminties (angl. *cache*) technologiją naudoti, priklausomai nuo naudojamos tarnybinės stoties galimybių. Ši funkcija realizuota naudojant

Zend_Cache klasę. Ji priklauso Zend Framework karkasui ir yra skirta dirbti su bet kokio tipo informacija: objektais, duomenų eilutėmis (angl. *string*), masyvais, funkcijomis klasėmis ir t.t. Zend_Cache leidžia pakeisti naudojamą technologiją keičiant parametrus, tačiau nemodifikuojant kodo [17]. Ji palaiko kelių tipų kešavimo sistemas: Sqlite, Memcached, Apc, Xcache, ZendPlatform, ZendServer ir File (duomenys saugomi failų sistemoje).

Siekiant sumažinti duomenų bazės apkrovimą taip pat reikia identifikuoti vietas, kuriuose yra nuskaitomi duomenys iš duomenų bazės. Kai duomenys yra sugeneruojami arba nuskaitomi iš duomenų bazės ir gali būti naudojami (ne iškarto) ateityje, generuoti juos per naują gali būti nuostolinga resursų atžvilgiu [4]. Šiuos duomenis reikia talpinti į atmintį. Pakartotinai prireikus duomenų, jų nereikės per naują nuskaityti iš duomenų bazės, o pakaks paimti iš laikinosios atminties. Indeksavime tokia vieta identifikuota tikrinant, ar žodis egzistuoja indekse. Indeksuojamuose tekstuose egzistuoja pasikartojantys žodžiai, todėl periodiškai yra tikrinama ar tas pats žodis jau egzistuoja indekse. Šioje vietoje vykdoma užklausa į duomenų bazės indeksų lentelę (1 pav.). Nuskaičius žodį pirmą kartą, jis yra patalpinamas į laikinąją atmintį, kurioje saugomas nurodytą laiko tarpą (sistemoje – 30 minučių). Kitą kartą pakartotinai ieškant to pačio žodžio yra patikrinama laikinoji atmintis.

4.5 Duomenų paieška

Paieškos proceso žingsnių aprašymas (7 pav.):

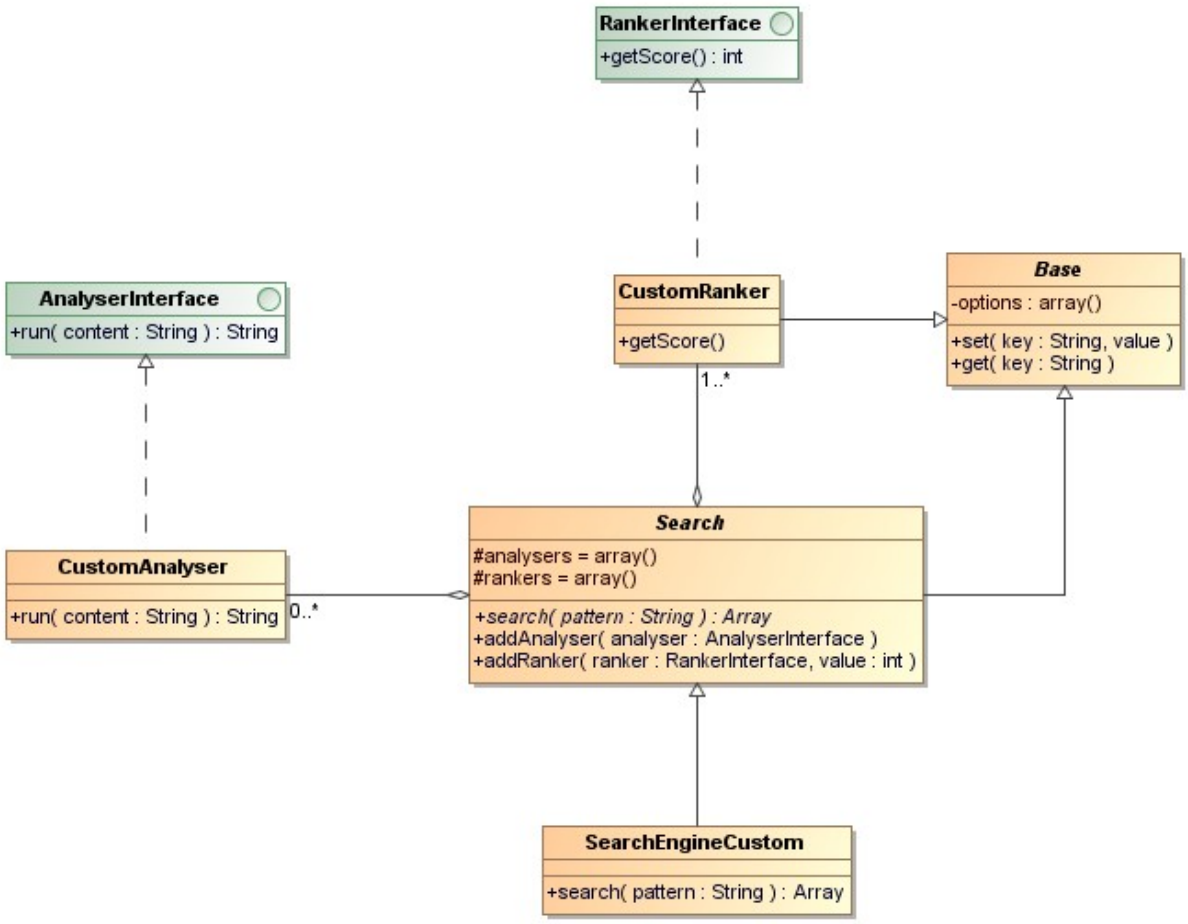
1. Vartotojas pateikia ieškomą užklausą paieškos varikliui. Užklausoje nurodomi ieškomi raktažodžiai, pagal kuriuos tikimasi gauti juos atitinkančius rezultatus.
2. Sistema, gavusi užklausą, ją analizuoja. Užklausos analizė priklauso nuo paieškos rūšies (žr. 3.2.3 sk.). Darbo metu realizuota paieškos rūšis veikia „bent vieno atitikmens“ principu. Vykdant šio tipo paiešką, atrinktuose rezultatuose privalo egzistuoti visi vartotojo pateiktoje užklausoje esantys žodžiai.
3. Užklausos raktiniams žodžiams pritaikomi analizatoriai. Šie analizatoriai veikia analogiškai kaip ir indeksavimo analizatoriai (žr. 4.4.1 sk.). Naudojami analizatoriai pateikti 4.5.2 sk.
4. Atsižvelgiant į paieškos rūšį, iš duomenų bazės atrenkami užklausą atitinkantys įrašai.
5. Gautiems rezultatams apskaičiuojami koeficientai, kurie parodo, kaip konkretus dokumentas atitinka vartotojo pateiktą užklausą. Tam naudojamos rezultatų reitingavimo funkcijos, realizuotos reitingavimo klasėse (žr. 4.5.3 sk.).

6. Galiausiai gauti rezultatai surikiuojami reitingavimo etape gautų koeficientų mažėjimo tvarka ir pateikiami vartotojui.

7 pav. Paieškos procesas

4.5.1 Paieškos modulio realizacija

Paieškos modulis, kaip ir indeksavimo, yra realizuotas taip, kad būtų galima nesudėtingai modifikuoti jo veikimą (paieškos vykdymą, reitingavimą ir žodžių analizę), pridedant naujas klases. Klasių diagramoje (8 pav.) pateikiama bendra klasių struktūra, atvaizduojanti jų pozicijas hierarchijoje. Klasės *CustomRanker*, *CustomAnalyser* ir *SearchEngineCustom* yra ne realios klasės, o jų pavyzdžiai. Realizuojant jų realius atitikmenis, privaloma paveldėti abstrakčias arba išplėsti interfeisines klases pagal anksčiau minėtą diagramą. Nubraižyta paieškos modulio klasių diagrama (8 pav.), toliau pateiktas elementų aprašymas (3 lentelė).



8 pav. Paieškos modulio klasių diagrama

3 lentelė Paieškos modulio struktūros klasės

Pavadinimas	Klasės tipas	Aprašymas
AnalyserInterface	Interfeisas	Apibrėžia analizatorių klasių bendrą struktūrą. Kiekvienas analizatorius privalo realizuoti šį interfeisą.
RankerInterface	Interfeisas	Apibrėžia reitingavimo funkcijų bendrą struktūrą. Kiekviena reitingavimo klasė privalo realizuoti šį interfeisą.
CustomAnalyser	Klasė	Analizatoriaus realizacija. Analogiška indeksavimo analizatoriams (2 lentelė).
CustomRanker	Klasė	Reitingavimo funkcijos realizacija. Šiai klasei perdavus ieškomus raktinius žodžius ir duomenų įrašų sąrašą yra paskaičiuojama skaitinė atitikimo užklausai reikšmė.

Pavadinimas	Klasės tipas	Aprašymas
Base	Abstrakti klasė	Klasė, turinti parametrų nustatymo ir nuskaitymo galimybes. Naudojama klasėse, kurios gali būti konfigūruojamos nustatant parametrus. Ta pati klasė kaip ir indeksavimo modulyje (2 lentelė).
Search	Abstrakti klasė	Paieškos pagrindinių funkcijų realizavimas. Klasėje realizuotos galimybės pridėti naudojamus analizatorius ir reitingavimo funkcijas. Ši klasė turi būti paveldima kuriant paieškos variklį. Klasė gali turėti neribotą kiekį analizatorių ir vertinimo funkcijų objektų. Pastarųjų turi būti bent viena.
SearchEngineCustom	Klasė	Paieškos variklio realizacija. Klasėje apibrėžiama vartotojo užklausos analizavimo ir rezultatų atrinkimo logika. Naudojant šios klasės objektus vykdoma paieška.

4.5.2 Paieškos sistemos analizatoriai

Kaip ir indeksavimo modulyje (4.4 sk.), analizatorių paskirtis – paruošti paieškos užklausoje esančius žodžius paieškai. Kadangi indeksuojant duomenis žodžiai taip pat yra atitinkamai apdorojami, tad tikslinga tiek indeksavimo, tiek ir paieškos metu naudoti tuos pačius analizatorius. Indeksavimo metu naudojami analizatoriai aprašyti 4.4.4 sk.

4.5.3 Rezultatų reitingavimas

Realizuotam paieškos varikliui programuotojas gali priskirti neribotą kiekį įvairių reitingavimo funkcijų, kurios realizuotos atskiromis klasėmis. Konkreti funkcija išnagrinėja paieškos variklio atrinktus rezultatus ir juos įvertina pagal funkcijos specifiką. Kad būtų galima palytinti skirtingų vertinimo metodų rezultatus, juos reikia normalizuoti. Vienu atveju didesnė reikšmė gali reikšti geresnį rezultatą, kitu – blogesnį [11]. Grąžinama normalizuota reikšmė patenka į intervalą nuo 0 iki 1. Kuo reikšmė arčiau 1, tuo geresnis rezultatas. Kiekviena iš šių klasių yra pridedama paieškos varikliui, kartu pateikiamas svarbos koeficientas. Šis koeficientas nurodo, kokią vertę bendram rezultatui turi konkreti funkcija. Koeficientų dydžiai yra reliatyvūs vienas kito atžvilgiu – nėra maksimalios galimos vertės,

tad funkcijų vertinimas vienas kito atžvilgiu koeficientais 1 ir 2 bus analogiškas vertinimui 2 ir 4. Galutinė įrašo vertė skaičiuojama pagal (1) formulę:

$$K = \sum_{i=1}^{n} a_i k_i \quad (1)$$

čia K – įrašo bendra vertė; n – reitingo skaičiavimo funkcijų kiekis; a – funkcijos įvertinimas; k – funkcijai priskirtas koeficientas.

4.5.3.1 Paieškos sistemos reitingo skaičiavimo funkcijos

Surikiuoti rezultatams, paieškos sistemoje naudojamos reitingo skaičiavimo funkcijos (4 lentelė).

4 lentelė Paieškos sistemoje naudojamos reitingo skaičiavimo funkcijos

Funkcija	Aprašymas
Žodžio dažnumas	Analizuojant duomenų sąrašą yra apskaičiuojama kiek kartų duomenyse pasikartoja ieškomas žodis. Pasikartojimai vertinami pateiktų įrašų atžvilgiu. Jei visuose įrašuose žodis pasikartoja po 1 kartą, tai grąžinama reikšmė bus lygi 1. Jei įrašuose žodis pasikartoja 1 ir 2 kartus, tai atitinkamai įrašų vertės bus 0,5 ir 1.
Pozicija dokumente	Vertinama žodžio pozicija dokumente. Kuo žodis yra arčiau dokumento pradžios, tuo grąžinamas rezultatas yra aukštesnis.
Žodžių atstumas	Jei pateiktoje užklausoje yra daugiau nei vienas žodis, vertinamas žodžių atstumas tarpusavyje. Remiantis šia vertinimo funkcija laikoma, kad jei žodžiai yra greta vienas kito, labiau tikėtina, kad analizuojamas rezultatas atitinka vartotojo pateiktą užklausą. Jei vartotojo užklausoje pateikiamas vienas žodis, grąžinamas rezultatas visiems duomenims yra lygus 1 ir neįtakoja reitingavimo.
Peržiūrų kiekis	Aukščiau iškeliami įrašai, kurie lankytojų buvo peržiūrėti daugiau kartų.
Vertinimas	Lankytojai turi galimybę vertinti duomenų bazėje esančių straipsnių kokybę. Aukštesnį reitingą turintys įrašai pateikiami arčiau rezultatų sąrašo pradžios.

4.6 Galimi patobulinimai

Dabartinė paieškos sistema atlieka pagrindinę savo funkciją – leidžia surasti įrašus pagal paieškos užklausą ir pateikia rezultatus surikiuotus pagal vertinimo funkcijas (4 lentelė

). Tačiau siekiant pagerinti rastų rezultatų tikslumą ir padidinti veikimo spartą, galima paiešką tobulinti.

4.6.1 Indeksavimas

Indeksavimą galima patobulinti pridedant naujų filtrų ir analizatorių. Analizatorius, skirtas žodžio šaknies atskyrimui ir pavertimui į bendrinę formą (angl. *stemming*) leistų pagerinti ne tik indeksavimo ir paieškos spartą, bet ir randamų rezultatų tikslumą.. Jo dėka indekse esantys žodžiai būtų paverčiami į bendrinę formą, todėl paieškos metu atrenkamų rezultatų neįtakotų skirtingomis formomis vartotojo pateikti ieškomi raktažodžiai. Kadangi žodžiai būtų saugomi bendrinėje formoje, sumažėtų indeksų lentelės įrašų kiekis, o tai turėtų įtakos indeksavimo ir paieškos spartai. Tačiau šis metodas yra skirtas tik konkrečiai kalbai. Jei tekste pasitaikytų kitų kalbų žodžių, jie būtų įtraukiami į indeksą ne bendrinėje formoje, o tokioje, kokioje pateikti. Todėl jei indeksuojamas turinys gali būti kelių kalbų, tektų taikyti kalbos atpažinimo metodus arba naudoti formos keitimą taikyti tik tai kalbai, kurios turinio yra daugiausiai.

Kitas galimas filtras – klaidų tikrinimas. Naudojant automatinio taisymo žodyną, galima būtų taisyti tiek indeksuojamame tekste, tiek ir vartotojo įvestoje užklausoje padarytas klaidas. Pastebimos įtakos indeksavimo ir paieškos spartai šis metodas neturėtų, tačiau galėtų pagerinti atrenkamų rezultatų tikslumą.

4.6.2 Paieška

Šiuo metu realizuota tik visų atitikmenų paieška. Jei užklausoje esantis raktažodis neegzistuoja indeksų lentelėje, jis yra ignoruojamas. Tačiau, jei raktažodis egzistuoja, ir nėra dokumento, kuriame šis žodis būtų kartu su kitais užklausoje pateiktais elementais, nei vienas rezultatas nebus grąžintas. Kitaip tariant, visi užklausoje pateikti žodžiai, jei jie egzistuoja indeksų lentelėje, privalo egzistuoti dokumente, kad jis atsirastų atrinktų rezultatų aibėje. Paiešką būtų galima patobulint įtraukiant daugiau paieškos tipų (žr. 3.1 sk.). Pirmiausiai reiktų sukurti „bent vieno raktažodžio“ rezultatų atrinkimą. Tokiu atveju reiktų keisti reitingų skaičiavimų funkcijas, kad aukščiau būtų iškeliami tie rezultatai, kuriuose aptikta daugiau užklausoje esančių raktažodžių. Šio tipo paieškoje taptų prasminga sukurti reitingo funkciją, paremtą raktažodžių svorių skaičiavimu pagal jų kiekį indeksų lentelėje. Tam būtų galima naudoti BM25 [9] ar panašų metodą. Realizavus minėtą paieškos tipą, jo pagrindu būtų galima kurti loginės paieškos tipą. Teoriškai, naudojant šiuos tipus paieškos trukmė padidėtų, nes būtų naudojamos sudėtingesnės rezultatų atrinkimo užklausos ir padidėtų atrenkamų įrašų kiekis, tačiau rezultatai būtų atrenkami tiksliau, o paieška taptų lankstesnė.

5 Paieškos sistemos tyrimas

Sukurtas paieškos variklis yra sudarytas iš dviejų pagrindinių dalių: indeksavimo ir paieškos modulių. Šie du moduliai yra ganėtinai skirtingi, todėl vertinant jų efektyvumą turi būti matuojami skirtingi parametrai.

- Indeksavimas – svarbi duomenų surašymo sparta.
- Paieška – svarbus atrenkamų rezultatų atitikimas vartotojui.

5.1 Tyrimo duomenys ir techninė bei programinė įranga

Testavimo rezultatams įtakos turi naudojama techninė ir programinė įranga bei naudojami duomenys. Naudojant kitą įrangą, analogiški matavimai, naudojant tuos pačius duomenis, gali smarkiai skirtis. Norint išmatuoti spartą, matavimams reikia naudoti tuos pačius duomenis.

5.1.1 Techninė ir programinė įranga

Matavimai buvo atliekami naudojant nešiojamąjį kompiuterį (5 lentelė). Toliau pateikta naudota programinė įranga (6 lentelė).

5 lentelė Tyrimams naudota techninė įranga

Parametras	Reikšmė
Modelis	DELL Vostro 1400
Centrinis procesorius	Core 2 Duo T5470 (1.6Ghz, 2MB L2 Cache, 800MHz FSB)
Operatyvioji atmintis	2GB DDR2 667Mhz
Kietasis diskas	160GB 5400RPM

6 lentelė Tyrimams naudota programinė įranga

Parametras	Reikšmė
Operacinė sistema	Linux Ubuntu 10.04
Linux branduolys	2.6.32-22
Failų sistema	ext3
Gnome	2.30.0
MySQL	5.0.83
PHP	5.3.2-1
Apache	2.2.14
Zend Framework	1.1

Parametras	Reikšmė
Memcached	1.2.2

5.1.2 Tyrimo duomenys

Sukurta paieška skirta ieškoti tekstinės informacijos duomenų bazėse. Tai gali būti internetinės parduotuvės, naujienų portalai ar panašaus turinio šaltiniai, kuriuose informacija saugoma straipsnių ar įrašų pavidalu. Tyrimui naudoti duomenys yra paimti iš RSS srautų agregatoriaus projekto (žr. 4.1 sk.) duomenų bazės. Šis turinys yra HTML formato, todėl reikalauja išvalymo, pašalinant HTML žymas. Eksperimento metu iš šios duomenų bazės buvo atsitiktinai atrinktas nustatytas įrašų kiekis, taip sudarant prielaidą, kad duomenys bus iš įvairių šaltinių (t.y. tinklaraščių).

7 lentelė Tyrimo duomenų charakteristikos

Parametras	Reikšmė
Formatas	HTML
Vidutinis įrašo ilgis	183 žodis[1]
Šaltinių kiekis	509 tinklaraščiai
Įrašų kiekis	> 31000

5.2 Indeksavimas

Indeksavimas (žr. 3.2.2 sk.) gali būti ilgai trunkantis procesas. Tiriant jo veikimą, buvo siekiama sumažinti indeksavimo trukmę, nedarant esminių pakeitimų programos kode, tačiau naudojant numatytas galimybes, leidžiančias sumažinti SQL užklausų kiekį. Eksperimento metu išbandytas indeksavimas su skirtingais duomenų kiekiais (100, 500, 1000 ir 2500 įrašų), duomenų bazių tipais ir duomenų saugojimu laikinojoje atmintyje.

5.2.1 Indeksavimo tyrimui naudoti parametrai

5.2.1.1 Ignoruojamų žodžių sąrašas

Indeksavimas tiriamas dviem atvejais: naudojant ignoruojamų žodžių sąrašą (žr. 4.4.2 sk.) ir jo nenaudojant. Naudojant šį sąrašą buvo paimta 140 dažniausiai pasikartojančių žodžių iš analizuojamų duomenų. Dažniausiai pasikartojantys žodžiai gauti eksperimento metu suindeksavus 2500 įrašų ir iš suindeksuotų žodžių sąrašo paėmus minėtą kiekį dažniausiai pasikartojančių žodžių neįtraukiant pavadinimų ar kitų galimų reikšmingų žodžių (tokių kaip Facebook, Youtube ar Google). Tai dažniausiai jungtukai, prieveiksmiai, prielinksniai ir

1 Vidutinis įrašo ilgis apskaičiuotas išanalizavus 2500 įrašų, naudotų indeksavimui. Esant kitokiam duomenų kiekiui vidutinis įrašų ilgis gali skirtis.

dalelytės, kurie retai gali būti užklausos esminiais raktiniais žodžiais. Dalis dažniausiai pasikartojančių žodžių pateikta priede A.

5.2.1.2 Informacijos saugojimas laikinojoje atmintyje

Informacijos saugojimo atmintyje (žr. 4.4.5.1 sk.) tyrimui buvo pasirinkta naudoti „memcached“ sistemą. Jos paskirtis padidinti internetinių sistemų veikimo spartą sumažinant duomenų bazės apkrovimą. Memcached programa veikia kaip atskiras servisas. Dėl šios priežasties jis gali būti ne visada pateikiamas tarnybinėse stotyse, kuriose yra talpinamos svetainės. Memcached duomenys yra saugomi operatyviojoje atmintyje. Kad ankstesni veiksmai neturėtų įtakos rezultatams, tyrimo metu, prieš kiekvieną indeksavimą laikinoji atmintinė buvo išvaloma. Nustatytas įrašo saugojimo periodas – 30 minučių.

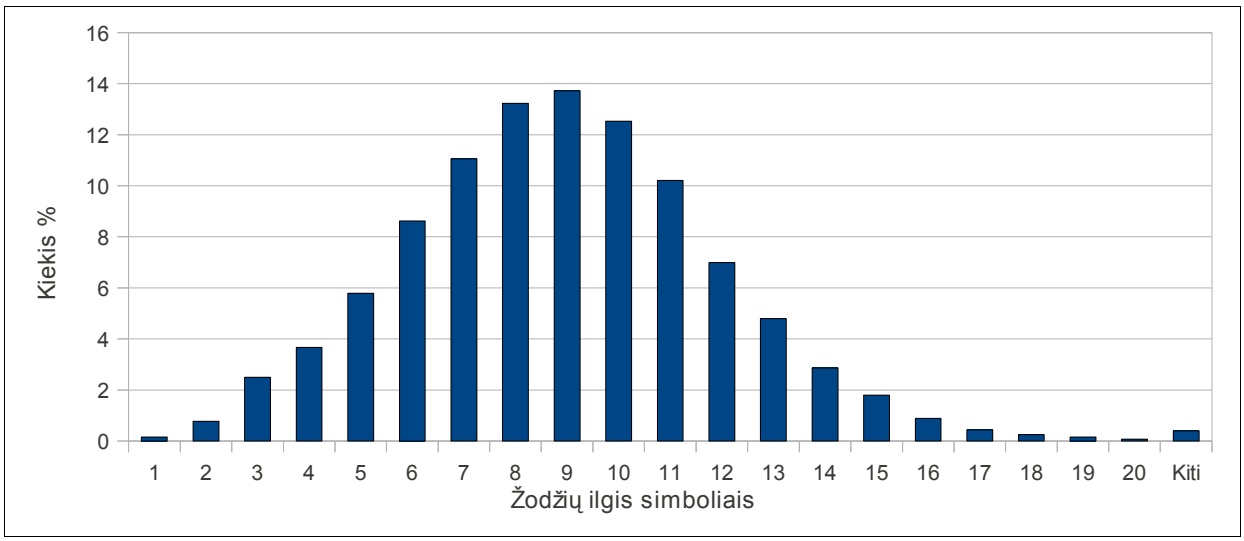
5.2.1.3 Duomenų bazės lentelių tipai

Tyrimo metu indeksavimo ir ryšių duomenys buvo saugomi dviejuose skirtinguose lentelių tipuose:

- MyISAM – pagal nutylėjimą MySQL DBVS pateikiamas tipas. Palaiko indeksus.
- InnoDB – MySQL DBVS palaikomas tipas. Palaiko transakcijas, išorinius raktus, indeksus.

5.2.1.4 Minimalus indeksuojamo žodžio simbolių kiekis

Šis parametras yra filtro reikšmė (žr 4.4.2 sk.), nurodanti minimalų žodžio ilgį, kuris gali būti įtraukiamas į indeksą. Šis parametras nebuvo keičiamas tyrimo metu ir yra lygus 3. Tokia reikšmė pasirinkta siekiant sumažinti indeksuojamų žodžių kiekį, tačiau neatmetant trumpų pavadinimų, kurių yra gana daug. Suindeksavus 2500 įrašų, kai minimalaus simbolių kiekio parametras lygus 0, buvo paskaičiuotas procentinis žodžių pasiskirstymas indeksų lentelėje pagal ilgį (9 pav.). Pagal gautus rezultatus, 2 simbolių ilgio žodžiai sudaro 0,77% visų žodžių. Tai šiek tiek mažiau nei 16 simbolių žodžiai (0,89%). Tačiau žodžiai, sudaryti iš dviejų simbolių, skirtingai nei kiti, pagrinde yra sudaryti iš skaitmenų arba yra nereikšmingi. Tuo tarpu 3 simbolių žodžiai sudaro jau 2,5%. Čia žodžiu laikomas simbolių junginys, atskirtas leksinės analizės metu (žr. 4.4.3 sk.). Žodžių pasiskirstymas pagal ilgį pateikiamas priede A.



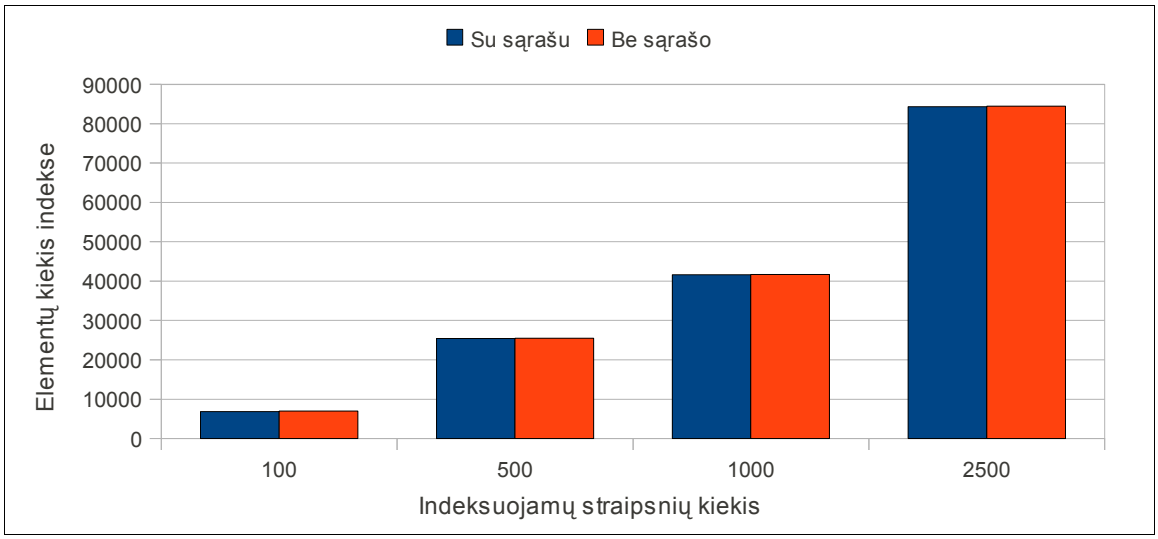
9 pav. Žodžių procentinis pasiskirstymas indeksų lentelėje.

5.2.2 Indeksavimo spartos analizė

Indeksavimas buvo vykdomas su įjungtu ir išjungtu ignoruojamų žodžių sąrašu. Ignoruojamų žodžių sąrašą sudarė 140 populiariausių žodžių (žr. 5.2.1.1 sk.). Šis kiekis paimtas nuo didžiausio tirto įrašų kiekio indeksų lentelės, todėl prie mažesnių tiriamų įrašų, aptiktų ignoruojamų žodžių kiekis buvo mažesnis.

8 lentelė Suindeksuotų žodžių kiekiai

	Įrašų kiekis			
	100	**500**	**1000**	**2500**
Be sąrašo	6959	25502	41713	84484
Su sąrašu	6850	25406	41623	84344

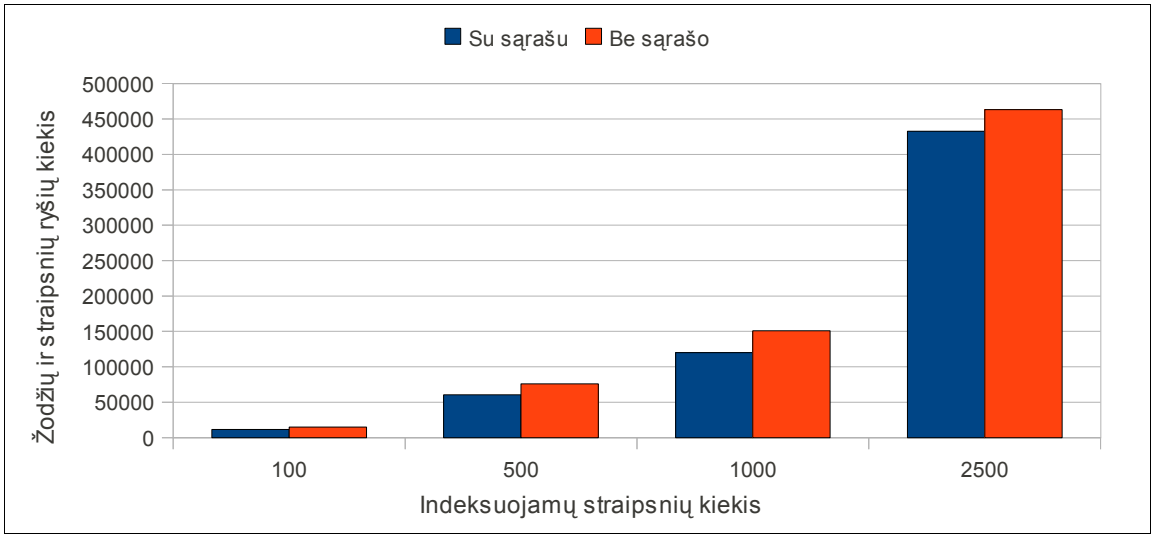


10 pav. Suindeksuotų žodžių kiekių palyginimas naudojant ir nenaudojant ignoruojamų žodžių sąrašus.

Ignoruojamų žodžių sąrašo įjungimas turi didelę įtaką sąryšių įrašų kiekiui. Į šį sąrašą atrenkami populiariausi žodžiai, todėl sąryšių tarp dokumentų ir žodžių kiekis tiesiogiai susijęs su sąrašo dydžiu.

9 lentelė Žodžių ir dokumentų sąryšių kiekiai

	Įrašų kiekis			
	100	**500**	**1000**	**2500**
Be sąrašo	14885	76054	151035	463330
Su sąrašu	11802	60463	119966	432704
Sumažėjimas, %	21%	20%	21%	7%



11 pav. Dokumentų ir indekso elementų kiekių palyginimas su ir be ignoruojamų žodžių sąrašu.

Atlikus tyrimą paaiškėjo, kad ignoruojamų žodžių kiekis sudaro labai mažą dalį visų indekse esančių elementų (8 lentelė ir 10 pav.). Šis sąrašas sumažina sąryšių lentelės elementų kiekį 7%-21% (9 lentelė). Tiriami įrašų kiekiai nėra parinkti tolygiai. Todėl norint įvertinti vidutinį sąryšių lentelės įrašų sumažėjimą, kiekvienam duomenų kiekiui reikia priskirti atitinkamą svorį (10 lentelė).

10 lentelė Svorio koeficientai

	Įrašų kiekis			
	100	**500**	**1000**	**2500**
Dalis, %	0.024	0.122	0.244	0.61

Atitinkamus sąryšių kiekių procentinius sumažėjimus padauginus iš parinktų svorių koeficientų gauta, kad naudojant 140 ignoruojamų žodžių sąrašą, sąryšių lentelės elementų kiekis vidutiniškai sumažėjo 12.33%.

Tyrimo metu buvo matuojama vidutinė indeksavimo sparta. Sparta matuota trimis etapais. Kiekvieno etapo matavimai atlikti po tris kartus, naudojant skirtingus duomenų kiekius. Tyrimo etapai:

1. Nenaudojant laikinosios atminties, nenaudojant ignoruojamų žodžių sąrašo.
2. Naudojant laikinąją atmintį, nenaudojant ignoruojamų žodžių sąrašo.
3. Naudojant laikinąją atmintį ir ignoruojamų žodžių sąrašą (140 elementų).

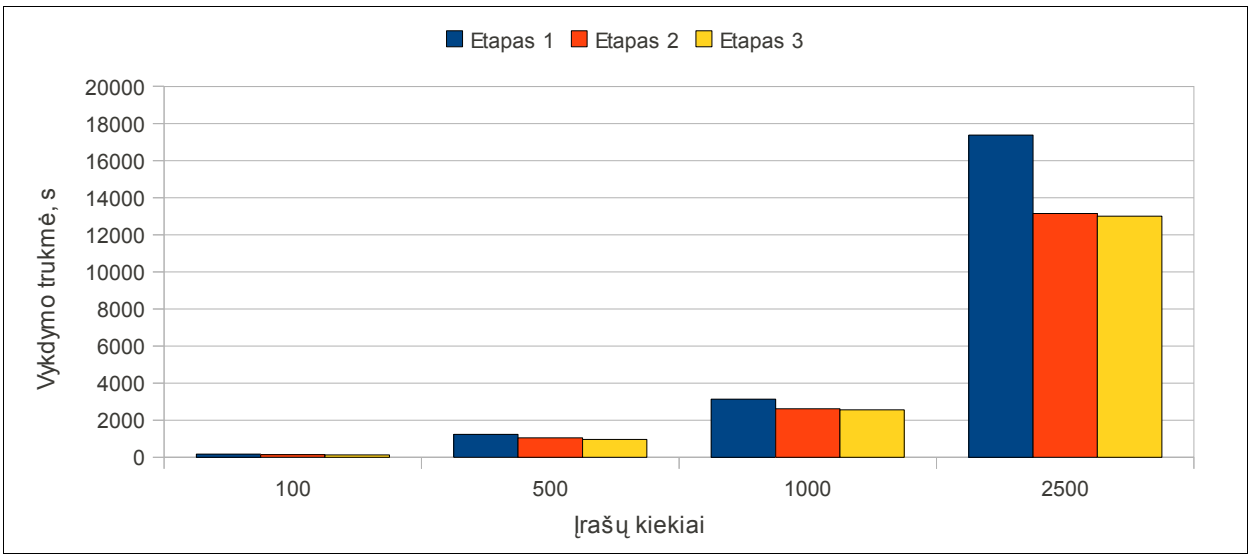
Pirmajame spartos matavimo etape gauti rezultatai buvo laikomi baziniu variantu, kurį siekta pagerinti. Antrajame etape buvo įjungtas informacijos saugojimas laikinojoje atmintyje, taip paspartinant pakartotinį duomenų nuskaitymą. Indeksavimo metu pagrinde vykdomas duomenų surašymas į duomenų bazę, o ne skaitymas iš jos. Todėl buvo laukta nežymaus spartos padidėjimo. Atliktas tyrimas patvirtino spėjimus, kad duomenys indeksuojami sparčiau. Trečio etapo metu buvo naudojama laikinoji atmintis ir pripildytas ignoruojamų žodžių sąrašas. Šio sąrašo naudojimas sumažina sąryšių lentelės elementų kiekį (11 pav.), tai reiškia, kad sumažėja įrašymų į duomenų bazę skaičius. Dėl to buvo tikimasi spartos padidėjimo. Indeksavimas buvo vykdomas duomenis saugant InnoDB ir MyISAM tipų lentelėse.

11 lentelė InnoDB indeksavimo sparta, s

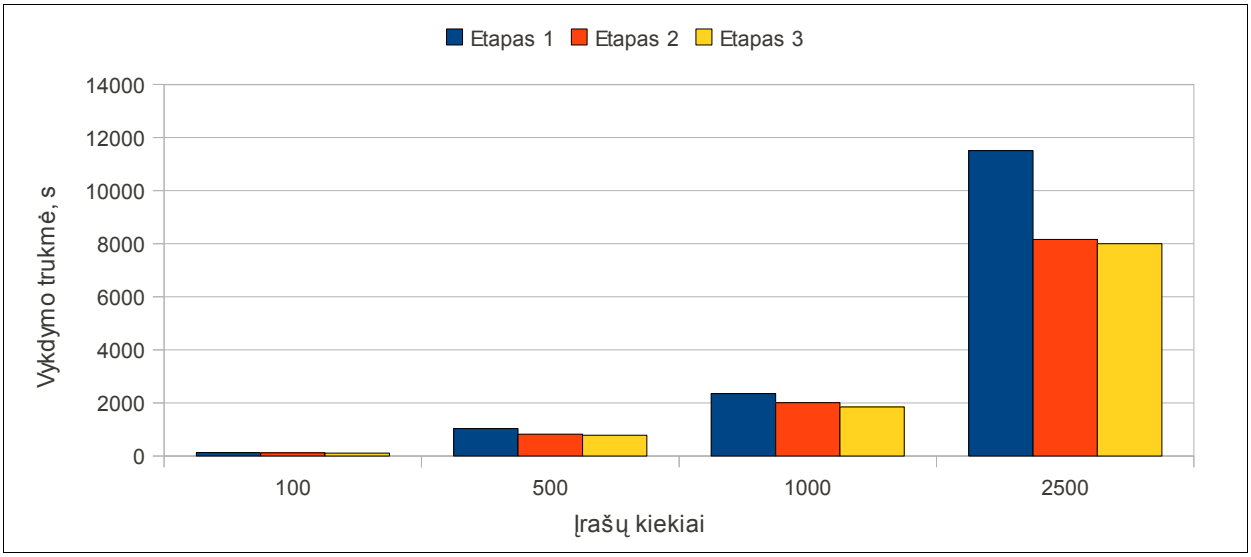
Laikinoji atmintis	**Ign. žodžių sąrašas, vnt.**	**Įrašų kiekis**			
		100	**500**	**1000**	**2500**
išjungta	0	166 s	1239 s	3142 s	17407 s
įjungta	0	149 s	1046 s	2620 s	13164 s
įjungta	140	129 s	961 s	2560 s	13014 s

12 lentelė MyISAM indeksavimo sparta, s

Laikinoji atmintis	**Ign. žodžių sąrašas, vnt.**	**Įrašų kiekis**			
		100	**500**	**1000**	**2500**
išjungta	0	133 s	1035 s	2354 s	11508 s
įjungta	0	119 s	824 s	2007 s	8162 s
įjungta	140	107 s	786 s	1849 s	8002 s



12 pav. InnoDB indeksavimo trukmės palyginimas skirtingais etapais.



13 pav. MyISAM indeksavimo trukmės palyginimas skirtingais etapais.

Naudojant ignoruojamų žodžių sąrašą, spartos padidėjimas nėra ženklus. Tai ypač atsispindėjo indeksuojant 2500 įrašų turintį duomenų komplektą – šiame žingsnyje jo sparta padidėjo tik 1% naudojant InnoDB lentelių tipą (13 lentelė) ir 2%, naudojant MyISAM. Tokį mažą spartos padidėjimą paaiškina tik 7% sumažėjusi sąryšių dalis, kai prie kitų duomenų kiekių ši dalis siekė 20 ar 21% (9 lentelė). Sąryšių sumažėjimas dirbant su 2500 įrašų yra mažesnis 3 kartus, tačiau spartos padidėjimas mažesnis net 1-12% (InnoDB) ir 1-9% (MyISAM). Tai parodo, kad dirbant su dideliais duomenų kiekiais, rašymo ir skaitymo operacijų trukmė mažesnė, nei dirbant su nedidelėmis duomenų bazių lentelėmis.

13 lentelė InnoDB indeksavimo spartos pokyčiai, %

Perėjimai tarp etapų	Įrašų kiekis			
	100	**500**	**1000**	**2500**
Iš 1 į 2	10%	16%	16%	24%
Iš 2 į 3	12%	7%	2%	1%
Iš 1 į 3 (bendras)	22%	23%	18%	25%

14 lentelė MyISAM indeksavimo spartos pokyčiai, %

Perėjimai tarp etapų	Įrašų kiekis			
	100	**500**	**1000**	**2500**
Iš 1 į 2	11%	20%	15%	29%
Iš 2 į 3	9%	4%	6%	1%
Iš 1 į 3 (bendras)	20%	24%	21%	30%

Iš indeksavimo spartos pokyčių (13 lentelė ir 14 lentelė) buvo pastebėta, kad įjungus laikinosios atminties naudojimą, sparta didėjo kartu augant ir indeksuojamų žodžių kiekiui. Iš to galima daryti prielaidą, kad kuo didesnis įrašų kiekis, tuo daugiau įtakos jam turi laikinosios atminties naudojimas. Tai dar kartą patvirtina anksčiau minėtą faktą, kad dirbant su daug įrašų turinčiomis duomenų bazės lentelėmis, operacijų trukmė auga. Dėl to duomenis, kurių gali prireikti pakartotinai, pravartu saugoti laikinojoje atmintinėje. Tuo tarpu įjungus ignoruojamų žodžių sąrašo palaikymą, stebimas spartos sumažėjimas didėjant įrašų kiekiui (iš 2 į 3 etapą). Ignoruojamų žodžių procentinė dalis tekste išlieka panaši, todėl sąryšių kiekio sumažėjimas taip pat išlieka pastovus (pagal atliktus skaičiavimus – 12.33%). Tačiau augant duomenų kiekiui, skaitymo ir rašymo operacijų sparta mažėja. Todėl įtaka spartai taip pat mažėja. Augant duomenų kiekiui ir turint fiksuoto dydžio ignoruojamų žodžių sąrašą, ši išraiška artėja prie 0.

Tyrimo metu pastebėta, kad indeksavimas InnoDB lentelėse vyksta lėčiau, nei MyISAM. Apskaičiuota, kiek procentų InnoDB indeksavimas yra lėtesnis nei MyISAM (15 lentelė).

15 lentelė Indeksavimo spartos skirtumai skirtinguose lentelių tipuose InnoDB atžvilgiu

Etapas	Įrašų kiekis			
	100	**500**	**1000**	**2500**
1	20%	17%	25%	34%
2	20%	21%	24%	38%
3	17%	18%	28%	38%

Atsižvelgiant į svorių koeficientus (10 lentelė), gauti vidutiniai spartos padidėjimai naudojant abi technologijas ir skirtingus lentelių tipus (16 lentelė).

16 lentelė Vidutiniai indeksavimo spartos padidėjimai procentais 1 etapo rezultatų atžvilgiu

	InnoDB	**MyISAM**
2 Etapas	20,7%	24%
3 Etapas	2,2%	2,8%
Viso	22,9%	26,8%

Įvertinus įrašų kiekių svarbos koeficientus, paskaičiuota, kiek procentų indeksavimas InnoDB veikia lėčiau nei MyISAM.

17 lentelė Indeksavimo palyginimas procentais skirtinguose lentelių tipuose

	InnoDB lėtesnis už MyISAM
1 Etapas	32,96%
2 Etapas	36,6%
3 Etapas	37,42%
Vidurkis	35,66%

5.3 Paieška

Pilno teksto paieškos sprendimas leidžia vykdyti paiešką greičiau nei tiesiogiai analizuojant tekstą ir ieškant paieškos užklausą atitinkančių įrašų duomenų bazėje. Šio tyrimo metu išmatuota paieškos trukmė su skirtingo ilgio užklausomis ir naudojant InnoDB, bei MyISAM lentelių tipus.

5.3.1 Tyrimo duomenys ir metodas

Paieškos tyrimui naudota duomenų bazė, sauganti 2500 straipsnių, 463330 sąryšių įrašų (9 lentelė) ir 84484 elementų indeksų lentelėje (8 lentelė). Matavimai atlikti tik su didžiausiu turimu suindeksuotu duomenų kiekiu. Duomenų kiekis nuolat auga, be to paieška gali būti atliekama tik visoje duomenų bazėje, o ne jos dalyje. Įvertinus šiuos dalykus nuspręsta tyrimus atlikti su didžiausia įrašų aibe, stebint, kokią įtaką trukmei turi skirtingo ilgio užklausos ir duomenų lentelių tipai.

Paieškos spartos matavimui atsitiktiniu būdu buvo sugeneruoti užklausų sąrašai. Kiekvienas sąrašas sudarytas iš 1000 atsitiktinai sugeneruotų užklausų. Vienoje užklausoje yra nurodytas kiekis žodžių, kurie paimti iš indeksų lentelės. Taip užtikrinama, kad užklausose esantys žodžiai tikrai egzistuoja tarp duomenų, kuriuose vykdoma paieška. Sugeneruotas ieškomas frazes sudaro nuo 1 iki 5 žodžių. Viename sąraše saugomos vienodo

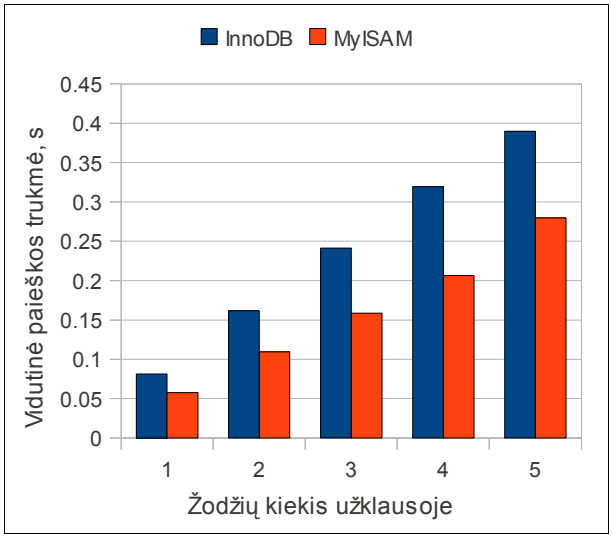
ilgio užklausos. Įvykdžius paieškas kiekvienam sąrašui, MySQL servisas buvo perkraunamas, kad ankstesni rezultatai neturėtų įtakos.

5.3.2 Paieškos spartos analizė

Buvo atlikti spartos matavimai (18 lentelė). Vykdant paiešką su skirtingais užklausų ilgiais, pastebėta, kad vykdymo laikai yra tiesiogiai proporcingi užklausoje esančiu žodžių kiekiams (14 pav.).

18 lentelė Paieškos vykdymo trukmė

	Vid. užklausos trukmė, s	
Žodžių skaičius užklausoje	**InnoDB**	**MyISAM**
1	0,0814	0,0577
2	0,1617	0,1094
3	0,2413	0,1588
4	0,3195	0,2064
5	0,3898	0,2797



14 pav. Paieškos vykdymo trukmės InnoDB ir MyISAM palyginimas.

Padalinus užklausos vykdymo trukmę iš žodžių kiekio, pastebėta, kad vidutinės vykdymo trukmės priklausomybė nuo žodžių kiekio yra tiesinė. Buvo apskaičiuoti trukmės augimo koeficientai abiems lentelių tipams (19 lentelė). Pagal šiuos koeficientus, galima nustatyti, kokia teoriškai gali būti vidutinė užklausos vykdymo trukmė vykdant paiešką su daugiau žodžių turinčiomis užklausomis. Turint kito dydžio duomenų aibę, šie koeficientai bus kitokie.

$$trukmės\ koeficientas = \frac{vid.\ vykdymo\ trukmė}{žodžių\ kiekis\ užklausoje} \quad (2)$$

19 lentelė Paieškos trukmės koeficientai

Žodžiai	InnoDB	MyISAM
1	0,0814	0,0577
2	0,0809	0,0547
3	0,0804	0,0529
4	0,0799	0,0516
5	0,0780	0,0559
Vidurkis:	0,0801	0,0546

Iš rezultatų matosi, kad paieška InnoDB lentelėse trunka ilgiau nei MyISAM lentelėse – šioje situacijoje InnoDB veikia 31,8% lėčiau.

6 Išvados

1. Indeksuojant lietuvišką turinį, tikslinga ignoruoti trumpesnius, nei trys simboliai žodžius. Vieno simbolio žodžiai yra nevertingi atliekant paiešką, o dviejų simbolių žodžiai, pagal ištirtus duomenis sudarantys 0,77% visų žodžių, daugiausiai yra nebūtinai reikšmingi raidžių ir skaičių junginiai.
2. Ignoruojamų žodžių sąrašas, kurį sudaro 140 elementų (dažniausiai pasikartojančių nereikšminių žodžių), sumažina ryšių lentelės įrašų kiekį nuo 7 iki 21%. Įvertinus duomenų kiekius, šios lentelės įrašų kiekis vidutiniškai sumažėjo 12,33%, vadinasi indeksavimas paspartėjo.
3. Indeksuojant duomenis su įjungtu ignoruojamų žodžių sąrašu, procentinis spartos pagerėjimas mažėja ir artėja prie 0, kai duomenų kiekis sąryšių lentelėje auga. Todėl prie didelių duomenų kiekių, šio sąrašo naudojimo įtaka indeksavimo spartai tampa nereikšminga.
4. Indeksuojant duomenis su įjungtu laikinosios atminties palaikymu, spartos pagerėjimas auga kartu su įrašų kiekio didėjimu sąryšių lentelėje. Tai patvirtina, kad skaitymo ir rašymo operacijų sparta su duomenų bazės lentelėmis mažėja, augant įrašų kiekiui. Kadangi duomenų kiekis linkęs augti, rekomenduojama visada naudoti laikinosios atminties spartinimą.
5. Realizuotoje sistemoje indeksavimas InnoDB lentelėse vyksta 35,66% lėčiau, nei MyISAM lentelėse, todėl siekiant didesnės spartos, šiam atvejui siūloma naudoti pastarojo tipo lenteles.
6. Naudojant ignoruojamų žodžių sąrašą ir laikinąją atmintį iš duomenų bazės nuskaitytiems rezultatams saugoti, indeksavimas InnoDB lentelėse pagreitėjo 22,9%, o MyISAM – 26,8%. Šie rezultatai parodo, kad minėtos priemonės gali ženkliai paspartinti indeksavimo procesą.
7. Indeksavimas – daug laiko ir resursų reikalaujantis procesas, todėl rekomenduojama pilną indeksavimą vykdyti retai ir analizuoti tik naujai įtrauktus duomenis.
8. Paieškos trukmė yra tiesiogiai proporcinga užklausoje esančių žodžių kiekiui. Ši priklausomybė yra tiesinė. Realizuotoje sistemoje, kai duomenys saugomi InnoDB lentelėse paieška vykdoma 31,8% lėčiau, nei tada, kai duomenys saugomi MyISAM.

7 Literatūra

1 *Apache Lucene - Overview* [interaktyvus]. 2009 [žiūrėta 2010-03-26]. Prieiga per internetą: <http://lucene.apache.org/java/docs/index.html>.

2 *Apache Lucene implementations* [interaktyvus]. 2010 [žiūrėta 2010-03-26]. Prieiga per internetą: <http://wiki.apache.org/lucene-java/LuceneImplementations>.

3 BAEZA, R. Y.; CASTILLO, C. *Soft Computing Systems : Design, Management and Applications: Balancing Volume, Quality and Freshness in Web Crawling.* University of Chile, 2002. ISBN 978-1-586-03297-5 .

4 DREPPER, U. *Whate Every Programmer Should Know About Memory*. Red Hat, Inc. 2007. Prieiga per internetą <http://people.redhat.com/drepper/cpumemory.pdf>

5 FOX, C. *Lexical analysis and stoplists.* Englewood Cliffs, NJ: Prentice Hall,.1992.

6 KEEFE, M. *Flash and PHP Bible.* Wiley Publishing, Inc., IN, 2008. ISBN 978-0-470-25824-8.

7 *MASHRAQI, F. Lucene and MySQL* [mokomoji medžiaga]. 2007. Prieiga per internetą: <http://www.slideshare.net/frankmashraqi/lucene-and-mysql>.

8 MOENS, M.F. *Automatic Indexing and abstracting of document texts.* Katholieke Universiteit Leuven, Belgium, 2000. ISBN 978-0-792-37793-1.

9 ROBERTSON, S.; ZARAGOZA H. *The Probabilistic Relevance Method: BM25 and beyond* [mokomoji medžiaga]. 2007. Prieiga per internetą: <http://www.zaragozas.info/hugo/academic/pdf/tutorial_sigir07_2d.pdf>.

10 SALTON, G. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* MA: Addison-Wesley, 1989,

11 SEGARAN, T. *Programming Collective Intelligence.* O'Reullt Media, Inc., 2007. ISBN 978-0-596-52932-1

12 *Snowball: A language for stemming algorithms* [interaktyvus]. 2001 [žiūrėta 2010-03-25]. Prieiga per internetą: <http://snowball.tartarus.org/>.

13 *Sphinx documentation - Intro* [interaktyvus]. 2009 [žiūrėta 2010-03-25]. Prieiga per internetą: <http://www.sphinxsearch.com/docs/current.html#intro>.

14 TIOBE *Programming Community Index for April 2010* [interaktyvus]. 2010 [žiūrėta 2010-04-21]. Prieiga per internetą: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>.

15 *Wikipedia: Size of Wikipedia* [interaktyvus]. 2010 [žiūrėta 2010-04-17]. Prieiga per internetą: < http://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia>.

16 *Zend Framework Documentation: Zend_Search_Lucene Introduction* [interaktyvus]. 2009 [žiūrėta 2010-03-26]. Prieiga per internetą: <http://framework.zend.com/manual/en/zend.search.lucene.overview.html>.

17 ZIMUEL, E. *Zend_Cache: how to improve performance of PHP applications.* [mokomoji medžiaga]. PHP Barcelona Conference, 2009. Prieiga per internetą: <http://www.slideshare.net/e.zimuel/zendcache-how-to-improve-the-performance-of-php-applications>.

A Priedas – Indeksuojamų žodžių analizė

Ignoruojami žodžiai buvo surinkti suindeksavus 10000 įrašų. Iš šių įrašų paimta 140 pirmų dažniausiai pasikartojančių žodžių, kurie nėra pavadinimai ar reikšmingi raktažodžiai, galintys figūruoti vartotojo užklausoje. Žemiau pateikta 60 dažniausiai pasikartojančių žodžių, gautų suindeksavus 2500 įrašų (A.1 lentelė).

A.1 lentelė Ignoruojamų žodžių sąrašas

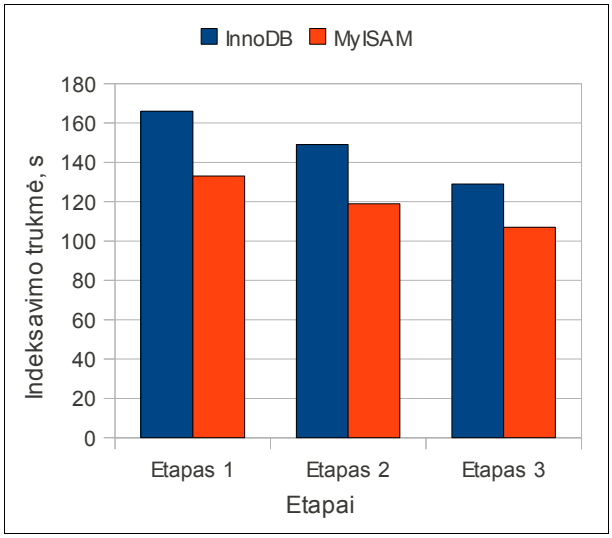
Žodis	Kiekis	Žodis	Kiekis	Žodis	Kiekis
kad	4591	the	1094	daugiau	744
&lt	3850	jis	1086	arba	716
amp	3761	čia	1080	nėra	710
tai	3485	nes	1078	dabar	702
yra	2905	net	978	turi	699
&gt	2555	nors	951	jie	675
tik	2273	man	932	ant	671
apie	1941	galima	898	gal	671
taip	1879	gali	897	bei	652
buvo	1869	bus	855	jog	635
dar	1864	prie	847	kur	623
jau	1770	tiek	815	todėl	614
savo	1762	mano	812	and	612
kas	1377	iki	802	reikia	609
labai	1302	nei	799	kurie	597
kai	1275	tikrai	796	kuri	592
nuo	1233	pat	794	būti	587
jei	1182	viena	783	būtų	585
per	1178	dėl	763	vis	582
tačiau	1110	daug	760	jos	570

A.2 lentelė Žodžių ilgių pasiskirstymas indekse išanalizavus 2500 straipsnių

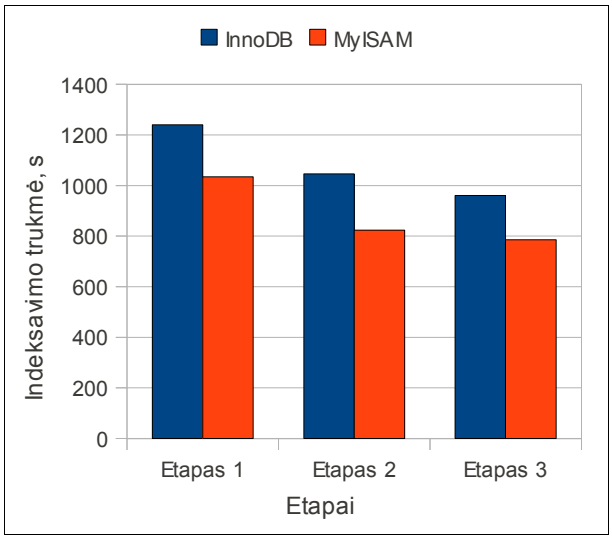
Žodžio ilgis	Kiekis, vnt.	Kiekis, %
1	130	0,15
2	654	0,77
3	2115	2,5
4	3100	3,64
5	4891	5,74
6	7285	8,54
7	9345	10,96
8	11180	13,11
9	11589	13,59
10	10582	12,41
11	8622	10,12
12	5906	6,93
13	4044	4,75
14	2414	2,83
15	1515	1,77
16	756	0,89
17	373	0,44
18	210	0,25
19	130	0,15
20	60	0,07
Kiti	330	0,39

B Priedas – InnoDB ir MyISAM trukmės palyginimo grafikai

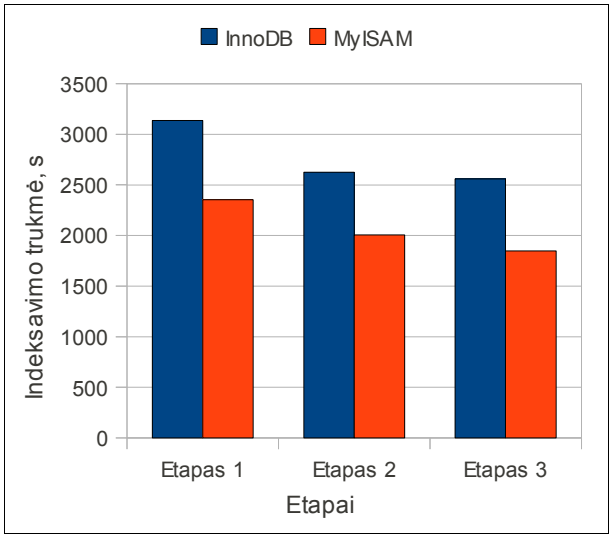
Žemiau pateikiami grafikai, atvaizduojantys indeksavimo trukmę, naudojant InnoDB (11 lentelė) ir MyISAM (12 lentelė) lentelių tipus.



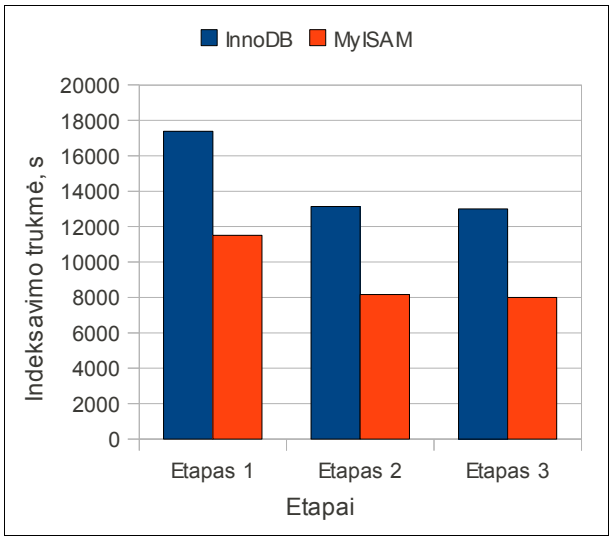
C.1 pav. InnoDB ir MyISAM trukmės palyginimas indeksuojant 100 straipsnių.



C.2 pav. InnoDB ir MyISAM trukmės palyginimas indeksuojant 500 straipsnių.



C.3 pav. InnoDB ir MyISAM trukmės palyginimas indeksuojant 1000 straipsnių.



C.4 pav. InnoDB ir MyISAM trukmės palyginimas indeksuojant 2500 straipsnių.

C Priedas - Iliustracijų sąrašas

D Priedas Priedas - Lentelių sąrašas