KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Eglė Mickevičiūtė

SPARQL užklausų vykdymo galimybių tyrimo metodika ir jos taikymas rekursinėms užklausoms

Magistro darbas

Darbo vadovas
prof. L. Nemuraitė

Kaunas, 2012

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Eglė Mickevičiūtė

SPARQL užklausų vykdymo galimybių tyrimo metodika ir jos taikymas rekursinėms užklausoms

Magistro darbas

Recenzentas
doc. dr. A. Lenkevičius
2012-05-28

Vadovas
prof. L. Nemuraitė
2012-05-28

Konsultantas
A. Šukys
2012-05-28

Atliko
IFM-0/4 gr. stud.
Eglė Mickevičiūtė
2012-05-24

Kaunas, 2012

Turinys

SUMMARY **4**

TERMINŲ IR SANTRUMPŲ ŽODYNĖLIS **5**

1. ĮVADAS **6**

2. ANALIZĖ **10**

2.1. TYRIMO SRITIS, OBJEKTAS IR PROBLEMA 10
2.2. SEMANTINIO TINKLO ŽINIŲ VAIZDAVIMO KALBŲ ANALIZĖ 10
2.3. SEMANTINIO TINKLO UŽKLAUSŲ KALBOS *SPARQL* ANALIZĖ 14
2.3.1. SPARQL palyginimas su SQL *14*
2.3.2. SPARQL 1.1 ..... *15*
2.3.3. SWRL taisyklės *16*
2.3.4. SPARQL užklausų klasifikacija *16*
2.4. SEMANTINIO TINKLO ĮRANKIŲ TYRIMO ETALONŲ ANALIZĖ 23
2.5. SIEKIAMO ETALONO KŪRIMO UŽDAVINIO FORMULUOTĖ 24
2.5.1. Siekiamos sistemos apibrėžimas *24*
2.5.2. Architektūros ir galimų įgyvendinimo priemonių variantų analizė *25*
2.6. ANALIZĖS IŠVADOS 29

3. SEMANTINIŲ UŽKLAUSŲ TYRIMO METODIKA **31**

3.1. TYRIMO METODIKA 31
3.2. GENEALOGINIO MEDŽIO ONTOLOGIJA TYRIMUI 31
3.3. UŽKLAUSŲ RINKINYS TYRIMUI 32

4. ONTOLOGIJOS EGZEMPLIORIŲ GENERAVIMO ĮRANKIO PROJEKTAS IR REALIZACIJA **34**

4.1. REIKALAVIMAI 34
4.1.1. Reikalavimų specifikacija *34*
4.1.2. Dalykinės srities modelis *39*
4.1.3. Reikalavimų apibendrinimas *40*
4.2. PROJEKTAS 40
4.2.1. Sistemos sprendimo pagrindimas ir esmės išdėstymas *40*
4.2.2. Sistemos architektūra *40*
4.2.3. Sistemos elgsenos modelis *43*
4.2.4. Duomenų bazės schema *49*
4.2.5. Detalus projektas *49*
4.2.6. Realizacijos modelis *52*
4.3. REALIZACIJA 54
4.3.1. Realizacijos ir veikimo aprašymas *54*
4.3.2. Testavimo modelis *58*
4.3.3. Testavimo duomenys ir rezultatai *58*

5. EKSPERIMENTINIS SISTEMOS TYRIMAS **60**

5.1. EKSPERIMENTO PLANAS 60
5.2. EKSPERIMENTO REZULTATAS 61
5.3. SISTEMOS VEIKIMO IR SAVYBIŲ ANALIZĖ, KOKYBĖS KRITERIJŲ ĮVERTINIMAS 70

6. IŠVADOS **73**

7. LITERATŪRA **74**

8. PRIEDAI **76**

1 PRIEDAS. STRAIPSNIS 76
2 PRIEDAS. SUGENERUOTI ONTOLOGIJOS INDIVIDAI 84

METHODOLOGY FOR INVESTIGATING SPARQL QUERYING CAPABILITIES AND ITS APPLICATION TO RECURSIVE QUERIES

SUMMARY

Semantic Web is Internet of the future, which is perceived as a global database where information is searched by its meaning rather than a textual representation. With the development of the Semantic Web new languages and tools emerge whose capabilities are interesting for users. In order to select a suitable technology so-called benchmarks are used. The aim of this research is to increase possibilities of evaluating ontology query engines by creating a dedicated methodology and implementing a tool to generate ontologies for evaluating capabilities of SPARQL 1.1 which were previously not analyzed by existing benchmarks. This research presents a classification of SPARQL queries; the ontology, which was chosen in order to evaluate recursive queries; a set of SPARQL queries under investigation along with a tool created for generating ontology examples for evaluation. The experiment was made to evaluate capabilities of Pellet query engine while executing the presented set of queries with and without using SWRL, and has revealed what improvement must be made in this area.

Key words: Semantic Web, ontology, benchmark, OWL 2, RDF, SPARQL 1.1.

Terminų ir santrumpų žodynėlis

1 lentelė Sąvokų sąrašas

Santrumpa	Sąvoka	Paaiškinimas/Vertimas
SPARQL	*Protocol and RDF Query Language*	*RDF* užklausų kalba, kuri taikoma ir *OWL 2* ontologijoms pateiktos tripletais, kurių formatas: „subjektas – predikatas – objektas“.
RDF	*Resource Description Framework*	Išteklių aprašymo sistema, kuri gali būti formalizuota nukreiptu, žymėtuoju grafu, kuris vaizduoja informaciją internete ir kurioje egzemplioriai yra aprašomi tripletais: „subjektas – predikatas – objektas“.
RDFS	*Resource Description Framework Shema*	Išplėsta *RDF*, kuri apibrėžia *RDF* schemą.
OWL 2	*Web Ontology Language*	Semantinio tinklo ontologijos kalba su formaliai apibrėžta semantika, praplečianti *RDFS*.
SWRL	*Semantic Web Rule Language*	Semantinio tinklo taisyklių kalba. Taikant *SWRL* taisykles ir išvedimo mechanizmus galima supaprastinti *SPARQL* užklausas.
XML	*Extensible Markup Language*	Duomenų saugojimo ir perdavimo formatas.
XSLT	*Extensible Stylesheet Language Transformations*	Deklaratyvi, *XML* pagrįsta kalba, naudojama *XML* transformavimui į kitus *XML* dokumentus.
XML Shema	*Extensible Markup Language Shema*	Apibūdina *XML* dokumento struktūrą.
OIL	*Ontology Interchange Language*	Pirmoji ontologijos vaizdavimo kalba.
DAML	*DARPA Agent Markup Language*	Semantinė kalba, *XML* ir *RDF* praplėtimas.
DBMS	*Database management system*	Duomenų bazių valdymo sistema.

1. Įvadas

Internetas – tai pats didžiausias informacijos tinklas pasaulyje, talpinantis labai daug informacijos, kuri dažnai yra perteklinė, tuo galima įsitikinti atliekant paieškas įvairiais paieškos varikliais. Tokioje gausybėje informacijos galima lengvai pasimesti, nerasti reikalingos informacijos arba tiesiog sugaišti nemažai laiko ieškant tinkamos informacijos, kartais tuomet, kai ji jau nebeaktuali. Tam, kad būtų galima gauti tikslią informaciją ir ja būtų galima pasitikėti, yra sukurta semantinių užklausų kalba *SPARQL*, kuri atlieka paiešką naudodama ontologijas. Priskiriant informacijos vienetams atitinkamus ryšius bei savybes – suteikiama galimybė pradėti įgyvendinti Semantinio tinklo viziją, kuri būtų taikoma įvairiose srityse.

Semantinis tinklas – tai terminas, kurį pirmas apibrėžė Tim Berners-Lee, ir kuris reiškia ateities interneto tinklą ir yra suvokiamas kaip globali duomenų bazė [3]. Semantinio tinklo technologijos padeda išskirti reikšmes iš duomenų (informacijos), dokumentų turinio, ar taikomųjų programų kodo, naudojant technologijas, kurios paremtos atviraisiais standartais. Jei kompiuteris suprastų dokumento semantiką, o ne tik interpretuotų jį kaip simbolių rinkinį, tuomet jis suprastų dokumento esmę, kam jis yra skirtas. Taigi, Semantinis tinklas turi bendrą struktūrą, kuri leidžia dalintis ir naudotis informacija taikomųjų programų, įmonės ar bendruomenės aplinkoje.

Semantinį tinklą būtų galima įsivaizduoti kaip vieną didelę globalią duomenų bazę, kurioje visa informacija turi prasmę ir kurią supranta Semantinio tinklo technologijos [9]. Jos naudojasi ontologijomis ir kitomis Semantinio tinklo technologijomis, pasinaudojus ontologijos ryšiais, savybėmis informacija gali būti randama pagal prasmę, o ne pagal tekstinį panašumą ar sutapimą.

Technologijos, kuriomis remiasi Semantinis tinklas yra *URIs* (*Uniform Resource Identifiers*), *RDF* (*Resource Description Framework*) *RDF Schema, OWL* (*Web Ontology Language*). *URIs* – globalių vardų schema, *RDF* – informacijos aprašymo standartinė sintaksė, išteklių aprašymo sistema, kuri gali būti formalizuota į nukreiptą, žymėtąjį grafą, kuris vaizduoja informaciją internete ir kurioje egzemplioriai yra aprašomi tripletais: subjektas – predikatas – objektas. *RDF Schema* – apibūdinantis informacijos savybes standartas, apibrėžia *RDF* savybių naudojimo taisykles ir pagrindinius komponentus. *OWL* – Semantinio tinklo ontologijos kalba su formaliai apibrėžta semantika. *OWL 2* – *OWL* kalbos vėlesnė ir patobulinta versija, praplėsta *RDFS* charakteringomis savybėmis ir klasėmis, suteikianti ontologijai didesnį išraiškingumą. *SWRL* – Semantinio tinklo taisyklių kalba. Taikant *SWRL* taisykles ir išvedimo mechanizmus galima supaprastinti *SPARQL* užklausas. Todėl šio darbo tyrimo sritis yra Semantinio tinklo kalbos: *OWL, RDF, SPARQL, SPARQL 1.1, SWRL*.

Tyrimo objektas – *SPARQL 1.1* užklausų sudarymo ir vykdymo procesas. Užklausų kalba *SPARQL* yra taikoma *OWL 2* ontologijoms ir iliustruoja Semantinio tinklo viziją. Kadangi ši

užklausų kalba yra besivystanti, o pasirodžius *SPARQL 1.1* užklausų kalbos versijai, trūksta informacijos apie *SPARQL* užklausų kalbos galimybes lyginant su kitomis etaloninėmis ontologijomis.

Šio darbo tikslas – padidinti ontologijų užklausų vykdymo priemonių įvertinimo galimybes, sudarant tam skirtą metodiką ir realizuojant įrankį, leidžiantį generuoti ontologijas *SPARQL 1.1* versijos galimybėms, kurių neapima esami *SPARQL* vykdymo priemonių tyrimo etalonai, tirti. Svarbu išsiaiškinti kaip užklausų vykdymo priemonės vykdo *SPARQL 1.1* užklausas su skirtingu egzemplioriu skaičiumi ir kokią įtaką užklausų kūrimui ir veikimui daro *SWRL* taisykles.

Darbo uždaviniai:

1. atlikti Semantinio tinklo kalbų ir ontologijų užklausų vykdymo priemonių vertinimo etalonų analizę;
2. sudaryti tyrimo metodiką ir *SPARQL 1.1* užklausų rinkinį;
3. suprojektuoti ir realizuoti įrankį, leidžiantį generuoti įvairių dydžių ontologijas;
4. atlikti eksperimentą, pritaikant sudarytą metodiką ir įrankį ontologijos užklausų vykdymo priemonėms įvertinti;
5. įvertinti gautus rezultatus.

Pasauliniame Semantiniame tinkle nuolat atsiranda naujų kalbų ir technologijų, kurių galimybes nori žinoti jas besirenkantys vartotojai. Norint pasirinkti tinkamą technologiją, joms palyginti taikomi taip vadinami etalonai (ang. *Benchmark*). Etaloną sudaro ontologija, generavimo įrankis ontologijos egzemplioriams generuoti ir užklausų rinkinys. Geriausiai žinomi ir taikomi etalonai, tokie kaip *LUBM*, *Berlin* ar *SP*^{2} nagrinėja Semantinio tinklo įrankių galimybes, *SPARQL* užklausų kalba – tai tik priemonė jiems analizuoti jų galimybes, tačiau tiriant Semantinio tinklo technologijas būtina išanalizuoti ir *SPARQL* užklausų kalbos galimybes.

Pasirodžiusi *SPARQL 1.1* versija buvo papildyta naujomis užklausų kalbos galimybėmis, kurias reikia ištirti norint žinoti Semantinio tinklo galimybes. Buvo sudaryta užklausų klasifikacija, į kurią įeina visos *SPARQL* užklausų kalbos galimybės.

Esamų etalonų analizė parodė, kad esami etalonai neapima naujų *SPARQL 1.1* galimybių ir ontologijos savybių, todėl buvo sukurtas naujas etalonas. Dėl rekursijos galimybės buvo pasirinkta genealoginio medžio ontologija. Pasinaudojus šiame darbe sudaryta *SPARQL* užklausų klasifikacija buvo sudarytas užklausų rinkinys tyrimui įtraukiant ir *SWRL* taisykles, kurios nebuvo įvertintos kaip užklausų optimizavimo galimybė ir nebuvo įtrauktos į užklausų sudarymą. Užklausų rinkinį sudaro 11 pagrindinių užklausų, 3 užklausos su *SWRL* taisyklėmis ir 2 *SWRL* taisyklės. Įrašų ir tripletų redagavimas bei šalinimas nebuvo įtraukti, kadangi šie veiksmai turėtų būti atliekami su tam pritaikytomis sąsajomis, o ne tiesiogiai per užklausas.

Pagrindiniam Semantinio tinklo technologijų *RDF*, *RDFS*, *OWL*, *SPARQL* supratimui buvo nagrinėti šaltiniai [13], [15], [18]. Tiriant plačiau Semantinio tinklo technologijas *RDF*, *RDFS*, *OWL* buvo remtasi literatūros šaltiniais [2], [1], kuriuose aprašytos pagrindinės savybės, pateikti pavyzdžiai, aprašyti *RDF* ir *RDFS* trūkumai. Kiti nagrinėti šaltiniai – [7], [6], [11] susiję su *SPARQL* užklausų kalba, jos skirtumai ir panašumai su *SQL* kalba, [19] – su naujomis *SPARQL 1.1* galimybėmis. *SWRL* kalbos samprata gauta remiantis [21] literatūros šaltiniu. *SPARQL* užklausų klasifikacija paremta [19], [20] literatūros šaltiniais. Etalonų analizė atlikta pasitelkus [10], [8], [4], [22] ir [17] literatūros šaltinius. Ontologijos redaktorių palyginimui nagrinėti [5], [16] literatūros šaltiniai, o išvedimo mechanizmų galimybių nustatymui – [14] literatūros šaltinis.

Šio darbo naujumas yra tai, kad sudarytas *SPARQL 1.1* užklausų rinkinys tyrimui, leidžiantis tirti naujų tipų užklausas, tarp jų rekursines, kurių neleidžia įvertinti esami Semantinio tinklo įrankių tyrimo etalonai, įtrauktos *SWRL* taisyklės jų įtakos užklausų kūrimui ir veikimui įvertinti.

Pasinaudojus sukurtu etalonu, buvo atliktas dviejų dalių eksperimentas su *Pellet* įrankiu nešiojamame kompiuteryje ir serveryje, kuris parodė, kad įrankio išvedimo biblioteka nėra optimizuota tam tikriems *SPARQL 1.1* užklausų tipams esant didesniam ontologijos egzempliorių skaičiui. Sukurtas etalonas leido įvertinti iki šiol nenagrinėtų užklausų tipų veikimą ir atskleidė, kokių tipų užklausoms ir *SWRL* taisyklėms tikslinga tobulinti *Pellet*.

Darbo struktūra:

- Antrame skyriuje apibrėžta tyrimo sritis, objektas ir problema, atlikta Semantinio tinklo žinių vaizdavimo kalbų analizė, įvardintos jų savybės, trūkumai, pateikti pavyzdžiai. Atlikta Semantinio tinklo *SPARQL* užklausų kalbos analizė: palygintos *SPARQL* su *SQL*, aprašytos naujos *SRAPQL 1.1* galimybės, susipažinta su *SWRL* kalba. Atlikta *SPARQL* užklausų klasifikacija, atlikta Semantinio tinklo įrankių tyrimo etalonų analizė. Suformuluota siekiamo etalono kūrimo uždavinio formuluotė. Atlikta architektūros ir galimų įgyvendinimo priemonių variantų analizė.
- Trečiame skyriuje pateikta Semantinio tinklo užklausų tyrimo metodika, pasirinkta genealoginio medžio ontologija ir užklausų rinkinys tyrimui.
- Ketvirtame skyriuje pateikti ontologijos egzempliorių generavimo įrankio funkciniai, nefunkciniai reikalavimai, projektas, realizacijos modelis, testavimo modelis su duomenimis ir rezultatais.
- Penktame skyriuje suformuluotas eksperimento apibrėžimas, planavimas, gauti rezultatai ir rezultatų įvertinimas.
- Šeštame skyriuje suformuluotos darbo išvados.

Remiantis šio darbo rezultatais tema „Įrankis *SPARQL 1.1* užklausų vykdymo galimybėms tirti“ atspausdintas straipsnis ir perskaitytas pranešimas XVII tarpuniversitetinėje magistrantų ir

doktorantų konferencijoje „Informacinės technologijos 2012“. Straipsnis išspausdintas konferencijos leidinyje [12] ir pateiktas 1 priede.

2. Analizė

2.1. Tyrimo sritis, objektas ir problema

Magistrinio darbo tyrimo objektas – *SPARQL 1.1* užklausų sudarymo ir vykdymo procesas.

Šio magistrinio darbo tyrimo sritis yra Semantinio tinklo kalbos: *OWL*, *RDF*, *SPARQL*, *SPARQL 1.1*, *SWRL*.

RDF (*Resource Description Framework*) yra Semantinio tinklo *W3C* (*World Wide Web Consortium*) projekto dalis. Pasitelkiant *RDF* siekiama suteikti prasmę interneto puslapių turiniui. *RDF* yra išteklių aprašymo sistema, kuri remiasi subjektas – predikatas – objektas sąvokomis ir kuri gali būti formalizuota į žymėtąjį grafą, kuris vaizduoja informaciją internete [15].

OWL 2 (*Web Ontology Language*) – Semantinio tinklo ontologijos kalba su formaliai apibrėžta semantika. *OWL 2* ontologijos remiasi *RDF* grafais. Jos aprašo klases, savybes, individus ir duomenų reikšmes ir yra saugomos Semantiniam tinkle *RDF/XML* dokumentų pavidalu [13], [18].

SPARQL (*SPARQL Protocol and RDF Query Language*) – *RDF* užklausų kalba, kuri taikoma ir *OWL* ontologijoms [15], [18]. *SPARQL* užklausų kalba renka užklausos rezultatus iš tripletų, esančių ontologijoje, kurioje vykdoma *SPARQL* užklausa. *SPARQL 1.1* tai *SPARQL* užklausų kalbos nauja versija, kuri buvo papildyta naujomis užklausų galimybėmis.

SWRL – Semantinio tinklo taisyklių kalba. Taikant *SWRL* taisykles bei išvedimo mechanizmus galima išvesti papildomų savybių ontologijoje ir supaprastinti *SPARQL* užklausas.

SPARQL užklausų kalba, kaip ir pats Semantinis tinklas, nuolat tobulinama ir atnaujinama. Semantinio tinklo galimybės nemažai dėmesio susilaukia mokslo, medicinos, verslo srityse. Nuolat tobulėjant Semantinio tinklo kalboms, įrankiams, atsiranda problema, nes trūksta naujausios informacijos apie jų galimybes. Šiai problemai spręsti reikia ištirti *SPARQL* užklausų kalbos vykdymo charakteristikas sukuriant metodiką, kuri galėtų būti taikoma įvairiems Semantinio tinklo vykdymo įrankiams.

2.2. Semantinio tinklo žinių vaizdavimo kalbų analizė

RDF analizė

RDF grafas – ryšių grafas, kuris aprašomas subjektas – predikatas – objektas sąvokomis. *RDF*, kuris išreiškiamas *XML*, yra informacijos keitimosi standartas Semantiniame tinkle. *RDF* suteikia pastovų, standartizuotą būdą apibūdinti ir vykdyti paieškas interneto resursuose, nuo tekstinių puslapių ir grafinių elementų iki garso ir vaizdo failų. *RDF* yra pagrindas kuriant Semantinį tinklą.

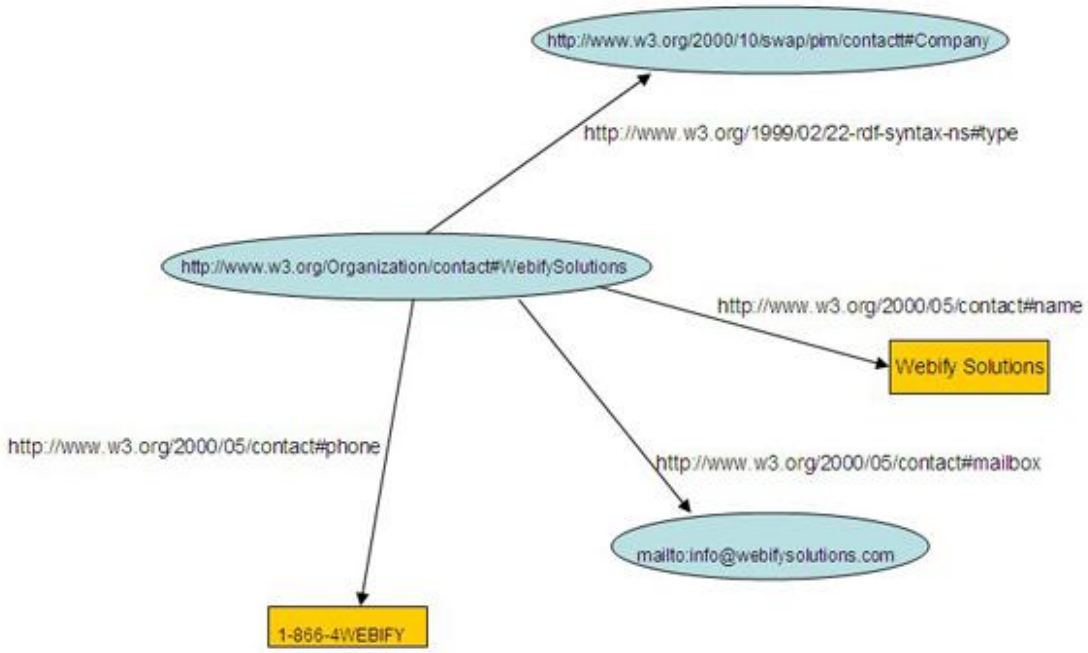
RDF pavyzdys, kuris pateikiamas XML formatu, pateiktas 2.1 paveiksle.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:contact="http://www.w3.org/2000/05/contact#">

  <contact:Company rdf:about="http://www.w3.org/Organization/contact#WebifySolutions">
    <contact:name>Webify Solutions</contact:name>
    <contact:mailbox rdf:resource="mailto:info@webifysolutions.com"/>
    <contact:phone> 1-800-4WEBIFY </contact:phone>
   </contact:Company>

</rdf:RDF>
```

2.1 pav. *RDF* pavyzdys [2]

Šis pavyzdys suteikia informaciją apie resursą, kuris šiuo atveju yra organizacija `http://www.w3.org/Organization/contact#WebifySolutions`. Organizacija gali būti identifikuota pagal *URI* `http://www.w3.org/Organization/contact#WebifySolutions`; jos pavadinimas yra `Weby Solutions`, elektroninis paštas – `info@webysolutions.com` ir telefono numeris – `1-800-4WEBIFY`.

RDF grafas, kuris iliustruoja 2.1 paveiksle pateiktą pavyzdį grafiniu būdu, pateiktas 2.2 paveiksle.



2.2 pav. *RDF* grafo pavyzdys [2]

RDF ir *RDFS* naudojimas atskleidė trūkumus, pagrindiniai iš jų:

- Vietinės apimties savybės. Pavyzdžiui, savybė „ėda“ gali būti apibrėžiama tik visoms klasėms, tokiu atveju naudojant *RDFS* negalima nusakyti, kad karvės ėda žolę, o liūtai - mėsą.
- Nesusikertančios klasės. Kartais reikia apibrėžti klases, kurios nesikerta (neturi bendrų reikšmių). *RDFS* galime nusakyti klasę „moteris“, tik kaip klasės „asmuo“ poklasę.

- Nesikertančių klasių kombinacija. Kartais reikia sukurti klasę iš jau egzistuojančių, kurios neturi bendrų reikšmių, naudojant *RDFS* to atlikti nėra galimybių. (Pavyzdžiui, reikia sukurti klasę „asmuo“ iš klasių „moteris“ ir „vyras“.)
- Kardinalumo apribojimai. *RDFS* nesuteikia galimybės apibrėžti apribojimus, kurie nusakytų, kiek reikšmių gali ar turi turėti klasės atributai. (Pavyzdžiui, žmogus gali turėti tik du tėvus.)
- Specialios klasės atributo charakteristikos. Kartais reikia nusakyti, kad vienas klasės atributas yra tranzityvus, unikalus ar yra kito atributo inversija. Tokių galimybių nesuteikia *RDFS*. [1]

***OWL* analizė**

Tam, kad taikomosios programos tikrai suprastų duomenis, yra reikalinga semantinė sintaksinė sąveika. Sintaksinė sąveika nusako tinkamą (teisingą) duomenų apdorojimą. Tam reikalingas susietumas tarp terminų, kuriam reikalinga turinio analizė. Turinio analizė reikalauja formalių ir aiškių apibrėžimo srities (*domain*) modelių specifikacijų, kurios nusako naudojamus terminus ir jų tarpusavio ryšius. Tokie formalūs apirėžimo srities modeliai dar vadinami ontologijomis. Ontologijos apibūdina duomenų modelius klasių, poklasių ir savybių požiūriu. Pasitelkiant *OWL* galima išreikšti ontologijas. *OWL* turi turtingesnį savybes ir klases apibūdinantį žodyną lyginant su *RDF* ar *RDF Schema*. Be to *OWL* gali apibrėžti: ryšius tarp klasių (tokių kaip nesusikertantys), kardinalumą, lygiateisiškumą, turtingesnį savybių aprašymą ir savybių charakteristiką.

OWL turi tris sub – kalbas, išraiškos mažėjimo tvarka:

- *OWL Full*
- *OWL DL*
- *OWL Lite*

OWL Full suteikia pilną *OWL* funkcionalumą, suteikiant visas galimas galimybes. Ši sub-kalba yra pilnai suderinama su *RDF* (sintaksiškai ir semantiškai).

OWL DL renkasi vartotojai, kurie nori gauti didžiausią išraiškingumą, kad kuo mažiau nukentėtų skaičiavimų išsamumas. *OWL DL* yra kalbos *OWL Full* sub-kalba.

OWL Lite renkasi vartotojai, kuriems reikia klasifikuotos hierarchijos ir nesudėtingai suvaržytų ypatybių (savybių). Ši kalba yra lengviausiai suprantama iš visų trijų, tačiau turi ribotą funkcionalumą, pavyzdžiui, palaiko kardinamumo reikšmes tik 0 arba 1.

Pagrindiniai *OWL* komponentai:

- Klasės
- Savybės

- Klasių egzemplioriai

Klasės yra pagrindiniai *OWL* ontologijos statybiniai blokai. Klasė yra konceptas apibrėžimo srityje. Paprastai jos sudaro hierarchijas (poklasės ir super-klasės hierarchija).

Joms apibūdinti naudojamas `owl:Class` elementas.

OWL klasės pavyzdys, kuris iliustruoja klasės apibūdinimą (apribojimas apibūdina `SavingAccount` kaip poklasę klasės `Account`), pateiktas 2.3 paveiksle.

```
<owl:Class rdf:ID="SavingsAccount">
 <rdfs:subclassOf rdf:resource="#Account"/>
</owl:Class>
```

2.3 pav. *OWL* klasės pavyzdys [2]

OWL palaiko šešis pagrindinius būdus klasėms aprašyti. Paprasčiausias yra klasė. Kiti: susikertančios, susijusios (sujungtos), papildančios, apribojamos, išvardintos klasės.

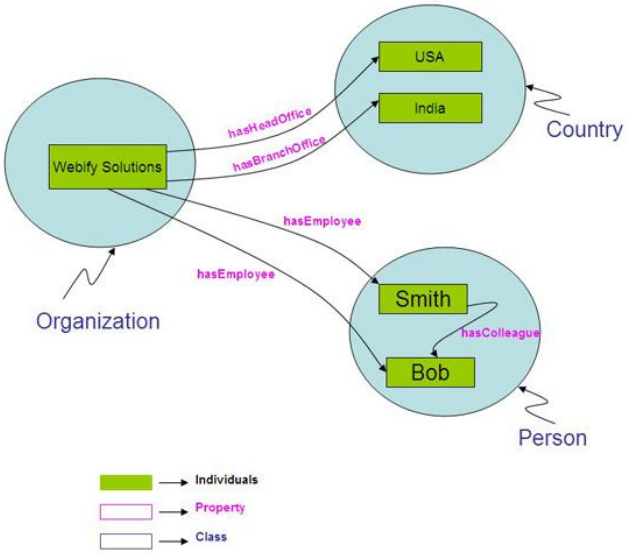
Savybės turi dvi pagrindines kategorijas:

- Objektų savybės (*Object properties*)
- Duomenų tipo savybės (*Datatype properties*)

Savybė gali turėti su ja susijusią apibrėžimo sritį. Kiekviena savybė gali priklausyti vienai iš šių kategorijų:

- Funkcinė. Savybė duotam objektui (pavyzdžiui, pastato aukštis, spalva, forma).
- Atvirkštinė funkcinė. Du skirtingi individai negali turėti tos pačios reikšmės (pavyzdžiui, asmens kodas yra skirtingas kiekvienam asmeniui).
- Simetrinė. Jei savybė jungia A su B, tuomet galima daryti išvadą, kad ji jungia ir B su A.
- Tranzityvi. Jei savybė jungia A su B ir B su C, tuomet galima daryti išvadą, kad ji jungia ir A su C.

Klasių egzemplioriai yra klasių atskiri atvejai, savybės gali jungti klasių egzempliorius tarpusavyje. Pavyzdžiui, galima apibūdinti egzempliorių, kurio vardas yra `Smith`, kaip atskirą atvejį klasės `Person`, galima panaudoti savybę `hasEmplyer` tam, kad būtų galima susieti `Smith` su egzemplioriumi `Webify Solutions`, taip išreiškiant, kad asmuo `Smith` yra organizacijos `Webify Solutions` darbuotojas. Pavyzdys, kuris iliustruoja `Webify Solutions` organizaciją, pateiktas 2.4 paveiksle [2].



2.4 pav. *OWL* komponento individo pavyzdys [2]

2.3. Semantinio tinklo užklausų kalbos *SPARQL* analizė

SPARQL yra Semantinio tinklo užklausų kalba, kuri naudojama gauti informaciją, pasitelkiant panašią sintaksę į *SQL*. Informacija gaunama išrenkant tripletus, kurie apibūdinami, kaip subjektas – predikatas – objektas sąvokomis. Nepaisant sintaksinio panašumo į *SQL*, šios užklausų kalbos turi nemažai skirtumų.

2.3.1. *SPARQL* palyginimas su *SQL*

Darbas su *RDF* užklausų rašymu progresavo, buvo taikomi įvairūs metodai tam, kad būtų sukurta tokia užklausų kalba, kuri būtų panaši į *SQL*, dėl jos paplitimo ir nesudėtingos, lengvai suprantamos sintaksės. Šie bandymai, kurti naudojant *SQL* sintaksę, buvo populiariausi, todėl *SPARQL* seka tuo pačiu keliu ir naudoja plačiai žinomą (*SQL* vartotojams) `SELECT` užklausų formą.

SQL sintaksės pavyzdys pateiktas 2.5 paveiksle.

```
SELECT salary
 FROM employees
WHERE emp_id = 'e13954'
```

2.5 pav. Užklausa *SQL* sintakse [11]

SPARQL užklausos pavyzdys pateiktas 2.6 paveiksle.

```
SELECT ?sal
WHERE {emps:e13954 HR:salary ?sal.}
```

2.6 pav. Užklausa *SPARQL* sintakse [11]

SPARQL negali modifikuoti *RDF* duomenų rinkinio, ji yra tik skaitomojo pobūdžio (rezultatams gauti) [7], [6]. Pasirodžius naujai *SPARQL 1.1* versijai, atsirado galimybė modifikuoti tiek pačius grafus, tiek jų tripletus.

Yra galimybė *SPARQL* užklausas transformuoti į *SQL* užklausas [6].

Lyginant *SQL* ir *SPARQL* reiktų nepamiršti, kad *SQL* užklausų kalba naudojama labai struktūrizuotai informacijai, tokiai, kuri yra naudojama daugelyje verslo procesų. Gerai žinomi pavyzdžiai galėtų būti: klasės ir mokiniai, darbuotojai ir skyriai. Tokia informacija dažnai turi reikšmę kiekviename lentelės stulpelyje. *SQL* kalbos sintaksė telkia dėmesį į duomenų išrinkimą, kur komponentai yra prieinami ir sujungti duomenys pagrįsti tų komponentų reikšmėmis. Pavyzdžiui, sujungti duomenis iš kelių lentelių galima pasinaudojant *SQL* operatoriais *UNION* arba *JOIN*.

RDF grafas – ryšių grafas, kuris aprašomas subjektas – predikatas – objektas sąvokomis. *RDF* vaizdavimas lentele pateiktas 2.1 lentelėje.

2.1 lentelė RDFTable lentelė

Subjektas	Predikatas	Objektas
emps:id105	rdb:employees/column#name	Jon Jonnson
emps:id105	rdb:employees/column#salary	110000.00
emps:id105	rdb:employees/column#department	Depts:id11

Šis pavyzdys, pateiktas 2.1 lentelėje, pateikia informaciją, kad rašant *SQL* užklausą yra gaunamas trivialus atsakymas.

```
SELECT Objektas
FROM RDFTable
WHERE Subjektas=”emps:id105”
```

Pateiktas *RDFTable* lentelės pavyzdys pateikia informaciją, kad lentelės stulpeliuose galima saugoti skirtingo formato duomenis (*SQL* atveju grąžinamų reikšmių duomenų formatai viename stulpelyje yra vienodi).

RDF gali nurodyti priklausomybę klasėms (7.1 lentelė), *RDF* yra grafas, kuris atvaizduoja duomenų modelį. *OWL* yra *RDF*, bet su praplėstomis galimybėmis.

Skirtumas tarp *SQL* ir *SPARQL*: *SPARQL* sintaksė sujungimo operacijas padaro numanomas, tuo tarpu *SQL* sintaksė paprastai padaro juos atvirus (tikslius). Norint parašyti užklausą *SQL* kalba, kuri būtų skirta *RDF* grafui, užklausa būtų ne tik ilga, bet ir sudėtingą, būtent dėl to, kad reiktų apibrėžti aiškų sujungimą. Tačiau rašant užklausas *SPARQL* užklausų kalba, kur reikėtų apibrėžti keletą sujungimo operacijų, užklausos yra trumpesnės. Kuo užklausa trumpesnė, tuo mažesnė galimybė, kad rašant bus padaryta klaidų [11].

2.3.2. *SPARQL 1.1*

2011 metų gegužės mėnesį *W3C* grupė išleido naujesnę *SPARQL* užklausų versiją *SPARQL 1.1*. Joje buvo įtrauktos tokios naujos galimybės [19]:

- Iteracijų galimybė

  ```
  SELECT (?price * ?qty AS ?total_price) ...
  ```

- Agregavimo funkcijos (`MIN`, `MAX`, `COUNT`, `AVG`, `SUM`)

  ```
  SELECT (MIN(?price) AS ?min_price) ...
  ```

- Vidinių užklausų galimybė

```
SELECT ?article ?author
WHERE{
      ?article ex:author ?author .
      {
      SELECT ?article WHERE{
      ...?article...
      } ORDER BY ...
      }
```

- Neigimo ir filtravimo galimybės

  `FILTER ... NOT EXISTS ..., HAVING, MINUS`

- *SPARQL 1.1* taip pat turi numatytas galimybes atlikti tokias operacijas kaip `INSERT`, `UPDATE`, `DELETE`. Turi galimybę kurti bei modifikuoti tripletus bei jų reikšmes.

SPARQL užklausų kalba, kuri yra naudojama su įvairiomis ontologijomis, yra naudojama šios užklausų kalbos kūrėjų ir asmenų, kurie siekia šią kalbą tobulinti tam, kad ateityje Semantinis tinklas įgautų tikrąja savo prasmę – internetas taptų viena globalia duomenų baze, kurioje būtų galima surasti reikiamą informaciją pasinaudojant prasminėmis reikšmėmis pagal pateiktus paieškos kriterijus.

SPARQL užklausų kalbos galimybės domina Semantinio tinklo programų kūrėjus. Kadangi *SPARQL* užklausų kalba yra tobulinama, atsiranda naujų jos galimybių, reikia jas ištirti tam, kad Semantinio tinklo programų kūrėjai žinotų jos galimybes, galėtų šias žinias pritaikyti, kad paieška būtų greitesnė, o rezultatai patikimesni.

2.3.3. *SWRL* taisyklės

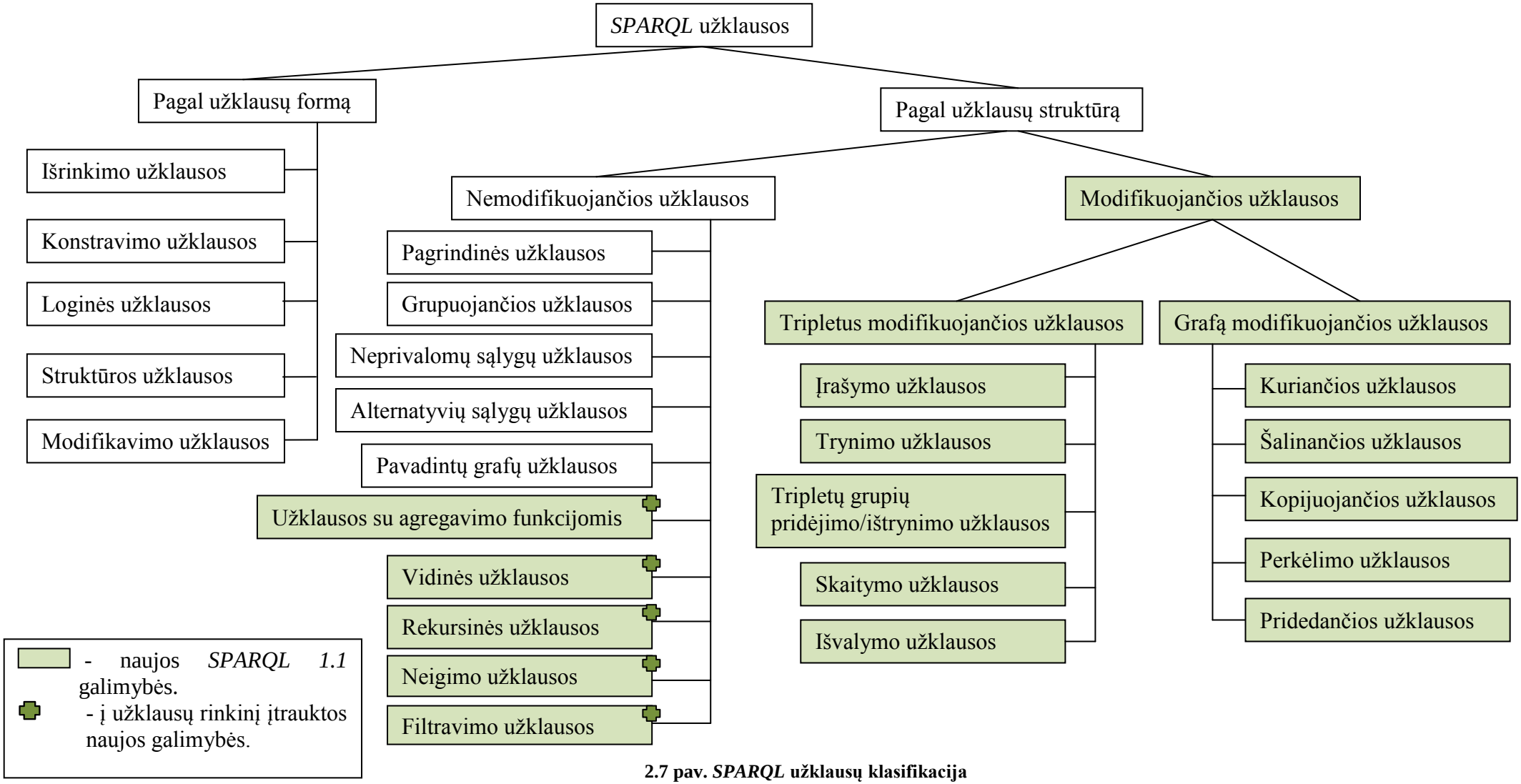
SWRL – Semantinio tinklo taisyklių kalba. Pasinaudojant šia taisyklių kalba, nesunkiai galima išvesti papildomų savybių, kurias galima naudoti rašant *SRARQL* užklausas [21]. Dėl *SWRL* taisyklių *SPARQL* užklausa tampa trumpesnė, todėl klaidų tikimybė užrašant užklausą sumažėja, o užklausa tampa paprastesnė, nes nereikia išvedinėti tarpinių rezultatų pačioje *SPARQL* užklausoje:

`hasParent(?x,?y) ∧ hasBrother(?y,?z) ⇒ hasUncle(?x,?z)`

Pateiktame pavyzdyje, pasitelkus savybėmis `hasParent` ir `hasBrother` buvo išvesta nauja savybė `hasUncle`. Naujos savybės yra išvedamos pasinaudojus išvedimo mechanizmais, tačiau ne visi išvedimo mechanizmai palaiko *SWRL* taisykles. Išvedimo mechanizmų analizė, kurie buvo nagrinėjami šiame darbe, pateikta 2.5 skyriaus 2.10 lentelėje.

2.3.4. *SPARQL* užklausų klasifikacija

Atsižvelgiant į naujos versijos *SPARQL 1.1* galimybes, buvo sudaryta *SPARQL* užklausų klasifikacija, kuri pateikta 2.7 paveiksle.



2.7 pav. *SPARQL* užklausų klasifikacija

Pagal pateiktą klasifikaciją *SPARQL* užklausos skirstomos [19]:

1. pagal užklausų formą;
2. pagal užklausų struktūrą.

Pagal užklausų formą užklausos skirstomos:

1.1. **Išrinkimo užklausos.** Užklausos, kurios užrašomos naudojant *SELECT*. Grąžina kintamuosius su užpildytomis reikšmėmis tiesiogiai. Išrinkimo užklausos pavyzdys pateiktas 2.8 paveiksle.

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?nameX ?nameY ?nickY
WHERE
  { ?x foaf:knows ?y ;
       foaf:name ?nameX .
    ?y foaf:name ?nameY .
    OPTIONAL { ?y foaf:nick ?nickY }
  }
```

2.8 pav. Išrinkimo užklausa [19]

1.2. **Konstravimo užklausos.** Užklausos, kurios užrašomos naudojant *CONSTRUCT*. Konstravimo užklausos pavyzdys pateikta 2.9 paveiksle.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX site: <http://example.org/stats#>

CONSTRUCT { [] foaf:name ?name }
WHERE
{ [] foaf:name ?name ;
     site:hits ?hits .
}
ORDER BY desc(?hits)
LIMIT 2
```

2.9 pav. Konstravimo užklausa [19]

1.3. **Loginės užklausos.** Užklausos, kurios užrašomos naudojant *ASK*. Grąžinama atsakymas *yes* arba *no*, priklausomai nuo to, ar tokia užklausa turi atsakymą (kokią nors reikšmę) ar ne. Patogu naudoti jeigu norima sužinoti, ar užklausa gali turėti reikšmių (jos aibė netuščia). Loginės užklausos pavyzdys pateiktas 2.10 paveiksle.

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
ASK  { ?x foaf:name  "Alice" }
```

2.10 pav. Loginė užklausa [19]

1.4. **Struktūros užklausos.** Užklausos, kurios užrašomos naudojant *DESCRIBE*. Grąžina informacijos grafą (struktūrą). Struktūros užklausos pavyzdys pateiktas 2.11 paveiksle.

```
PREFIX ent:  <http://org.example.com/employees#>
DESCRIBE ?x WHERE { ?x ent:employeeId "1234" }
```

2.11 pav. Struktūros užklausa [19]

1.5. **Modifikavimo užklausos.** Užklausos, kuriomis galima modifikuoti grafą ar jame esančius tripletus.

Pagal užklausų struktūrą užklausos skirstomos:

2.1. **Nemodifikuojančios užklausos.** Užklausos, kurios tik pateikia rezultatą nekeisdamos grafo.

2.2. **Modifikuojančios užklausos.** Užklausos, kurios gali modifikuoti grafą ar jo viduje esančius tripletus.

Nemodifikuojančios užklausos skirstomos:

2.1.1. **Pagrindinės užklausos.** Užklausos, kurios susideda iš *SELECT* ir *WHERE* sakinių. Pagrindinės užklausos pavyzdys pateiktas 2.12 paveiksle.

```
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
PREFIX  : <http://example.org/book/>

SELECT  $title
WHERE   { :book1  dc:title  $title }
```

2.12 pav. Pagrindinė užklausa [19]

2.1.2. **Grupuojančios užklausos.** Užklausos, kurios susideda iš *SELECT* ir *WHERE* sakinių atskirtų riestiniais skliaustais. *WHERE* sakinyje sąlygos yra grupuojamos rašant jas atskiruose skliaustuose. Grupuojančios užklausos pavyzdys pateiktas 2.13 paveiksle.

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  { { ?x foaf:name ?name . }
         { ?x foaf:mbox ?mbox . }
       }
```

2.13 pav. Grupuojanti užklausa [19]

2.1.3. **Neprivalomų sąlygų užklausos.** Užklausos, kurios aprašytoje *WHERE* sakinyje turi neprivalomų sąlygų, jei sąlyga yra neprivaloma ji rašome *OPTIONAL* sakinyje. Neprivalomų sąlygų užklausos pavyzdys pateiktas 2.14.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  { ?x foaf:name  ?name .
         OPTIONAL { ?x  foaf:mbox  ?mbox }
       }
```

2.14 pav. Neprivalomų sąlygų užklausa [19]

2.1.4. **Alternatyvių sąlygų užklausos.** Užklausos, kurios aprašytoje *WHERE* sakinyje turi alternatyvių sąlygų, jei sąlyga yra neprivaloma ji rašome *UNION* sakinyje. Alternatyvių sąlygų užklausos pavyzdys pateiktas 2.15 paveiksle.

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>

SELECT ?x ?y
WHERE  { { ?book dc10:title ?x } UNION { ?book dc11:title  ?y } }
```

2.15 pav. Alternatyvių sąlygų užklausa [19]

2.1.5. **Pavadintų grafų užklausos.** Užklausos, kuriose naudojamas *GRAPH* sakinys, kuris pakeičia vieną grafą, kad jis atitiktų kitą grafą per užklausos šablono dalį. Pavadintų grafų užklausos pavyzdys pateiktas 2.16 paveiksle.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?src ?bobNick
FROM NAMED <http://example.org/foaf/aliceFoaf>
FROM NAMED <http://example.org/foaf/bobFoaf>
WHERE
  {
    GRAPH ?src
    { ?x foaf:mbox <mailto:bob@work.example> .
      ?x foaf:nick ?bobNick
    }
  }
```

2.16 pav. Pavadintų grafų užklausa [19]

2.1.6. **Užklausos su agregavimo funkcijomis.** Užklausos, kurios naudoja agregavimo funkcijas. Šiose užklausose rezultatams sugrupuoti naudojami *GROUP BY* ir *HAVING*. Galimos agregavimo funkcijos pateiktos 2.2 lentelėje.

2.2 lentelė Agregavimo funkcijos

Funkcija	Naudojimas
COUNT	Kiekio skaičiavimui
SUM	Sumos skaičiavimui
MIN	Minimalios reikšmės radimui
MAX	Maksimalios reikšmės radimui
AVG	Vidurkio radimui
GROUP_CONCAT	Rezultato grupavimui
SAMPLE	Paprastose ir vidinėse užklausose su agregavimo funkcijomis reikšmės, kurios gaunamos iš užklausos šablono, bet nėra *GROUP BY* sakinyje, negali būti naudojami išraiškose. Siekiant padidinti išraiškingumą naudojamas *SAMPLE* sakinys

Užklausos su agregavimo funkcija *SUM* pavyzdys pateiktas 2.17 paveiksle.

```
PREFIX : <http://books.example/>
SELECT (SUM(?lprice) AS ?totalPrice)
WHERE {
  ?org :affiliates ?auth .
  ?auth :writesBook ?book .
  ?book :price ?lprice .
}
GROUP BY ?org
HAVING (SUM(?lprice) > 10)
```

2.17 pav. Užklausa su agregavimo funkcija *SUM* [19]

2.1.7. **Vidinės užklausos.** Užklausos, kuriose panaudojamas vidinis *SELECT* sakinys ar sakiniai. Tokio tipo užklausos suteikia galimybę išrinkti reikiamus rezultatus sumažinant tarpinių rezultatų kiekį užklausoje.

2.1.8. **Rekursinės užklausos.** Tai tokios užklausos, kuriose galima realizuoti rekursijas. Galima naudoti vienos krypties (naudojant * simbolį), apibrėžto dydžio (galima nurodyti reikalingo gylio rekursiją riestiniuose skliaustuose {n, m}) ir dviejų krypčių (atskiriant apskliaustus predikatus simboliu | ir už skliaustų naudojant * simbolį) rekursijas.

2.1.9. **Neigimo užklausos.** Užklausos, kurios išveda rezultatus atmetant (*MINUS*) arba išfiltruojant (*(NOT) EXISTS*) netinkamus rezultatus taip juos paneigiant.

```
PREFIX :       <http://example/>
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>

SELECT DISTINCT ?s
WHERE {
   ?s ?p ?o .
   MINUS {
      ?s foaf:givenName "Bob" .
   }
}
```

2.1.10. **Filtravimo užklausos.** Užklausos, kuriose naudojant *FILTER* ir sąlygą yra išfiltruojami reikalingi rezultatai, yra galimybė naudoti vidinius filtrus.

```
{
  ?x foaf:knows/^foaf:knows ?y .
  FILTER(?x != ?y)
}
```

Modifikuojančios užklausos skirstomos [20]:

2.2.1. tripletus modifikuojančios užklausos;

2.2.2. grafą modifikuojančios užklausos.

Tripletus modifikuojančios užklausos skirstomos:

2.2.1.1. **Įrašymo užklausos.** Užklausos, kurios įrašo duotus tripletus į grafą, naudojant *INSERT DATA*.

2.2.1.2. **Trynimo užklausos.** Užklausos, kurios ištrina nurodytus tripletus iš grafo, jeigu jie jame egzistuoja, naudojant *DELETE DATA*.

2.2.1.3. **Tripletų grupių pridėjimo/ištrynimo užklausos.** Užklausos, kurios prideda/ištrina grupę tripletų iš grafo, naudojant *INSERT/DELETE*.

2.2.1.4. **Skaitymo užklausos.** Užklausos, kurios perskaito grafo turinį, naudojant *LOAD*.

2.2.1.5. **Išvalymo užklausos.** Užklausos, kurios pašalina visus grafo ar grafų tripletus, naudojant *CLEAR*.

Grafą modifikuojančios užklausos skirstomos:

2.2.2.1. **Kuriančios užklausos.** Užklausos, kurios gali sukurti grafą, naudojant *CREATE*.

2.2.2.2. **Šalinančios užklausos.** Užklausos, kurios gali pašalinti grafą su visu jo turiniu, naudojant *DROP*.

2.2.2.3. **Kopijuojančios užklausos.** Užklausos, kurios gali modifikuoti grafą, kad jis turėtų kito grafo kopiją, naudojant *COPY*.

2.2.2.4. **Perkėlimo užklausos.** Užklausos, kurios gali perkelti vieno grafo turinį į kitą grafą, naudojant *MOVE*.

2.2.2.5. **Pridedančios užklausos.** Užklausos, kurios gali pridėti visą vieno grafo informaciją į kitą grafą, naudojant *ADD*.

Ši užklausų klasifikacija parodo, kokios yra galimos *SPARQL* užklausos, kokios jų galimybės ieškant ar modifikuojant grafo tripletus. Reikėtų nepamiršti, kad *SPARQL* turi rezultatų pateikimo modifikatorius, tokius kaip *ORDER BY*, *DISTINCT*, *REDUCED*, *OFFSET* ir *LIMIT*. Šių modifikatorių aprašymas pateikiamas 2.3 lentelėje.

2.3 lentelė Rezultatų modifikatoriai

Modifikatorius	Naudojimas
ORDER BY	Rezultato rikiavimui
DISTINCT	Dubliavimo panaikinimui rezultatuose, užtikrina, kad rezultatai unikalūs
REDUCED	Dubliavimo panaikinimui rezultatuose, leidžia eliminuoti kai kurios pasikartojančius rezultatus
OFFEST	Nustatant nuo kurio gaunamo rezultato rinkinio pateikti rezultatus
LIMIT	Nustato grąžinamų rezultatų skaičių. Nustatant į 0 rezultatų aibė bus tuščia

2.4. Semantinio tinklo įrankių tyrimo etalonų analizė

Tiriant įrankių savybes *SPARQL* etalonu, tyrimai atliekami su tos pačios ontologijos pavyzdžiais, turinčiais skirtingą sugeneruotų egzempliorių skaičių. Tiriamas – ontologijų užkrovimo laikas, galimybės išvesti rezultatus priklausomai nuo *OWL* sub-kalbų (*OWL-Lite, OWL-DL*), užklausų vykdymo laikas, užbaigtumas (grąžinamas sistemos atsakymas į užklausą) bei tikslumas (užbaigtumo tikslumas), be to, Semantinio tinklo įrankiai ir jų charakteristikos, lyginamos su kitomis technologijomis (pvz.: reliacinėmis duomenų bazių valdymo sistemomis (DBVS)). Kadangi kiekvienas etalonas turi tik vieną ontologiją ir užklausų rinkinį, vieno etalono nepakanka, kad būtų galima ištirti visas *SPARQL* užklausų kalbos ir įrankių charakteristikas.

Geriausiai žinomi ir taikomi yra seniausias *LUBM* (*Lehigh University Benchmark*) etalonas, kuris buvo patobulintas į *OUBM* (*University Ontology Benchmark*) ir *Berlin SPARQL* etalonas.

LUBM etalone naudojama universiteto ontologija ir 14 užklausų [8]. Jis leidžia palyginti ontologijų užkrovimo ir užklausų vykdymo greitį, bei įvertinti gaunamų atsakymų teisingumą priklausomai nuo ontologijos egzempliorių skaičiaus. Tačiau *LUBM* negali pilnai ištestuoti visų ar bent jau daugelio Semantinio tinklo įrankių galimybių [10]: neapima visų *OWL – Lite* ir *OWL – DL* konstrukcijų, o jo sugeneruojami egzemplioriai, kurie sudaro medžius, tarpusavyje nesusieti. Šie trūkumai buvo pataisyti patobulintame *LUBM* variante *OUBM*.

Berlin SPARQL etalonas naudoja e-komercijos ontologiją aprašančią produktus, kuriuos siūlo įvairūs pardavėjai ir pirkėjus, kurie gali rašyti atsiliepimus apie siūlomus produktus. *Berlin SPARQL* etalonas turi 12 užklausų ir matuoja šias metrikas: užklausų įvairovę per valandą, užklausas per sekundę, duomenų užkrovimą. Šiuo etalonu buvo tiriamos, pavyzdžiui, *RDF* saugyklų (*Sesame, Virtuaso, Jena TDB* ir *Jena SDB*), *SPARQL-to-SQL* transliatorių (*D2R Server* ir *Virtuoso RDF Views*), reliacinių DBVS (*MySQL* ir *Virtuaso RDBMS*) galimybės [4].

Iš esamų etalonų dar galima paminėti SP^2Bench, kurio autoriai giriasi, kad jų etalonas sugeneruoja itin tikslius duomenis, atitinkančius realaus pasaulio reikalavimus. Jų pasirinkta ontologija – publikacijos. Šis etalonas skiriasi nuo jau paminėtų tuo, kad norima išanalizuoti užklausų veikimą bendruoju atveju, netaikant konkrečiam įrankiui. SP^2Bench etalone yra 17 užklausų [17], tačiau jos neapima rekursinių bei kitų naujų *SPARQL 1.1* užklausų tipų.

Minėti etalonai didžiausią dėmesį skiria į Semantinio tinklo užklausų įrankių galimybes, tačiau nevertina loginio išvedimo [22] ir pačios *SPARQL* užklausų kalbos. Paprastai jie gali parinkti geriausius įrankius atskiriems taikymo scenarijams, bet ne įvertinti *SPARQL* galimybes bendruoju atveju.

Nei vienas etalonas neatsižvelgia į užklausų optimizavimą taikant *SWRL* taisykles, kurios leidžia išvesti papildomų savybių ir supaprastinti *SPARQL* užklausas.

Etalonų savybių aprašymas įtraukiant ir siekiamą įrankį pateikta 2.4 lentelėje.

2.4 lentelė Etalonų savybių aprašymas

Įrankis / savybė	***LUBM***	***Berlin Benchmark***	***SP²Bench***	**Siekiamas įrankis**
Ontologijų kalba	*OWL*	*RDF*	*OWL*	*OWL 2*
Dalykinė sritis	Universitetų veikla	E-komercija	Publikacijos	Genealoginis medis
Tiriami parametrai	Egzempliorių skaičius	Egzempliorių skaičius	Egzempliorių skaičius ir užklausų tipai	Hierarchijos gylis ir plotis, ryšių skaičius, *SWRL* taikymas
Vertinimo objektas	Užklausų vykdymo galimybės ir saugojimo mechanizmai	Užklausų vykdymo galimybės ir saugojimo mechanizmai	Užklausų vykdymo galimybės	Užklausų vykdymo galimybės
Užklausų tipai	Savybėmis (*OUBM* ir ryšiais) grindžiamos užklausos	Savybėmis grindžiamos užklausos	Visos *SPARQL* užklausos	Rekursinės ir naujų tipų *SPARQL 1.1* užklausos

2.5. Siekiamo etalono kūrimo uždavinio formuluotė

2.5.1. Siekiamos sistemos apibrėžimas

Siekiami sprendimai: sukurti *SPARQL* užklausų rinkinį remiantis atlikta *SPARQL* užklausų klasifikaciją, aprašyti metodiką, kuri būtų naudojama Semantinio tinklo *SPARQL* užklausų kalbos bei įrankių tyrimui, sukurti pasirinktos ontologijos egzempliorių generavimo įrankį, kuris leistų tirti naujas *SPARQL 1.1* užklausų kalbos galimybes didžiausią dėmesį skiriant rekursinėms užklausoms, išanalizuoti *SPARQL* užklausų kalbos vykdymo galimybes su pasirinktu vykdymo įrankiu: ištirti šių užklausų vykdymo laiką, pateikti vykdymo rezultatus priklausomai nuo naudojamos kompiuterinės įrangos.

Šio darbo tikslas – padidinti ontologijų užklausų vykdymo priemonių įvertinimo galimybes, sudarant tam skirtą metodiką ir realizuojant įrankį, leidžiantį generuoti ontologijas *SPARQL 1.1* versijos galimybėms, kurių neapima esami *SPARQL* vykdymo priemonių tyrimo etalonai, tirti. Šio darbo tyrimo rezultatas – ontologijos egzempliorių generavimo įrankis, užklausų rinkinys, vykdymo metodika bei vykdymo rezultatai gauti vykdant užklausas su skirtingus parametrus turinčia kompiuterine įranga.

Uždaviniai:

1. Atlikti Semantinio tinklo kalbų ir ontologijų užklausų vykdymo priemonių vertinimo etalonų analizę.
2. Sudaryti tyrimo metodiką ir *SPARQL 1.1* užklausų rinkinį.

3. Suprojektuoti ir realizuoti įrankį, leidžiantį generuoti įvairių dydžių ontologijas.
4. Atlikti eksperimentą, pritaikant sudarytą metodiką ir įrankį ontologijos užklausų vykdymo priemonėms įvertinti.
5. Įvertinti gautus rezultatus.

Kadangi egzistuojantys etalonai turi trūkumų, buvo nuspręsta sukurti naują etaloną, kuriam pasirinkta genealoginio medžio ontologija, kurioje galima vykdyti rekursines bei kitas naujas *SPARQL* užklausas. Genealoginio medžio *OWL 2* ontologijos fragmentas ir sugeneruoti egzemplioriai pateikti 3.2 skyriaus 3.2 paveiksle.

Nei vienas etalonas neatsižvelgia į užklausų optimizavimą taikant *SWRL* taisykles, kurios leidžia išvesti papildomų savybių ir supaprastinti *SPARQL* užklausas. Be to, užklausų rinkinys apims naujas *SPARQL 1.1* galimybes, kurios nebuvo įtrauktos į kitus egzistuojančius etalonus.

2.5.2. Architektūros ir galimų įgyvendinimo priemonių variantų analizė

Šiame darbe yra reikalinga pasirinktos ontologijos egzempliorius generuojanti sistema, ontologijos ir jos egzempliorių peržiūros programa ir *SPARQL* užklausas vykdanti programa, kuri reikalinga eksperimentiniam tyrimui atlikti. Įrankis, kuris turėtų visų šių įrankių funkcionalumą (ar bent dviejų iš jų) ir tiktų tiriamajam darbui, nebuvo rastas.

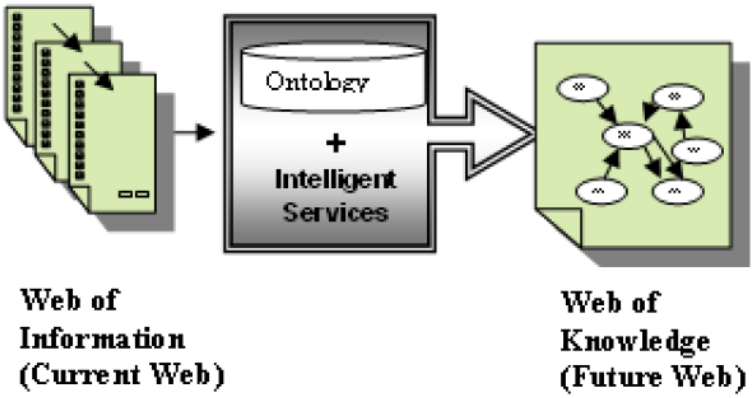
- **Ontologijos egzempliorių generavimo įrankis**

Pasirinkta ontologija yra genealoginis medis. Šiai ontologijai, kad būtų galima atlikti tyrimą, reikia sukurti egzempliorių generavimo sistemą. Generavimo sistemai realizuoti nebuvo specialių reikalavimų, todėl buvo pasirinkta atviro kodo tarpusavyje suderinama programinė įranga.

Buvo pasirinkta atviro kodo *PHP* programavimo kalba, kuri pasižymi tuo, kad turi didelį funkcionalumą, jos sintaksė yra nesudėtinga. Atviro kodo sistemos duomenų bazių valdymo sistema *MySQL* buvo pasirinkta dėl jos gero suderinamumo su *PHP*. Duomenų bazei valdyti – kurti, redaguoti duomenų bazės lenteles bei informaciją jose, naudojamas įrankis *PhpMyAdmin*. Sistemos kodui rašyti pasirinktas įrankis *Notepad++*. Sistemos dizainas sukurtas naudojant *CSS* (*Cascading Style Sheets*) kalba. Visų šių įrankių pagrindinė pasirinkimo priežastis ta, kad tarpusavio integracija nesudėtinga, diegimas yra nemokamas bei paprastas (neužimantis daug laiko ir nereikalaujantis daug nustatymų.

- **Ontologijos redagavimo (peržiūros) įrankis**

Norint atlikti ir išbandyti *SPARQL* užklausų kalbą, reikia pasirinkti vieną iš keleto siūlomų ontologijos kūrimo (redagavimo, peržiūros) įrankių ontologijos peržiūrai. Egzistuoja keletas įrankių, kurie leidžia naudoti jau išvystytas ontologijas (tokias kaip *OWL*). Visi ontologijos redaktoriai iliustruoja naujos kartos interneto realizaciją, kuri vaizdžiai yra pateikta 2.18 paveiksle



2.18 pav. Esamo ir būsimo interneto realizacija [9]

Internete galima rasti įvairių ontologijos kūrimo įrankių tokių kaip *Protégé*, *SWOOP*, *OntoTrack, WebOnto, TopBrain,* Internet *Business Logic, IsaViz* bei daugelį kitų. Pasirenkant tinkamiausią reikia išanalizuoti įrankių savybes ir nuspręsti, kuris įrankis darbui yra tinkamiausias. Kadangi ontologija nėra didelė – įrankių galimybės neturi būti išsamiai analizuojamos. Svarbiausia – pasirinkti tokį įrankį, kuris būtų palaikomas (nuolat atnaujinamas, nes Semantinis tinklas, ontologijos yra naujos sąvokos ir įgyvendinami tikslai, kurie nuolat tobulinami). Taip pat svarbu tai, kad būtų galima rasti pagalbos internete ir kiti aspektai pateikti 2.5 lentelėje. Funkcinės galimybės aptartos 2.6 – 2.9 lentelėse.

Šiame skyriuje atliekama lyginamoji analizė tokių įrankių: *Protégé*, *SWOOP*, *IsaViz*.

2.5 lentelė Ontologijos redaktorių palyginimas

Įrankis / **Parametrai**	***Protégé***	***SWOOP***	***IsaViz***
Kūrėjas	Standford University School of Medicine	University of Maryland	Emmanuel Pietriga
Paskutinis atnaujinimas	2011.10.04	2009.09.29	2007.05
Palaikymas	+	+	+/-
Populiariausias [16]	+	-	-
Parsisiuntimo galimybė	+	+	+
Interfeiso patogumas	+	+	+/-

Lentelėse 2.6 – 2.9 yra pateikiama lyginamoji ankščiau aprašytų įrankių analizė. Kiekvienoje lentelėje yra pateikiamas skirtingų aspektų palyginimas.

Lentelėje 2.6 yra pateikiamas įrankių palyginimas pagal įrankių architektūrą (pagal praplečiamumas ir ontologijų saugojimo formatus). Informacija apie praplečiamumą ir ontologijų saugojimą. Praplečiamumą turi tik *Protégé* įrankis, o ontologijos saugomos visuose įrankiuose failuose, tik *Protégé* turi papildomą formatą – *DBMS* (naudoja duomenų bazes saugoti ontologijoms) [5].

2.6 lentelė Įrankių palyginimas pagal architektūrą

Savybė	*Protégé*	*SWOOP*	*IsaViz*
Praplečiamumas	Taip	Ne	Ne
Ontologijų saugojimas	Failai ir *DBMS*	Failai	Failai

Lentelėje 2.7 yra pateikiamas įrankių palyginimas pagal sąveikas (pagal importavimo, eksportavimo formatus ir sujungimą).

2.7 lentelė Įrankių palyginimas pagal sąveiką

Savybė	*Protégé*	*SWOOP*	*IsaViz*
Importavimo formatas	*XML, RDF(S), XML Shema* ir *OWL*	*RDF(S), OIL, DAML*	*XSLT, RDF(S), OIL, DAML+OIL, OWL*
Eksportavimo formatas	*XML, RDF(S), XML Shema, Java, html*	*RDF(S), OIL, DAML*	*XSLT, RDF(S), OIL, DAML+OIL, OWL*
Sujungimas	Taip	Ne	Ne

Lentelėje 2.8 yra pateikiamas įrankių palyginimas pagal išvadų darymo paslaugas (pagal išimčių tvarkymą ir nuoseklumo tikrinimą).

2.8 lentelė Įrankių palyginimas pagal išvadų darymo paslaugas

Savybė	*Protégé*	*SWOOP*	*IsaViz*
Išimčių tvarkymas	Ne	Taip	Ne
Nuoseklumo tikrinimas	Taip	Tik rašybos klaidas	Tik paveldimumo ir ciklų aptikimo hierarchijose

Lentelėje 2.9 yra pateikiamas įrankių palyginimas pagal įrankių panaudojimą (pagal bendradarbiavimą su kitais įrankiais, ontologijų bibliotekas, galimybėmis vykdyti *SPARQL* užklausas). Įrankis *Protégé* turi labiausiai pažengusias savybes, kurios susijusios su ontologijų konstravimu.

2.9 lentelė Įrankių palyginimas pagal įrankių panaudojimą

Savybė	*Protégé*	*SWOOP*	*IsaViz*
Bendradarbiavimas su kitais įrankiais	Ne	Ne	Ne
Ontologijų biblioteka	Taip	Ne	Ne
SPARQL palaikymas	Taip	Taip	Taip

Išanalizavus visus aspektus galima pastebėti, kad *Protégé* turi daugiau galimybių (kurios yra ir gali būti naudingos magistrinio darbo metu) dirbant su ontologijomis, negu *SWOOP* ar *IsaViz*. Ontologijas galima saugoti ne tik failuose, bet ir duomenų bazėje, taip pat palaiko pilną nuoseklumo tikrinimą, turi ontologijų biblioteką ir turi labiausiai pažengusias savybes, kurios susijusios su ontologijų konstravimu.

Įvertinus visus kriterijus buvo pasirinktas labiausiai išvystytas įrankis *Protégé*. Nors šiame įrankyje nėra galimybės peržiūrėti ontologijos egzempliorių grafiškai, tačiau yra suteiktos galimybės pasitikrinti susietumą tarp ontologijos egzempliorių. Taip pat šis įrankis yra palaikomas universiteto, populiariausias tarp vartotojų [16], bei lengvai išmokstamas.

Šių įrankių *SPARQL* palaikymas nėra tinkamas tyrimui vykdyti, nes nėra galimybės apskaičiuoti vykdymo laiką, dėl šios priežasties reikia turėti įrankį, kuriame būtų galima matuoti *SPARQL* užklausų vykdymo laiką.

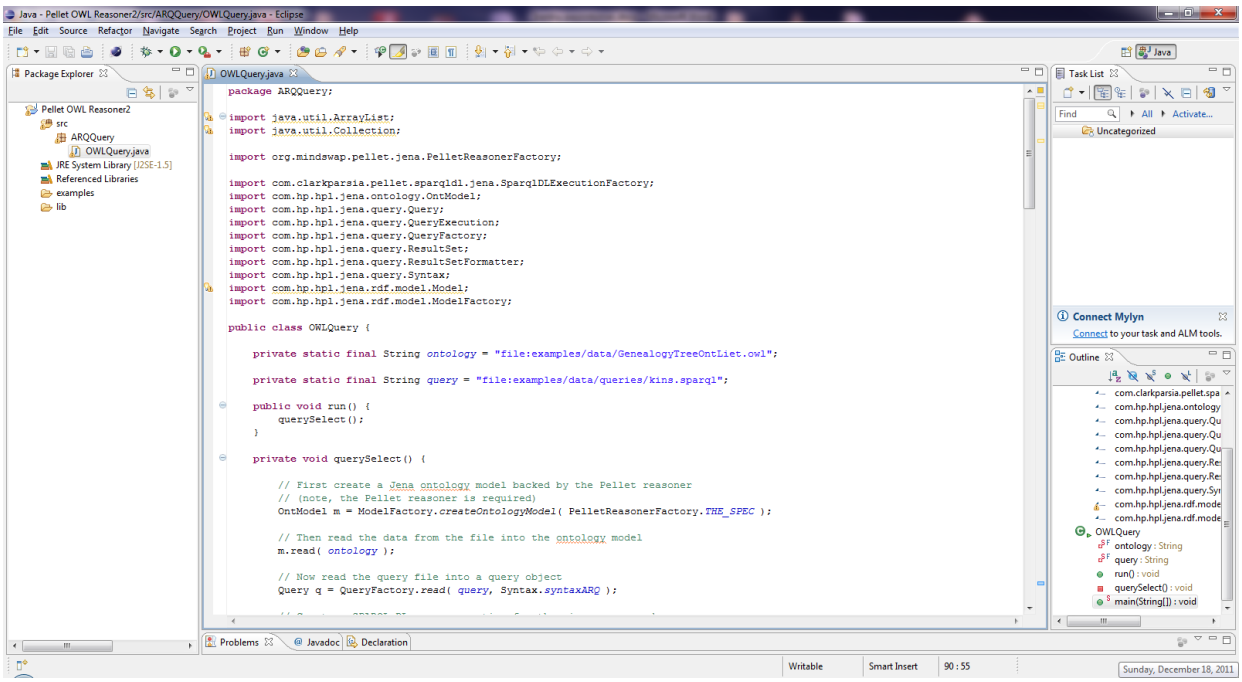
- ***SPARQL* užklausų vykdymo įrankis**

Tam, kad būtų galima atlikti eksperimentą, reikia turėti *SPARQL* užklausų vykdymo įrankį. Reikia pasirinkti išvedimo mechanizmą, kuris susidorotų su ontologijos įvairiomis charakteristikomis ir kad būtų galima vykdyti *SPARQL* užklausas. Buvo atlikta lyginamoji analizė šių išvedimo mechanizmų: *FaCT++*, *HermiT* ir *Pellet*. Jų lyginamoji analizė pateikta 2.10 lentelėje. Pateikiamos tos charakteristikos, kurios turi įtaką tyrimui.

2. 10 lentelė Išvedimo mechanizmų palyginimas

Išvedimo mechanizmas / **Savybės**	***FaCT++***	***HermiT***	***Pellet***
OWL-DL palaikymas	+	+	+
Nuoseklumo tikrinimas	+	+	+
SWRL taisyklių palaikymas	-	+	+
Galimybė vykdyti *SPARQL* užklausas	-	-	+

Kadangi *FaCT++* ir *HermiT* išvedimų mechanizmai nepalaiko galimybės vykdyti *SPARQL* užklausas (naudoja *OWL API*, kuris neturi palaikymo su *SPARQL*), o *FaCT++* dar ir nepalaiko *SWRL* taisyklių. Eksperimentui atlikti – *SPARQL* užklausoms vykdyti – buvo pasirinktas *Pellet OWL Reasoner* išvedimo mechanizmas. Kadangi tai yra atviro kodo *Java* pagrįstas *OWL DL* loginis išvedimo mechanizmas [14], buvo pasirinktas *Eclipse* įrankis *SPARQL* užklausoms vykdyti (sukuriant vykdymo metodą). Pasinaudojant *Pellet OWL Reasoner* galima susikurti metodą, kuris vykdytų *SPARQL* užklausas. Taip pat galima įvertinti vykdymo laiką įvedus papildomus kintamuosius. *Eclipse* programos langas su *Pellet OWL Reasoner* pateiktas 2.19 paveiksle.



2.19 pav. *Eclipse* programos langas

2.6. Analizės išvados

1. Išanalizavus *RDF*, *OWL* ir *SPARQL*, *SWRL* technologijas, paaiškėjo jų vaidmuo realizuojant Semantinio tinklo viziją. Kadangi *RDF* turi trūkumų (vietinės apimties savybės, nesikertančios klasės, nesikertančių klasių kombinacija, kardinalumo apribojimai, specialios klasės atributo charakteristikos), buvo sukurta *OWL*, kuri praplečia *RDF* galimybes ir suteikia didesnes paieškos galimybes. *SPARQL* užklausų kalba leidžia pagal aprašytas ontologijas gauti atsakymus pagal apibrėžtus reikalavimus.

2. *SPARQL* ir *SQL* užklausų kalbų palyginimas parodė, kad *SPARQL* užklausų kalba yra panaši į *SQL* užklausų kalbą, nes buvo kuriama jos pagrindu. *SPARQL* sintakse parašyta užklausa yra trumpesnė negu rašant *SQL* užklausų kalba, taip yra išvengiama galimų klaidų dėl sudėtingos ir ilgos užklausos.

3. Atlikta *SPARQL* užklausų kalbos analizė leido identifikuoti naujas *SPARQL 1.1* galimybes, sudaryti šių užklausų klasifikaciją ir išskirti tiriamus *SPARQL 1.1* užklausų tipus.

4. Esamų ontologijų užklausų vykdymo priemonių vertinimo etalonų analizė parodė, kad:

- nei vienas esamų etalonų neapima visų reikiamų ontologijų ir užklausų savybių, tarp jų ir naujų *SPARQL 1.1* savybių;
- nei vienas etalonas nenagrinėja rekursinių užklausų;
- nei vienas etalonas nenaudoja *SWRL* taisyklių užklausų optimizavimui;
- didžiausias dėmesys skiriamas Semantinio tinklo įrankių, bet ne užklausų tipų veikimui vertinti.

5. Šiai spragai užpildyti buvo nutarta sukurti užklausų rinkinį ir ontologijų generavimo įrankį bei išbandyti jį vertinant užklausų vykdymo priemonės *Pellet* galimybes.

6. Ontologijos egzempliorių generavimo įrankio realizavimui nebuvo keliami specialūs reikalavimai, todėl realizacijai pasirinkti atviro kodo įrankiai dėl gero suderinamumo: *PHP, MySQL, PhpMyAdmin, CSS.* Buvo pasirinkti ontologijų redaktorius *Protégé* bei *Pellet* išvedimo mechanizmas užklausų vykdymui tirti, nes tai labiausiai išvysyti ontologijų įrankiai atitinkantys keliamus reikalavimus.

3. Semantinių užklausų tyrimo metodika

3.1. Tyrimo metodika

SPARQL užklausų tyrimo metodika susideda iš 6 žingsnių:

1 žingsnis. Nustatyti tyrimo tikslą, pasirinkti vertinamas priemones.

2 žingsnis. Suplanuoti tyrimą: apibrėžti tiriamų ontologijų apimtis, techninės įrangos tipus, matuojamus parametrus, bandymų skaičių.

3 žingsnis. Ontologijos egzempliorių generavimo įrankiu sugeneruoti nustatytos apimties ontologijas.

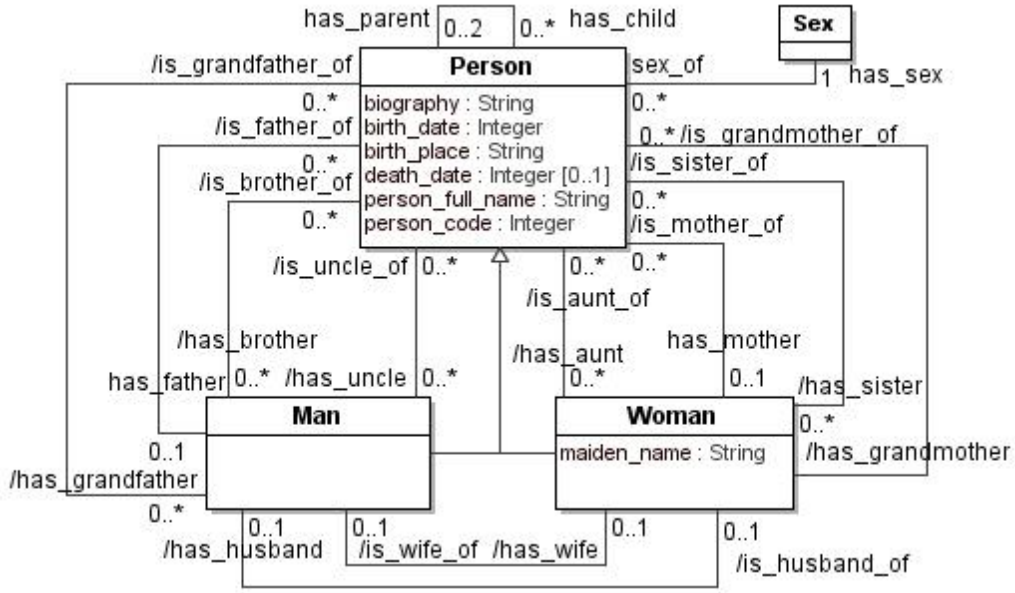
4 žingsnis. Vykdyti tipines užklausas įvairių dydžių ontologijose pagal sudarytą planą, fiksuoti rezultatus.

5 žingsnis. Įvertinant gautus rezultatus, modifikuoti tyrimo planą.

6 žingsnis. Apibendrinti gautus rezultatus, vaizdžiai juos pateikti, suformuluoti tyrimo išvadas.

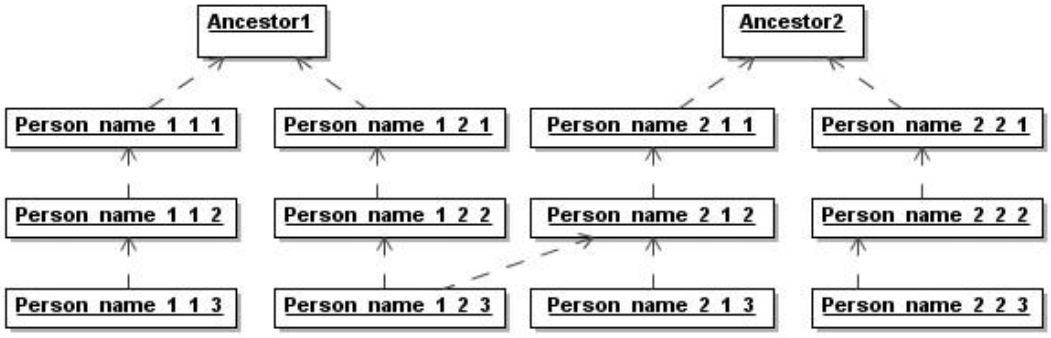
3.2. Genealoginio medžio ontologija tyrimui

SPARQL užklausų kalbos tyrimui (ontologijos egzempliorių generavimo įrankio dalykinės srities klasių diagrama) buvo pasirinkta genealoginio medžio *OWL* ontologija dėl rekursijos galimybės. Genealoginio medžio ontologija pateikta 3.1 paveiksle.



3.1 pav. Genealoginio medžio ontologijos klasių modelis

Ontologijoje egzemplioriams susieti jų generavimo metu naudojamas pagrindinis ryšys `has_parent`, turintis sau priešingą ryšį `has_child`, kuris atsiranda atliekant ontologijos išvedimą. Genealoginio medžio egzempliorių generavimui suteikiant tikrumo įtraukiamas ryšys `has_sex` turintis vieną reikšmę iš dviejų: vyras arba moteris. Ontologijos egzempliorių pavyzdys susietas `has_parent` ryšiais pateiktas 3.2 paveiksle.



3.2 pav. Ontologijos egzempliorių pavyzdys

Ontologijos dalykinės srities klasių diagramos (genealoginio medžio) pavadinimų paaiškinimai pateikti 4.6 lentelėje.

3.1lentelė Genealoginio medžio klasių diagramos pavadinimų paaiškinimai

Klasės pavadinimas	Vertimas/paaiškinimas
Person	Asmuo
Woman	Moteris
Man	Vyras
Sex	Lytis

3.3. Užklausų rinkinys tyrimui

Įvertinant *SPARQL 1.1* naujas galimybes buvo sudarytas užklausų rinkinys, kuris pateiktas 3.1 lentelėje. Tripletų ar įrašų modifikavimas, redagavimas ar šalinimas turėtų būti atliekamas ne tiesiogiai užklausomis, bet su atitinkama vartotojo sąsaja, todėl į rinkinį šios *SPARQL* užklausos neįtrauktos.

3. 2 lentelė Užklausų rinkinys

Nr.	Žymėjimas	Užklausa	Kodas
1.	Q1	Surasti visus vieno asmens gimines	`SELECT ?kin WHERE {gen:Person_name_1_2_2(gen:has_child|gen:has_parent)* ?kin}`
2.	Q2	Surasti visų asmenų visas gimines	`SELECT ?person ?kin WHERE {?person rdf:type gen:Person . ?person (gen:has_child|gen:has_parent)* ?kin FILTER ( ?person != ?kin )}`
3.	Q3	Surasti kartų skirtumą tarp dviejų asmenų	`SELECT (fn:abs((?count1 - ?count2)) AS ?range) WHERE {{SELECT (count(?x) AS ?count1) WHERE {gen:Person_name_1_1_1 (gen:has_parent)* ?x}} {SELECT (count(?x) AS ?count2) WHERE {gen:Person_name_1_1_2 (gen:has_parent)* ?x}}}`
4.	Q4	Surasti asmens pirmtaką	`SELECT ?ancestor WHERE {gen:Person_name_2_2_2 gen:has_parent* ?ancestor. NOT EXISTS {?ancestor gen:has_parent ?y}}`
5.	Q5	Surasti visų asmenų pirmtakus	`SELECT ?person ?ancestor WHERE {?person rdf:type gen:Person . ?person (gen:has_parent)* ?ancestor NOT EXISTS {?ancestor gen:has_parent ?y}}`

Nr.	Žymėjimas	Užklausa	Kodas
6.	Q6	Surasti asmens senelį (1) be *SWRL* ir (2) su *SWRL*	(1)SELECT ?antecedent WHERE {gen:Person_name_1_2_3 (gen:has_parent){2,2} ?antecedent} (2)SELECT ?antecedent WHERE {gen:Person_name_1_2_3 (gen:has_grandparent) ?antecedent}
7.	Q7	Surasti visų asmenų senelius (1) be *SWRL* ir (2) su *SWRL*	(1)SELECT ?person ?antecedent WHERE {?person (gen:has_parent){2,2} ?antecedent} (2)SELECT ?person ?antecedent WHERE {?person (gen:has_grandparent) ?antecedent}
8.	Q8	Surasti asmenis, kurie turi ne mažiau negu 2 vaikus	SELECT ?parent (count(?child) AS ?child_count) WHERE {?parent gen:has_child ?child} GROUP BY ?parent HAVING (?child_count > 1)
9.	Q9	Surasti nurodytos kartos asmenis	SELECT ?people (count(?ancestor) AS ?range) WHERE {?people rdf:type gen:Person . ?people (gen:has_parent)* ?ancestor} GROUP BY ?people HAVING (?range = 2)
10.	Q10	Surasti visus asmenis, kurie yra paskutiniai kartoje	SELECT DISTINCT ?last WHERE {?people rdf:type gen:Person . ?people (gen:has_child)* ?last NOT EXISTS {?last gen:has_child ?x}}
11.	Q11	Surasti visų asmenų dėdes (1) be *SWRL* ir (2) su *SWRL*	(1)SELECT ?person ?uncle WHERE {?person gen:has_parent ?x . ?x gen:has_parent ?y . ?y gen:has_child ?uncle . ?uncle gen:has_sex gen:male_sex FILTER (?x != ?uncle)} (2)SELECT ?person ?uncle WHERE {?person (gen:has_uncle) ?uncle}
12.		*SWRL*, papildomai aprašomos *SWRL* taisyklės	has_parent(?x, ?y), has_parent(?y, ?z) -> has_grandparent(?x, ?z); Man(?z), has_child(?y, ?z), has_grandparent(?x, ?y), has_parent(?x, ?f), DifferentFrom (?z, ?f) -> has_uncle(?x, ?z);

Pasinaudojus šiame darbe sukurtu ontologijos egzempliorių generavimo įrankiu sugeneruojami genealoginio medžio ontologijos su skirtingu egzempliorių skaičiumi, jose vykdomos *SPARQL* užklausų rinkinio užklausos, fiksuojami ontologijos užrovimo ir užklausų vykdymo laikai. Tyrimas atliekamas su skirtingus parametrus turinčia kompiuterine įranga.

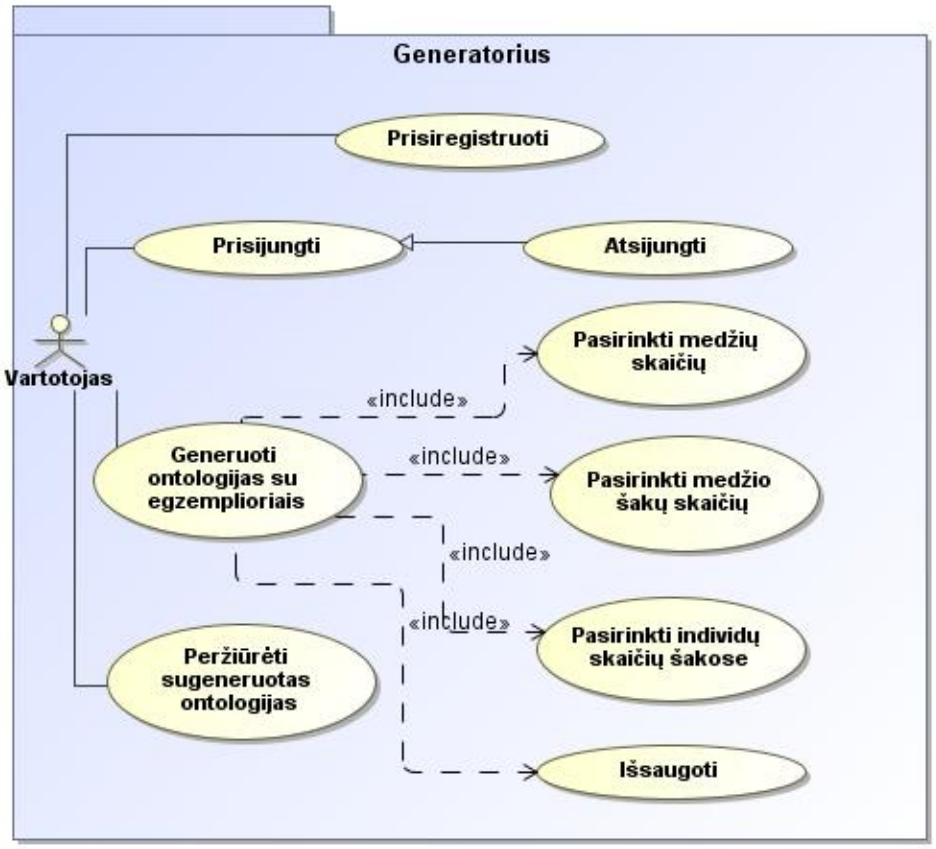
4. Ontologijos egzempliorių generavimo įrankio projektas ir realizacija

4.1. Reikalavimai

4.1.1. Reikalavimų specifikacija

Funkciniai reikalavimai

Kompiuterizuojamų panaudojimo atvejų diagrama, kuri vaizduoja ontologijos egzempliorių generatoriaus (ontologijos egzempliorių generavimo įrankio) veikimo funkcijas, pateikta 4.1 paveiksle. Generatoriuje galima atlikti tokias funkcijas: prisiregistruoti, prisijungti, atsijungti, generuoti ontologiją su egzemplioriais (tuomet vartotojas turi įvesti medžių, medžių šakų, individų skaičius), peržiūrėti sugeneruotas ontologijas.



4. 1 pav. Panaudojimo atvejų diagrama

Kompiuterizuojamų panaudojimo atvejų specifikacijos pateikiamos specifikacijos lentelėse. Taip pat pateikiamos panaudojimo atvejų veiklos diagramos, kurios susijusios su ontologijos su egzemplioriais generavimu ir sugeneruotų ontologijų peržiūra.

Panaudojimo atvejo „Prisiregistruoti“ specifikacija pateikiama 4.1 lentelėje.

4.1 lentelė Panaudojimo atvejo „Prisiregistruoti“ specifikacija

PA „Prisijungti“		
Tikslas. Prisiregistruoti prie generavimo įrankio		
Aprašymas. Šis PA parodo kaip prisiregistruoti prie ontologijų generavimo įrankio		
Prieš sąlyga		Aktorius įsijungęs interneto naršyklę, pasirinkęs registraciją
Aktorius		Vartotojas
Sužadinimo sąlyga		Vartotojas nori prisiregistruoti prie generavimo įrankio
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		Sistemos reakcija ir sprendimai
1. Vartotojas užpildo registracijos duomenis		
2. Vartotojas pasirenka „Registruoti“		2.1. Sistema patikrina vartotojo duomenis
		2.2. Sistema sugeneruoja slaptažodį
		2.3. Sistema išsiunčia laišką su prisijungimo duomenimis
3. Vartotojas baigia PA		
Po sąlyga:		Vartotojas yra prisiregistravęs ir gali prisijungti prie sistemos
Alternatyvūs scenarijai		
2.1.a		Sistema patikrinus duomenis praneša, kad el. paštas įvestas nekorektiškai
2.1.b		Sistema patikrinus duomenis praneša, kad tokiu el. paštu jau registruotas vartotojas
Pastabos		
1. Vartotojas gali bet kada baigti PA		

Panaudojimo atvejo „Prisijungti“ specifikacija pateikiama 4.2 lentelėje.

4. 2 lentelė Panaudojimo atvejo „Prisijungti“ specifikacija

PA „Prisijungti“		
Tikslas. Prisijungti prie generavimo įrankio		
Aprašymas. Šis PA parodo kaip prisijungti prie ontologijų generavimo įrankio		
Prieš sąlyga		Aktorius įsijungęs interneto naršyklę
Aktorius		Vartotojas
Sužadinimo sąlyga		Vartotojas nori prisijungti prie generavimo įrankio
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	Atsijungti
Pagrindinis įvykių srautas		Sistemos reakcija ir sprendimai
1. Vartotojas įrašo prisijungimo duomenis		
2. Vartotojas pasirenka „Prisijungti“		2.1. Sistema patikrina vartotojo duomenis ir prijungia vartotoją
3. Vartotojas baigia PA		
Po sąlyga:		Vartotojas prisijungęs prie generavimo įrankio
Alternatyvūs scenarijai		
2.1.a		Sistema patikrinus duomenis praneša, kad toks vartotojas yra neregistruotas
Pastabos		
1. Vartotojas gali bet kada baigti PA		

Panaudojimo atvejo „Atsijungti“ specifikacija pateikiama 4.3 lentelėje.

4. 3 lentelė Panaudojimo atvejo „Atsijungti“ specifikacija

PA „Atsijungti“		
Tikslas. Atsijungti nuo generavimo įrankio		
Aprašymas. Šis PA parodo kaip atsijungti nuo generavimo įrankio		
Prieš sąlyga		Aktorius prisijungęs prie generavimo sistemos
Aktorius		Vartotojas
Sužadinimo sąlyga		Vartotojas nori atsijungti nuo generavimo įrankio
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka „Atsijungti“		1.1. Sistema atjungia vartotoją nuo generavimo įrankio
2. Vartotojas baigia PA		
Po sąlyga:		Vartotojas atsijungęs nuo generavimo įrankio
Alternatyvūs scenarijai		
Pastabos		
1. Vartotojas gali bet kada baigti PA		

Panaudojimo atvejo „Generuoti ontologijas su egzemplioriais“ specifikacija pateikiama 4.4 lentelėje.

4. 4 lentelė Panaudojimo atvejo „Generuoti ontologijas su egzemplioriais“ specifikacija

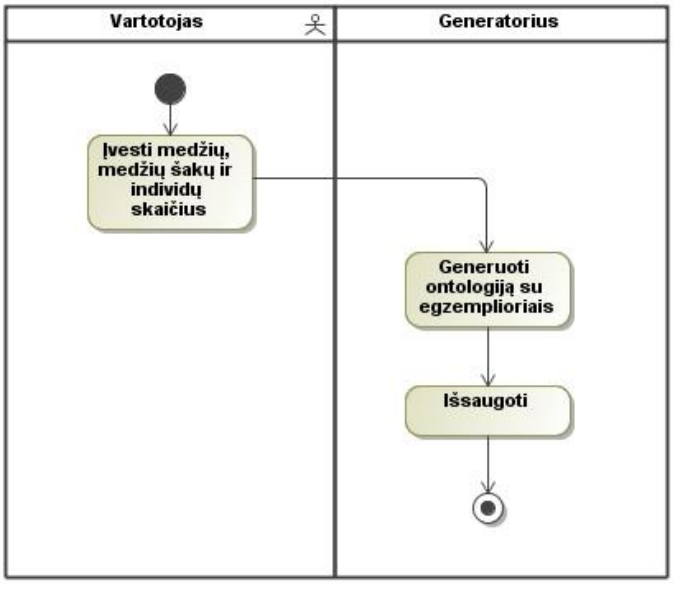
PA „Generuoti ontologijas su egzemplioriais“		
Tikslas. Susigeneruoti ontologijos egzempliorius		
Aprašymas. Šis PA parodo kaip susigeneruoti ontologijų egzempliorius		
Prieš sąlyga		Aktorius įsijungęs generavimo programos generavimo funkciją
Aktorius		Vartotojas
Sužadinimo sąlyga		Vartotojas nori susigeneruoti ontologijos egzempliorius
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		Sistemos reakcija ir sprendimai
1. Vartotojas įrašo medžių, medžio šakų ir individų skaičius		
2. Vartotojas pasirenka „Generuoti“		2.1. Sistema sugeneruoja ontologijų egzempliorius
3. Vartotojas baigia PA		
Po sąlyga:		Išsisaugota sugeneruota ontologija su egzemplioriais
Alternatyvūs scenarijai		
Pastabos		
1. Vartotojas gali bet kada baigti PA		

Panaudojimo atvejo „Peržiūrėti sugeneruotas ontologijas“ specifikacija pateikiama 4.5 lentelėje.

4. 5 lentelė Panaudojimo atvejo „Peržiūrėti sugeneruotas ontologijas“ specifikacija

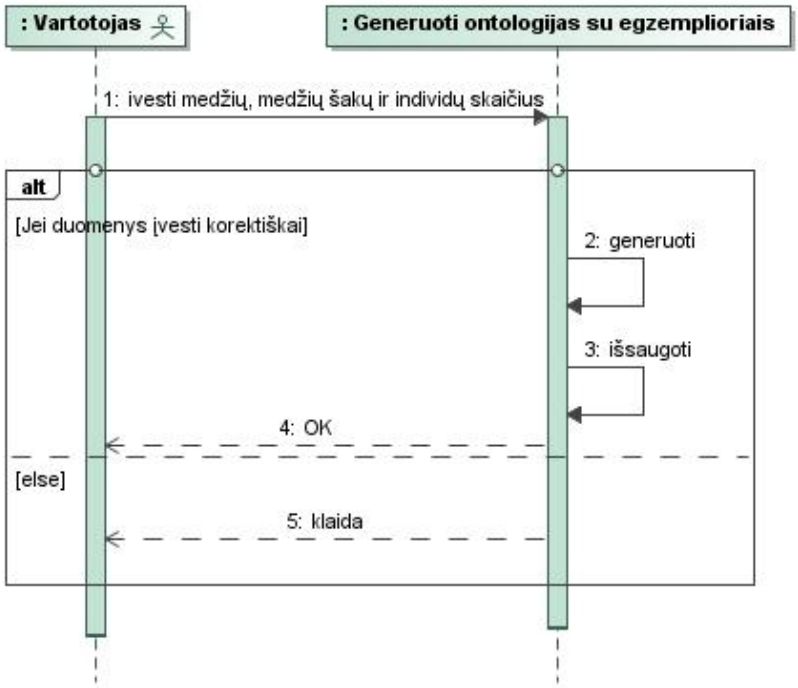
PA „Peržiūrėti sugeneruotas ontologijas“		
Tikslas. Peržiūrėti sugeneruotas ontologijas		
Aprašymas. Šis PA parodo kaip peržiūrėti vartotojo sugeneruotas ontologijas		
Prieš sąlyga		Aktorius prisijungęs prie generavimo sistemos
Aktorius		Vartotojas
Sužadinimo sąlyga		Vartotojas nori peržiūrėti sugeneruotas ontologijas
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		Sistemos reakcija ir sprendimai
1. Vartotojas pasirenka ontologijų peržiūrą		1.1. Sistema pateikia vartotojo sugeneruotų ontologijų su egzemplioriais failų sąrašą
2. Vartotojas pasirenka vieną ontologijos failą		2.1. Sistema pateikia pasirinkto ontologijos failo turinį
3. Vartotojas pasirenka ontologijos failą išsaugojimui		3.1. Sistema pateikia failą išsaugojimui
4. Vartotojas baigia PA		
Po sąlyga:		Pateiktas sugeneruotos ontologijos turinys vartotojui
Alternatyvūs scenarijai		
Pastabos		
1. Vartotojas gali bet kada baigti PA		

Norint susigeneruoti ontologijų egzempliorius, reikia vartotojui įrašyti medžių, medžių šakų ir individų skaičius, tuomet patvirtinti pasirinkimą (spausti „Generuoti“). Sistema pagal įvestus duomenis sugeneruoja ontologiją su jos egzemplioriais ir išsaugo gautą rezultatą (failą). Panaudojimo atvejo „Generuoti ontologijas su egzemplioriais“ veiklos diagrama pateikta 4.2 paveiksle.



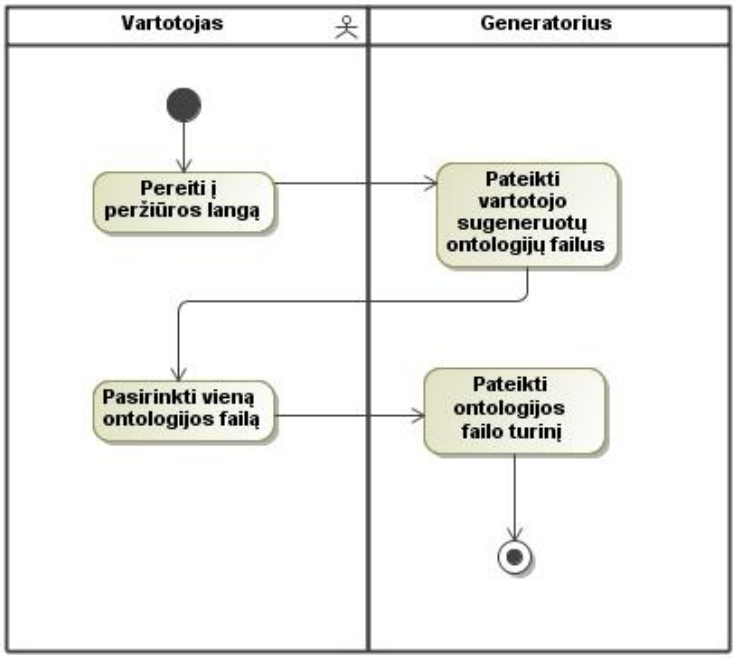
4.2 pav. Panaudojimo atvejo „Generuoti ontologijas su egzemplioriais“ veiklos diagrama

Panaudojimo atvejo „Generuoti ontologijas su egzemplioriais“ sekų diagrama pateikta 4.3 paveiksle.



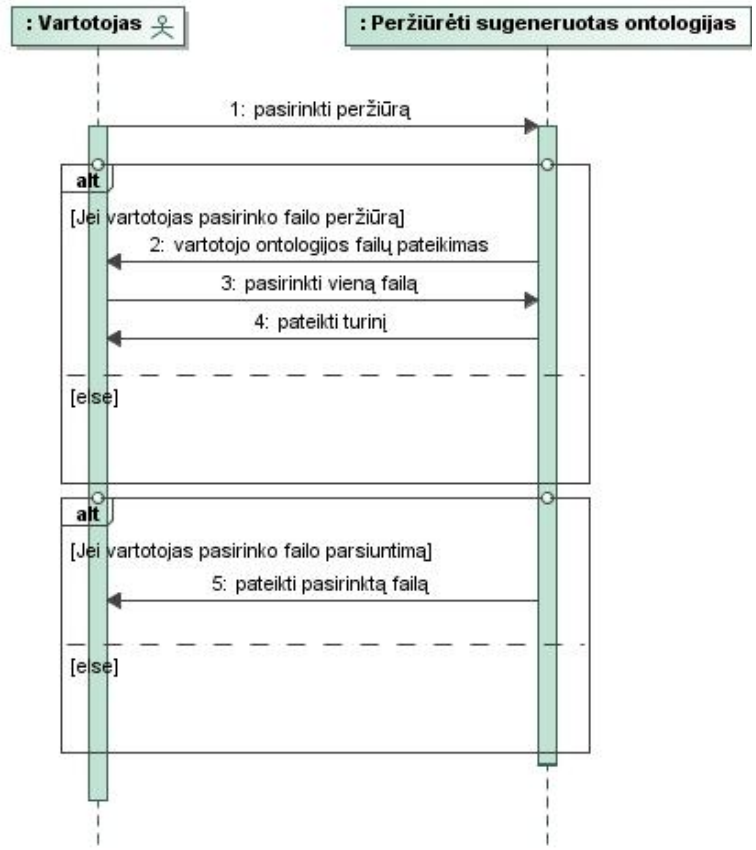
4.3 pav. Panaudojimo atvejo „Generuoti ontologijas su egzemplioriais“ sekų diagrama

Sugeneruotas ontologijas (su ontologijos egzemplioriais) vartotojas gali peržiūrėti. Panaudojimo atvejo „Peržiūrėti sugeneruotas ontologijas“ veiklos diagrama pateikta 4.4 paveiksle.



4.4 pav. Panaudojimo atvejo „Peržiūrėti sugeneruotas ontologijas“ veiklos diagrama

Panaudojimo atvejo „Peržiūrėti sugeneruotas ontologijas“ sekų diagrama pateikta 4.5 paveiksle.



4.5 pav. Panaudojimo atvejo „Peržiūrėti sugeneruotas ontologijas“ sekų diagrama.

Nefunkciniai reikalavimai

Generatoriaus paskirtis: generatorius naudojamas *SPARQL* užklausų kalbos galimybių tyrimui. Didžiausias dėmesys – rekursinėms užklausoms, pagal tai pasirinkta ontologija.

Naudojama ontologija: naudojama genealoginio medžio ontologija.

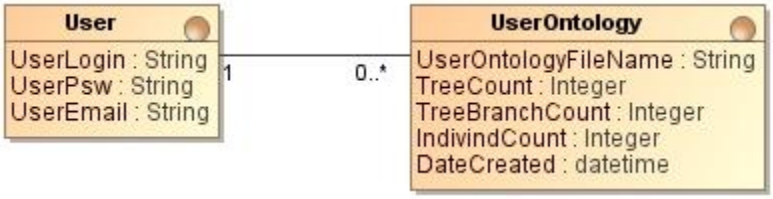
Reikalavimas sprendimui: generatorius kuriamas naudojant *PHP*, *MySQL* technologijas.

Diegimo aplinka: *Windows 7* operacinė sistema.

4.1.2. Dalykinės srities modelis

Genealoginio medžio *OWL* ontologija, kaip dalykinės srities modelis, pateikta 3 šio darbo skyriuje 3.1 paveiksle. Ontologijos dalykinės srities klasių diagramos (genealoginio medžio) pavadinimų vertimai pateikti 3 skyriaus 3.1 lentelėje.

Generatoriaus esybių klasių diagrama pateikta 4.6 paveiksle.



4.6 pav. Generatoriaus esybių klasių modelis

4.1.3. Reikalavimų apibendrinimas

Šiame skyriuje buvo aptarti funkciniai ir nefunkciniai reikalavimai ontologijos ir jos egzempliorių generatoriui. Pagrindinės įrankio (generatoriaus) funkcijos yra: ontologijos egzempliorių generavimas ir sugeneruotų ontologijų peržiūra.

Funkciniai reikalavimai buvo pateikti kompiuterizuojamų panaudojimo atvejų diagrama (4.1 paveikslas), specifikavimo lentelėmis (4.1 – 4.5 lentelės), veiklos (4.2, 4.4 paveikslai) ir sekų (4.3, 4.5 paveikslai) diagramomis.

Nefunkciniai reikalavimai buvo pateikti ontologijos egzempliorių generavimo įrankiui.

Ontologijos dalykinės srities klasių modelis (4.6 paveikslas) ir modelio klasių pavadinimų vertimai (4.6 lentelė) pateikti pasirinkus genealoginio medžio ontologiją, generatoriaus klasių modelis pateiktas 4.7 paveiksle.

Ontologijos egzempliorių generavimo įrankio pagrindinė idėja – suteikti galimybę vartotojui susigeneruoti ontologijos su skirtingais egzempliorių skaičiumi failą ar failus, kuriuos būtų galima naudoti *SPARQL* užklausų kalbos tyrimą su pasirinktu užklausų vykdymo įrankiu.

4.2. Projektas

4.2.1. Sistemos sprendimo pagrindimas ir esmės išdėstymas

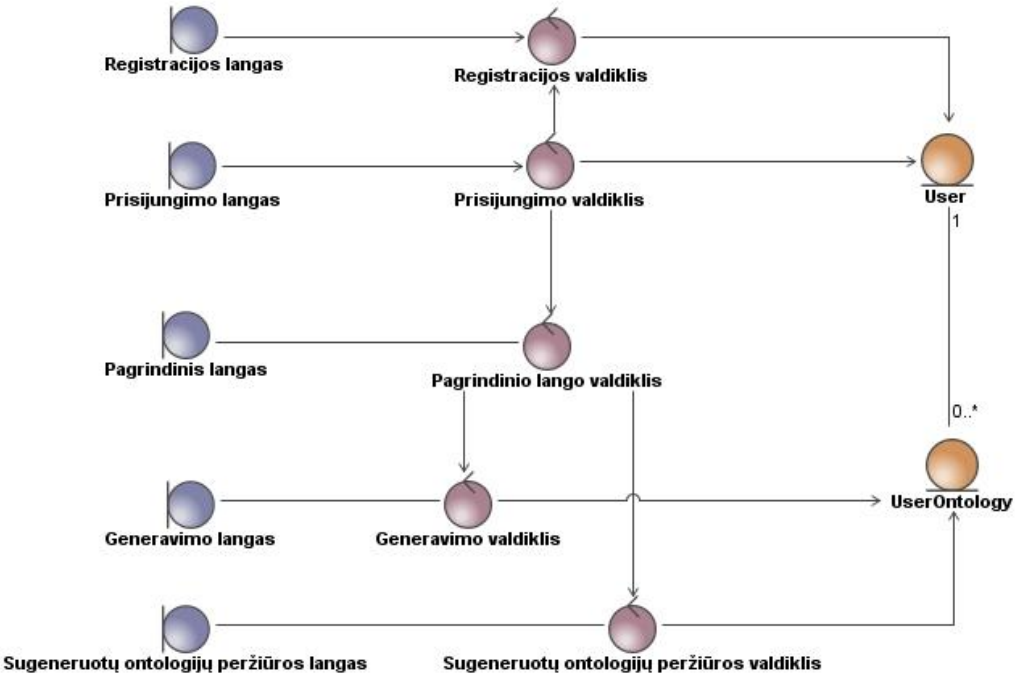
Ontologijos egzempliorių generavimo įrankis yra *SPARQL* užklausų tyrimo metodikos dalis. Šiuo įrankiu sugeneruoti ontologijų failai su skirtingu egzempliorių skaičiumi naudojami *SPARQL* užklausų tyrimui.

4.2.2. Sistemos architektūra

Reikalavimų analizė

Projekto tikslas – suprojektuoti ontologijos egzempliorių generavimo įrankį naudojant *CASE* įrankius. Identifikuoti vartotojo sąsają apibrėžiant vartotojo sąsajos navigavimo planą. Nustatyti reikalingas sistemai programines klases, kurios realizuotų funkcijas ir sąsajos langus, kurie yra reikalingi sistemai bei suprojektuoti duomenų bazės modelį.

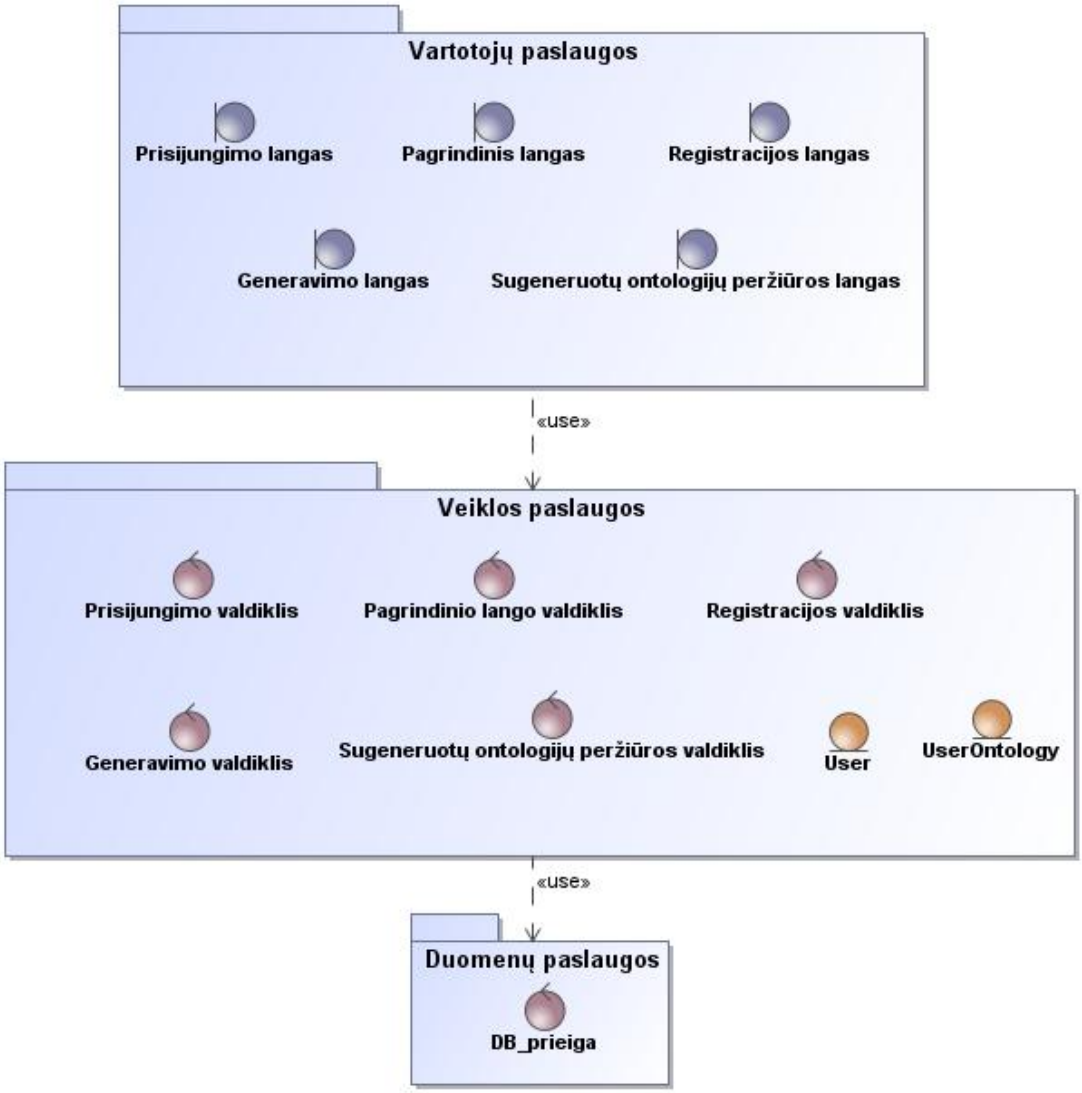
Analizės (robastiškumo) diagramos parodo, kokios vartotojo sąsajos, programinės ir esybių (duomenų) klasės turi būti sukurtos, norint realizuoti kompiuterizuojamus panaudojimo atvejus. Prisijungęs vartotojas patenka į pagrindinį langą, kuriame gali pasirinkti, kokį veiksmą nori atlikti: generuoti, peržiūrėti sugeneruotus failus ar atsijungti nuo generavimo sistemos. Dar neprisijungęs, vartotojas gali prisiregistruoti, kaip naujas sistemos vartotojas. Robastiškumo diagrama pateikiama 4.7 paveiksle.



4.7 pav. Robastiškumo diagrama

Loginė visos sistemos architektūra

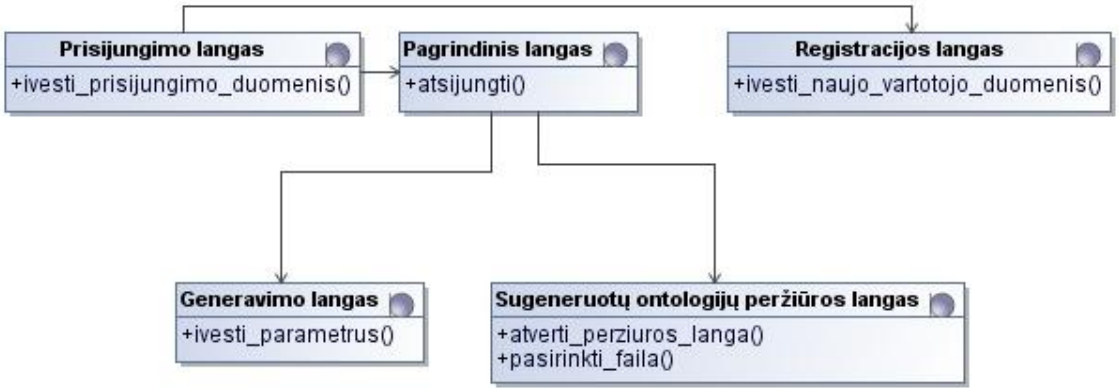
Sistemos loginė architektūra susideda iš vartotojo, veiklos ir duomenų paslaugų. Ontologijos egzempliorių generavimo įrankio sistemos loginė architektūra su ja sudarančiais elementais pateikiama 4.8 paveiksle.



4.8 pav. Sistemos loginė architektūra

Vartotojo paslaugos

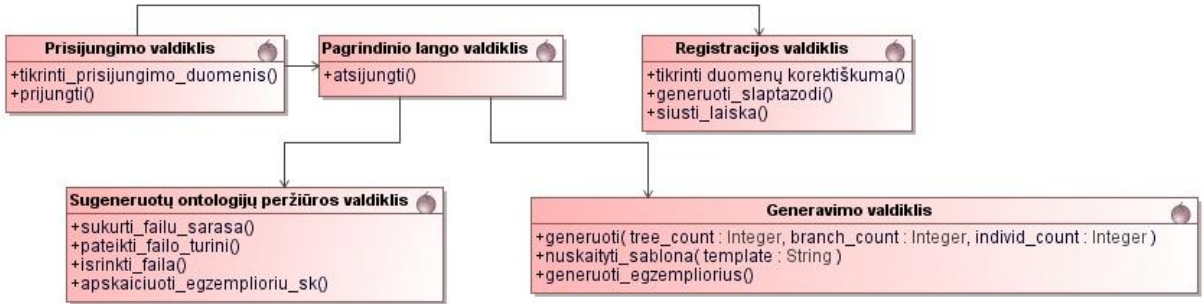
Vartotojas prisijungęs prie sistemos patenka į pagrindinį langą. Jame vartotojas gali pasirinkti, kokius veiksmus nori atlikti. Neprisijungęs vartotojas gali patekti į registracijos langą. Vartojo sąsajos navigavimo planas yra pateiktas 4.9 paveiksle.



4.9 pav. Vartotojo sąsajos navigavimo planas

Veiklos paslaugos

Valdymo klasių modelis yra pateiktas 4.10 paveiksle. Šiame modelyje yra pavaizduotos klasės ir jų metodai, kurie yra aprašyti lentelėse 4.2.5 skyriuje. Metodų parametrų sąrašas nenurodytas.



4.10 pav. Valdymo klasių modelis

Duomenų paslaugos

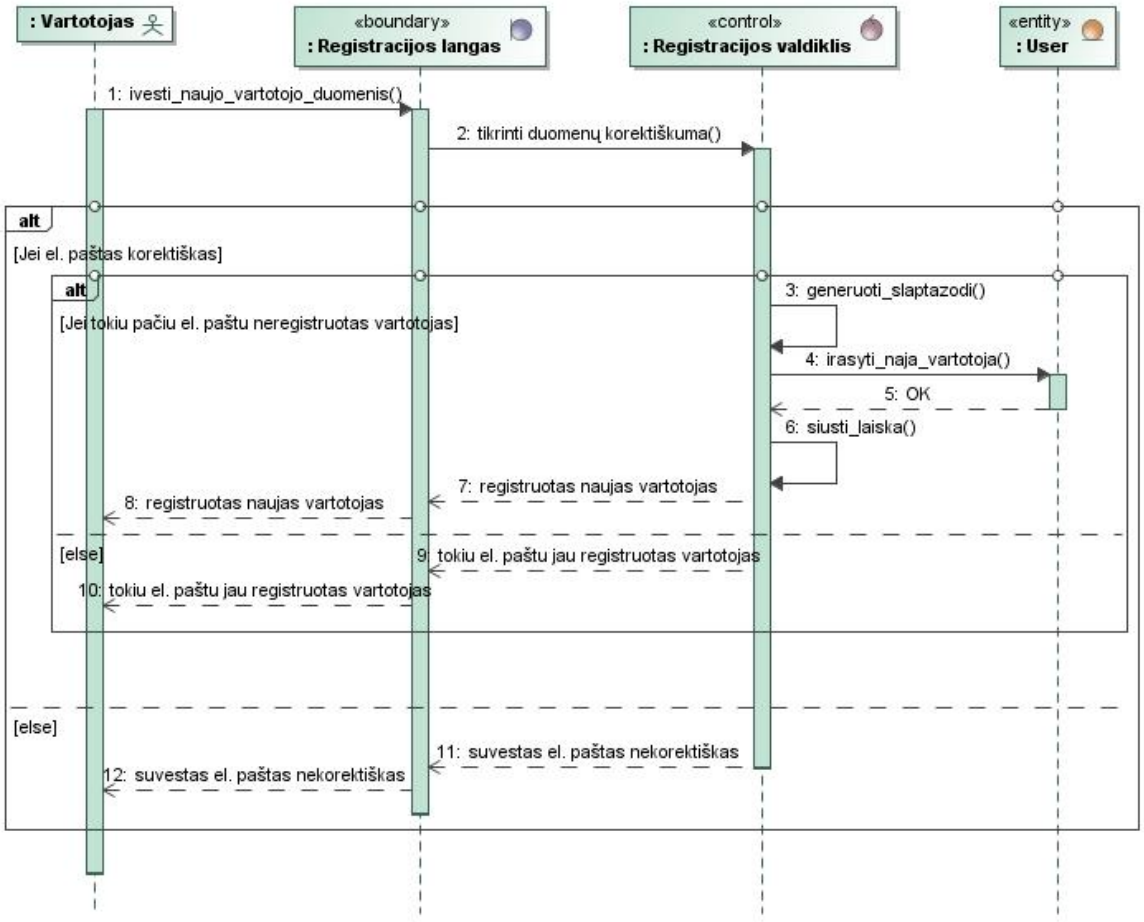
Duomenų paslaugas sudaro DB_prieiga, kuri pateikta 4.11 paveiksle. Ji reikalinga, kad būtų galima ontologijos egzempliorių generavimo įrankiui pasiekti duomenų bazėje esančius duomenis.

4.11 pav. DB_prieiga

4.2.3. Sistemos elgsenos modelis

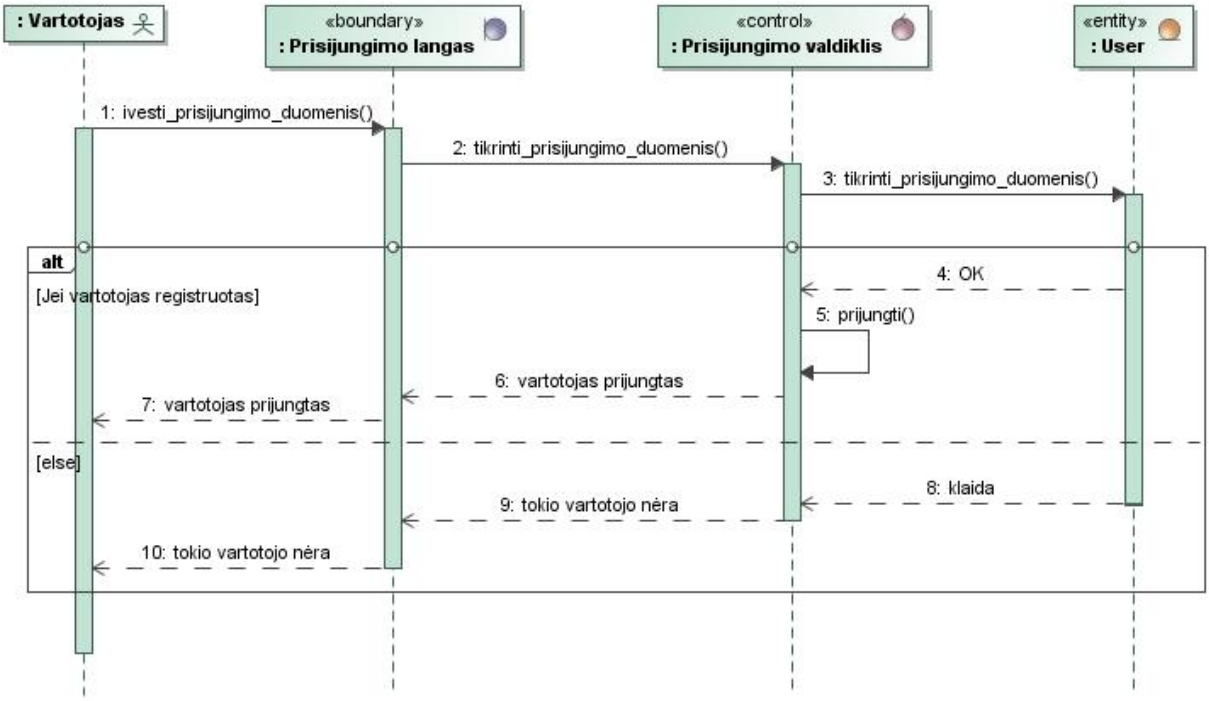
Panaudojimo atvejų diagramos, kurios parodo bendravimą tarp klasių, valdiklių ir langų yra pateikiamos šiame skyriuje. Metodai, kurie kreipiasi į esybių klases yra užklausos, kurios atlieka tam tikrus veiksmus su duomenimis. Jei užklausa įvykdyta sėkmingai, tuomet yra grąžinamas rezultatas *OK*.

Panaudojimo atvejo „Prisiregistruoti“ sekų diagrama vaizduojama 4.12 paveiksle.



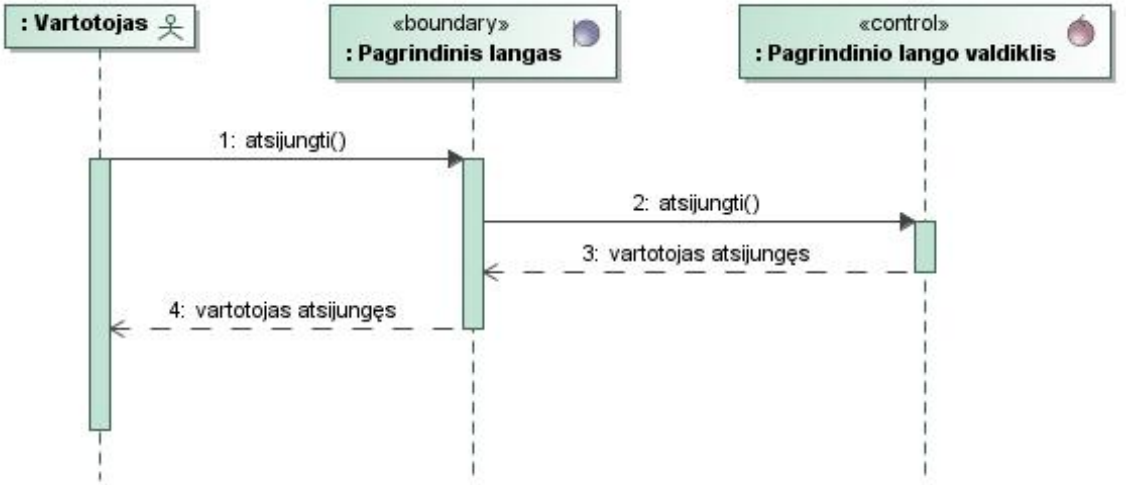
4.12 pav. Panaudojimo atvejo „Prisiregistruoti“ sekų diagrama

Panaudojimo atvejo „Prisijungti“ sekų diagrama vaizduojama 4.13 paveiksle.



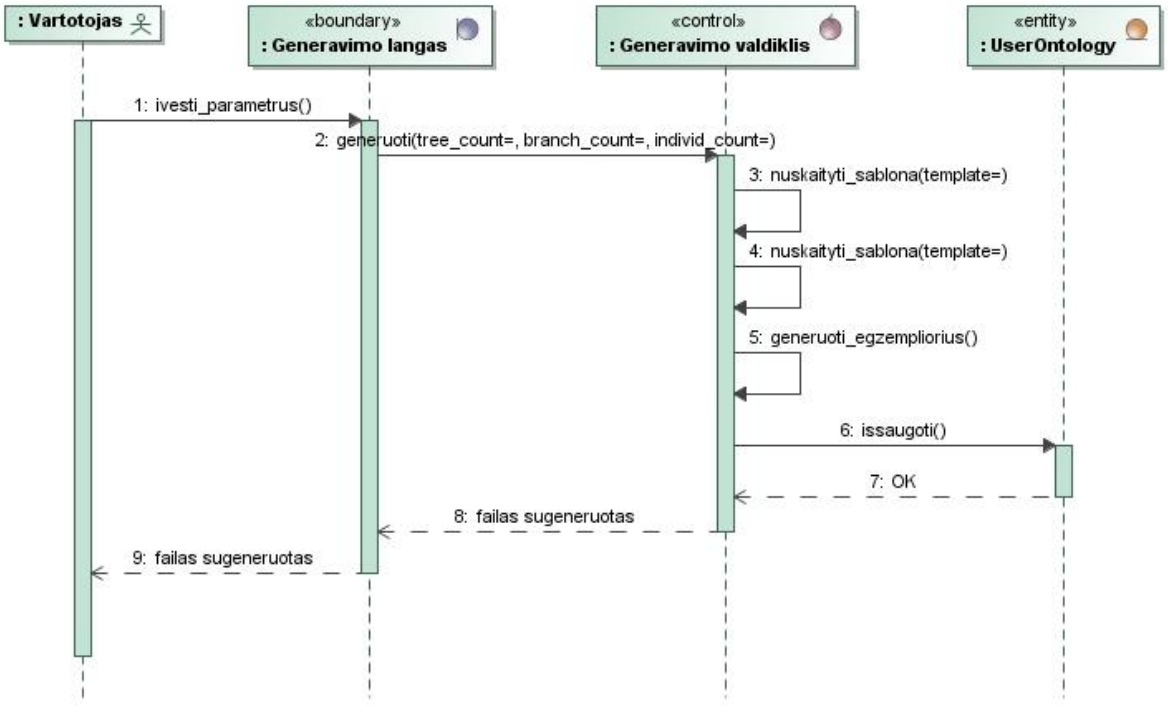
4.13 pav. Panaudojimo atvejo „Prisijungti“ sekų diagrama

Panaudojimo atvejo „Atsijungti“ sekų diagrama vaizduojama 4.14 paveiksle.



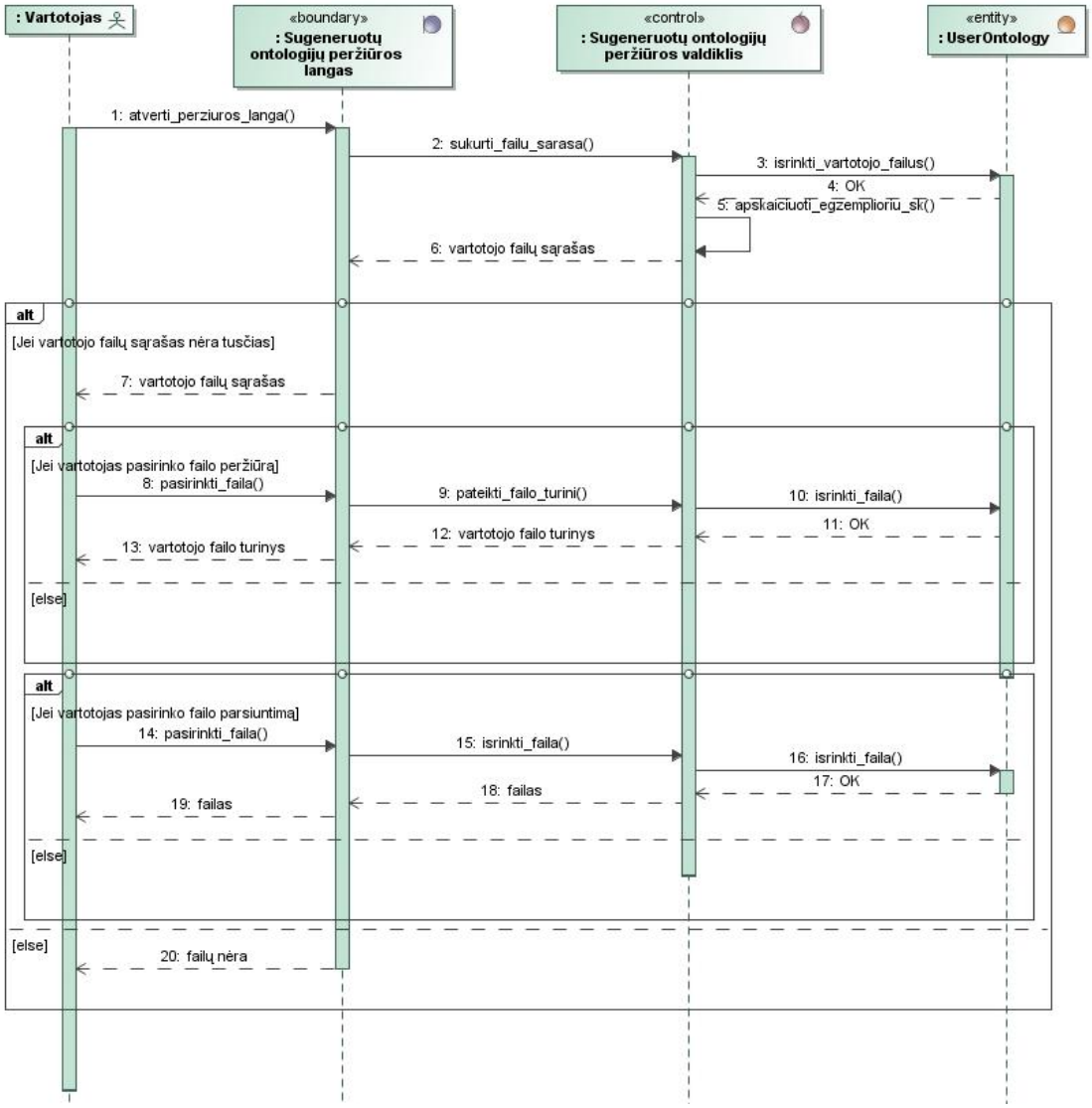
4.14 pav. Panaudojimo atvejo „Atsijungti“ sekų diagrama

Panaudojimo atvejo „Generuoti ontologijų egzempliorius“ sekų diagrama vaizduojama 4.15 paveiksle.



4.15 pav. Panaudojimo atvejo „Generuoti ontologijų egzempliorius“ sekų diagrama

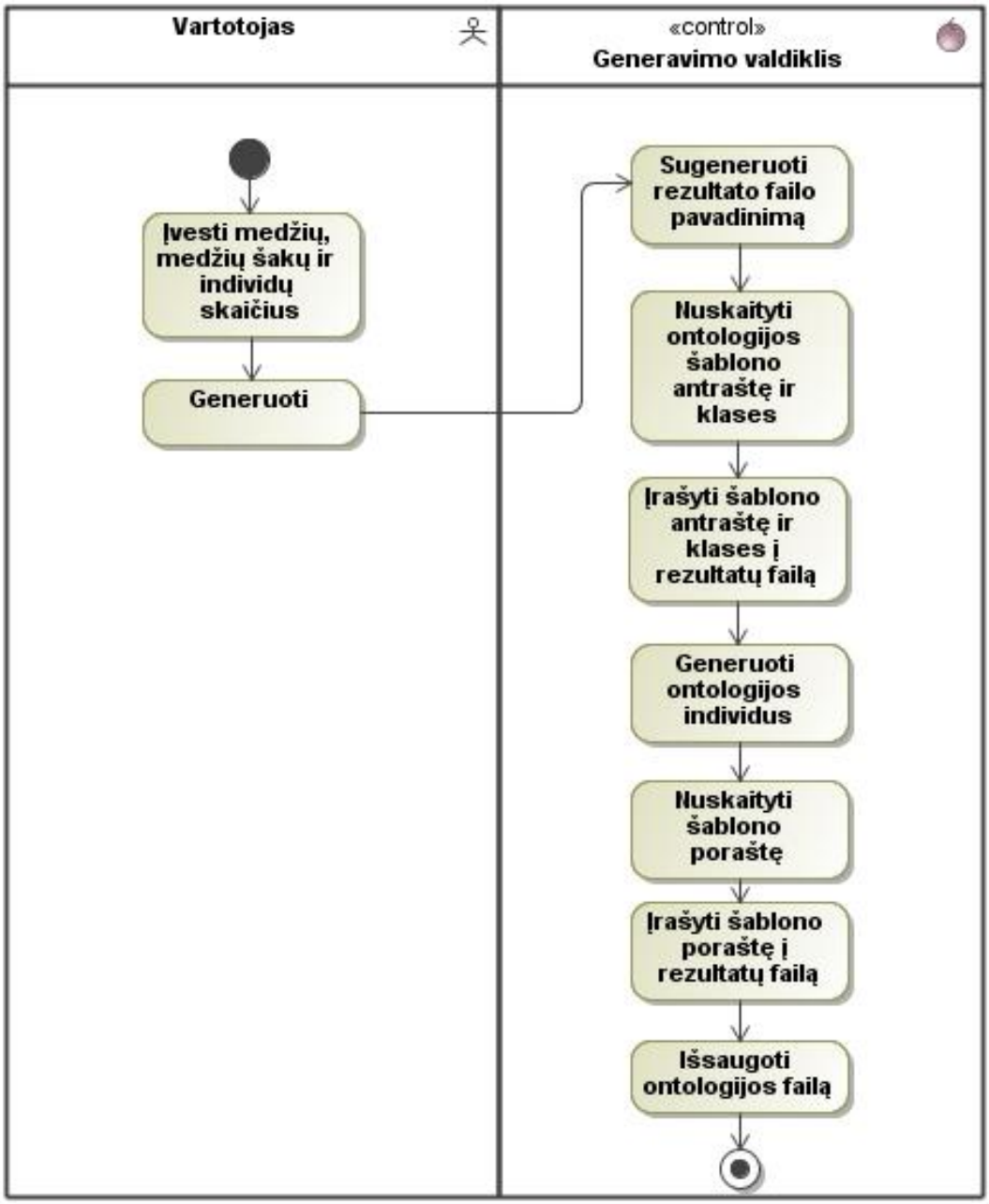
Panaudojimo atvejo „Peržiūrėti sugeneruotas ontologijas“ sekų diagrama vaizduojama 4.16 paveiksle.



4.16 pav. Panaudojimo atvejo „Peržiūrėti sugeneruotas ontologijas“ sekų diagrama

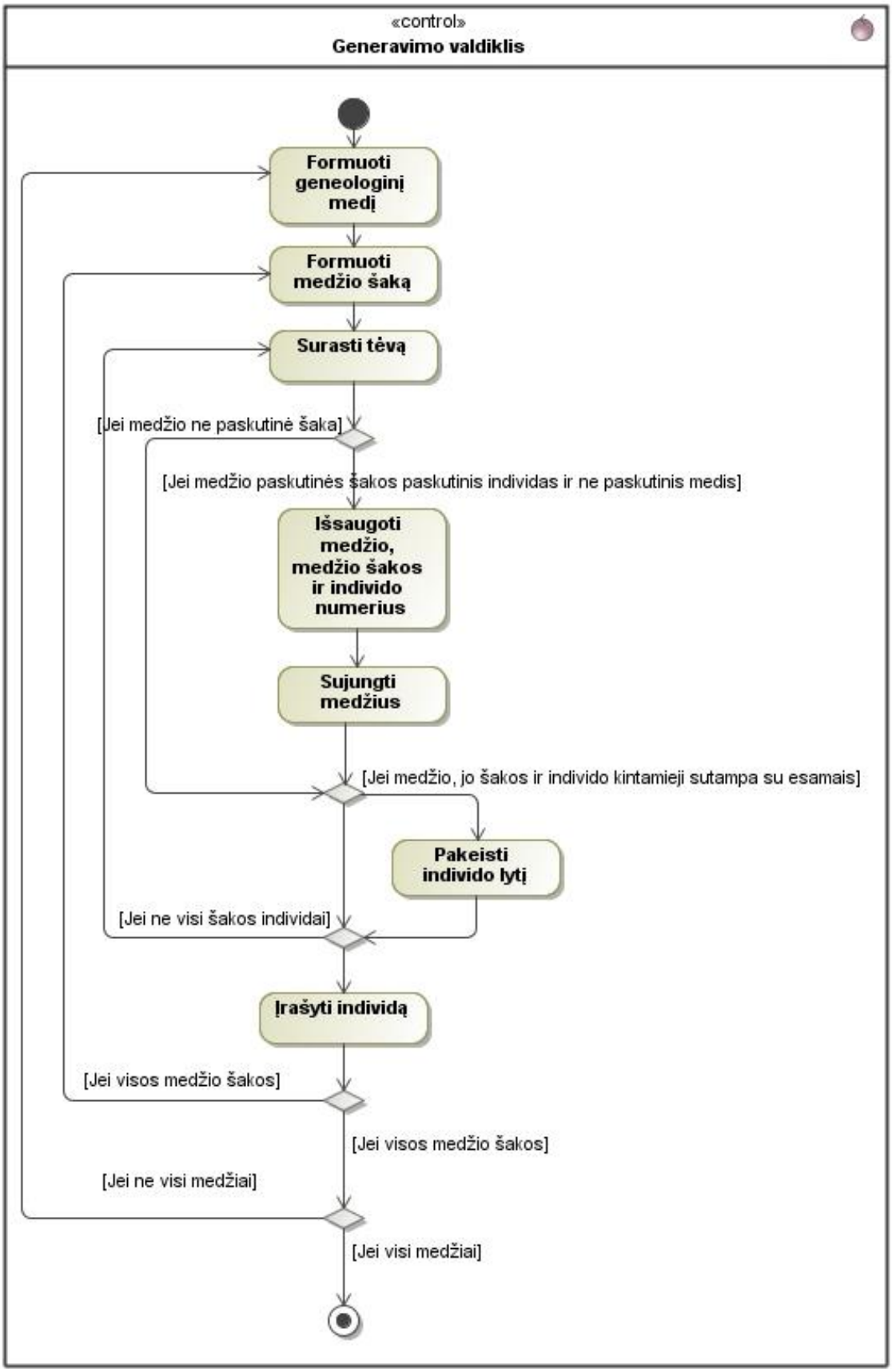
Ontologijos egzempliorių generavimo įrankio pagrindinė funkcija yra ontologijos egzempliorių generavimas. Veiklos diagramos, kurios pateiktos 4.17 ir 4.18 paveiksluose, parodo kaip yra sugeneruojamas ontologijos failas pagal vartotojo įvestus parametrus.

Veiklos diagrama, pateikta 4.17 paveiksle, nurodo vartotojo ir generatoriaus sąveiką.



4.17 pav. Ontologijos failo generavimo algoritmas

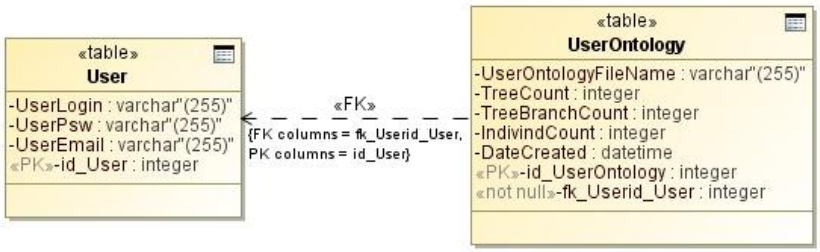
Veiklos diagrama, pateikta 4.18 paveiksle, nurodo generatoriaus vidinius veiksmus, kai vartotojas suveda reikalingus duomenis ontologijos individų generavimui, parodo detaliau, kas vyksta „Generuoti ontologijos individus“ veiklos metu.



4.18 pav. „Generuoti ontologijos individus“ veikla detaliau

4.2.4. Duomenų bazės schema

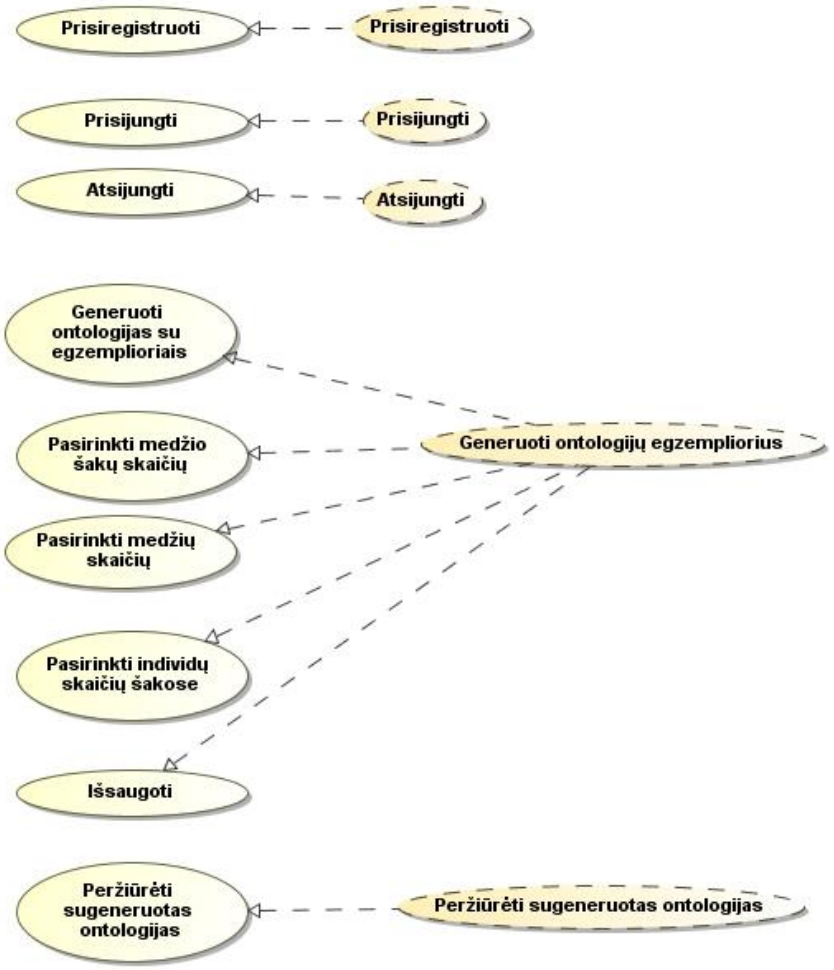
Duomenų bazės schema pateikta 4.19 paveiksle. Šiame paveiksle pateiktos lentelės, kurios sudaro ontologijų generavimo įrankio duomenų bazę.



4.19 pav. Duomenų bazės schema

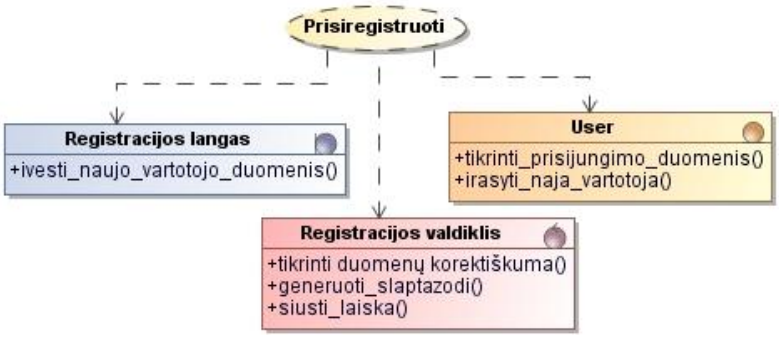
4.2.5. Detalus projektas

Panaudojimo atvejų realizacijos yra pateiktos 4.20 paveiksle. Šioje diagramoje yra vaizduojami visi panaudojimo atvejai, kurie yra kompiuterizuojami realizuojant ontologijos egzempliorių generavimo įrankį.



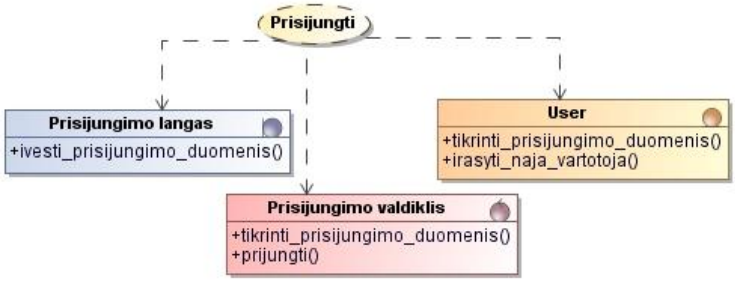
4.20 pav. Panaudojimo atvejų realizacijos

Panaudojimo atvejo „Prisiregistruoti“ realizacijos klasių diagrama yra vaizduojama 4.21 paveiksle.



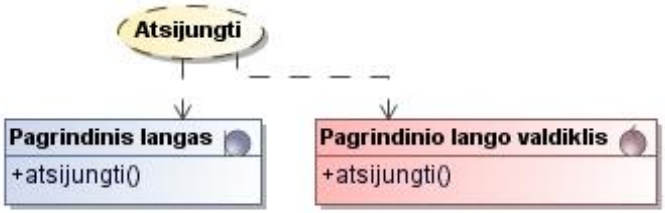
4.21 pav. PA „Prisiregistruoti“ realizacijos klasių diagrama

Panaudojimo atvejo „Prisijungti“ realizacijos klasių diagrama yra vaizduojama 4.22 paveiksle.



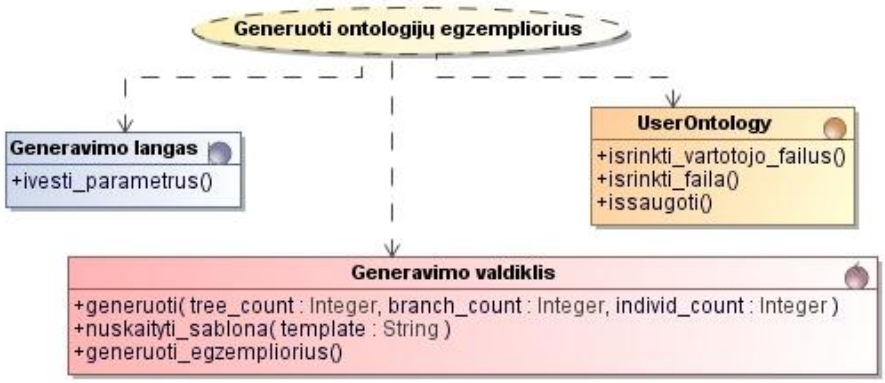
4.22 pav. PA „Prisijungti“ realizacijos klasių diagrama

Panaudojimo atvejo „Atsijungti“ realizacijos klasių diagrama yra vaizduojama 4.23 paveiksle.



4.23 pav. PA „Atsijungti“ realizacijos klasių diagrama

Panaudojimo atvejo „Generuoti ontologijų egzempliorius“ realizacijos klasių diagrama yra vaizduojama 4.24 paveiksle.



4.24 pav. PA „Generuoti ontologijų egzempliorius“ realizacijos klasių diagrama

Panaudojimo atvejo „Peržiūrėti sugeneruotas ontologijas“ realizacijos klasių diagrama yra vaizduojama 4.25 paveiksle.

4.25 pav. PA „Peržiūrėti sugeneruotas ontologijas“ realizacijos klasių diagrama

Valdiklių operacijos ir jų paskirties aprašymai yra pateikiami 4.6 – 4.10 lentelėse.

Registracijos valdiklio operacijų specifikacija pateikta 4.6 lentelėje.

4. 6 lentelė Registracijos valdiklio operacijų specifikacija

Metodas	Paskirtis
tikrinti_duomenu_korektiskuma()	Patikrinti ar suvesti duomenys yra korektiški (pvz.: elektroninis paštas).
generuoti_slaptazodi()	Sugeneruoti vartotojui slaptažodį.
siusti_laiska()	Siųsti laišką naujai prisiregistravusiam vartotojui su jo prisijungimo duomenimis.

Prisijungimo valdiklio operacijų specifikacija pateikta 4.7 lentelėje.

4. 7 lentelė Prisijungimo valdiklio operacijų specifikacija

Metodas	Paskirtis
tikrinti_prisijungimo_duomenis()	Patikrinti ar galimi tokie prisijungimo duomenys.
prijungti()	Jeigu vartotojas egzistuoja, jį prijungti prie sistemos (patikrinant lauką ar administratorius ir pateikiant atitinkamą meniu).

Pagrindinio lango valdiklio operacijų specifikacija pateikta 4.8 lentelėje.

4. 8 lentelė Pagrindinio lango valdiklio operacijų specifikacija

Metodas	Paskirtis
atsijungti()	Atjungti vartotoją nuo generavimo sistemos.

Generavimo valdiklio operacijų specifikacija pateikta 4.9 lentelėje.

4. 9 lentelė Generavimo valdiklio operacijų specifikacija

Metodas	Paskirtis
generuoti (tree_count:Integer, branch_count:Integer, individ_count:Integer)	Sugeneruoti ontologijos failą.
nuskaityti_sablona(template:String)	Nuskaityti pradžios arba pabaigos šabloną.
generuoti_egzempliorius()	Sugeneruoti individus.

Sugeneruotų ontologijų peržiūros valdiklio operacijų specifikacija pateikta 4.10 lentelėje.

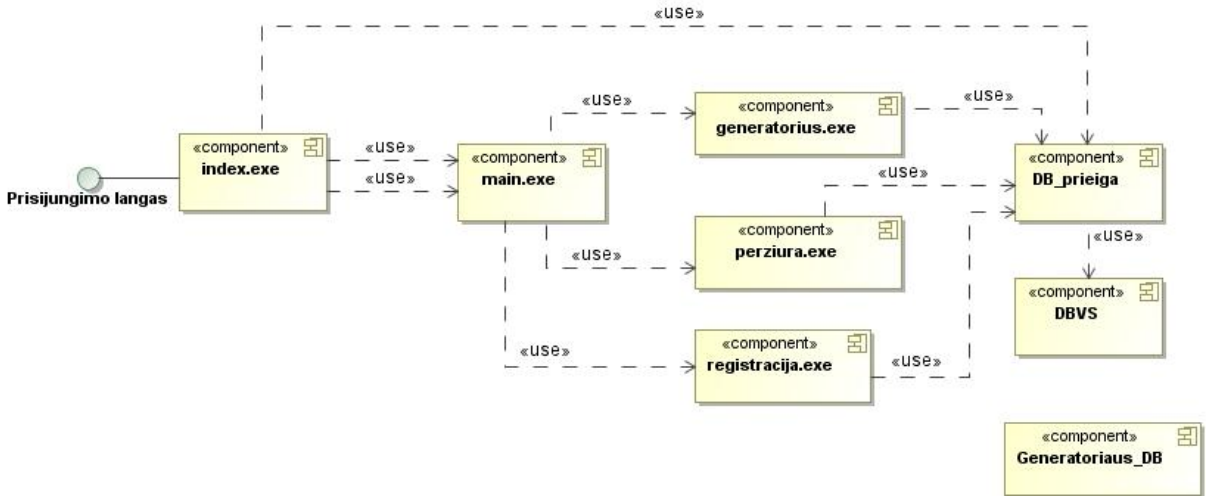
4. 10 lentelė Sugeneruotų ontologijų (failų) peržiūros valdiklio operacijų specifikacija

Metodas	Paskirtis
sukurti_failu_sarasa()	Išrinkti vartotojo ontologijos failus.
pateikti_failo_turini()	Pateikti vartotojui ontologijos failo turinį.
isrinkti_faila()	Išrinkti vartotojo pasirinktą failą parsisiuntimui.
apskaičiuoti_egzemplioriu_sk()	Apskaičiuoti ontologijos sugeneruotų egzempliorių bendrą skaičių.

4.2.6. Realizacijos modelis

Programinių komponentų architektūra

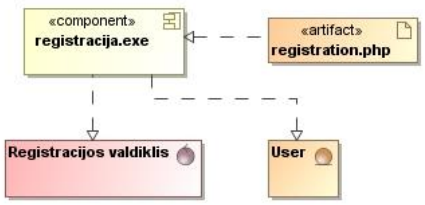
Realizacijos modelį sudaro komponentų, komponentų realizavimo ir diegimo diagramos. Komponentų diagrama pateikta 4.26 paveiksle. Ji parodo iš kokių komponentų susideda ontologijos egzempliorių generavimo įrankis.



4.26 pav. Komponentų diagrama

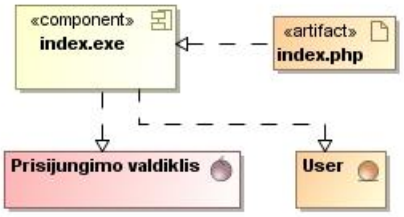
Komponentų realizavimo diagramos parodo iš ko susideda komponentai iš komponentų diagramos.

Komponento registracija.exe realizavimo diagrama pateikta 4.27 paveiksle.



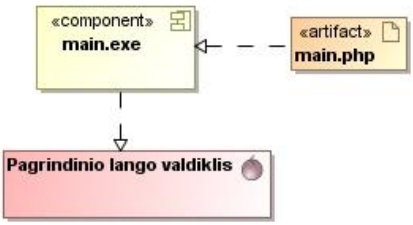
4.27 pav. Komponento registracija.exe realizavimo diagrama

Komponento index.exe realizavimo diagrama pateikta 4.28 paveiksle.



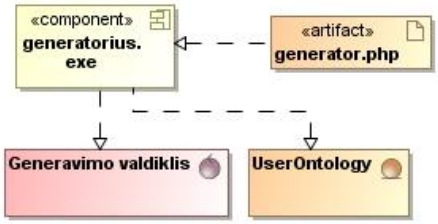
4.28 pav. Komponento index.exe realizavimo diagrama

Komponento main.exe realizavimo diagrama pateikta 4.29 paveiksle.



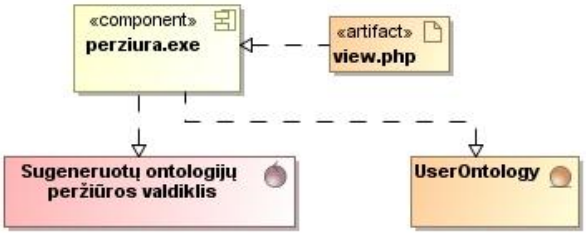
4.29 pav. Komponento main.exe realizavimo diagrama

Komponento generatorius.exe realizavimo diagrama pateikta 4.30 paveiksle.



4. 30 pav. Komponento generatorius.exe realizavimo diagrama

Komponento perziura.exe realizavimo diagrama pateikta 4.31 paveiksle.



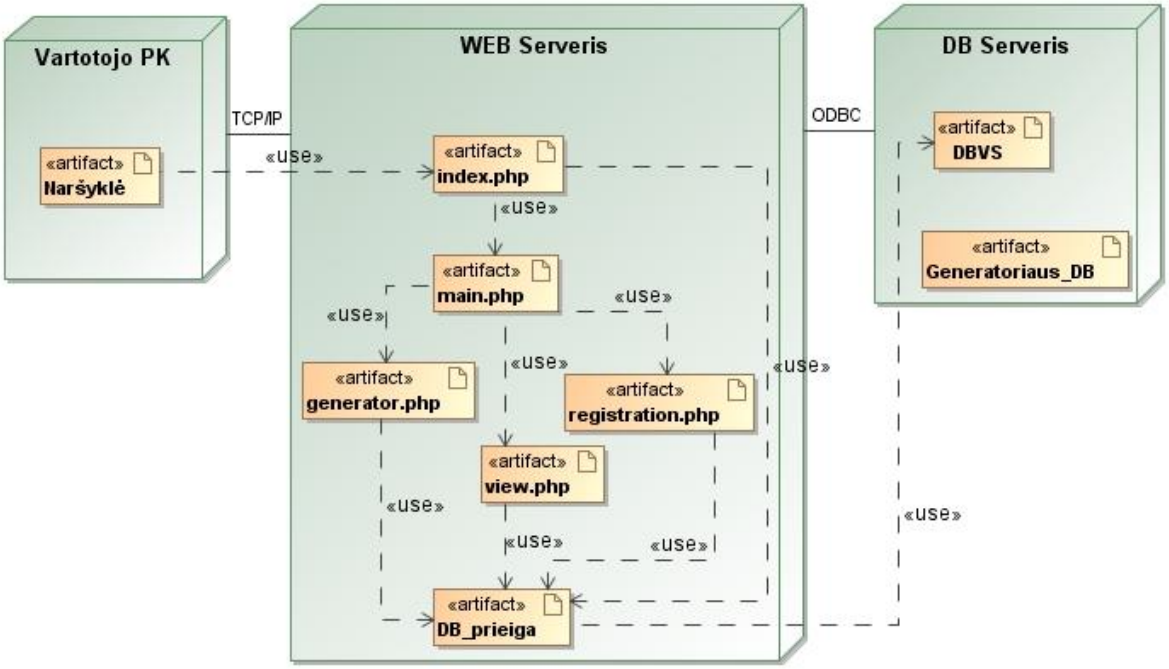
4.31 pav. Komponento perziura.exe realizavimo diagrama

DB prieiga, DBVS ir Generatoriaus_DB komponentų realizavimo diagrama pateikta 4.32 paveiksle.

4.32 pav. Komponentų DB prieiga, DBVS ir Generatoriaus_DB realizavimo diagrama

Diegimo modelis

Ontologijų generavimo įrankio diegimo diagrama pateikta 4.33 paveiksle.



4.33 pav. Diegimo diagrama

4.3. Realizacija

4.3.1. Realizacijos ir veikimo aprašymas

Tam, kad būtų galima naudotis sistema, reikia prisijungti prie sistemos pasinaudojant prisijungimo duomenimis – vartotojo vardu ir slaptažodžiu. Tokiu būdu vartotojai mato tik savo sugeneruotus failus. Taip pat yra suteikiama galimybė prisiregistruoti naujam vartotojui.

Norint susigeneruoti ontologijos su jos egzemplioriais failą, vartotojui reikia suvesti tris kintamuosius – medžių, medžių šakų ir individų šakose skaičius. Šiems skaičiams yra taikomi apribojimai: medžių skaičius turi būti ne mažesnis negu 2, medžių šakų skaičius turi būti ne mažesnis negu 1 ir individų skaičius turi būti ne mažesnis negu 2. Visi šie apribojimai vartotojui yra matomi prieš įvedant parametrus generavimui (5.4 paveikslas), tokiu būdu leidžiant iš karto įvesti teisingas parametrų reikšmes.

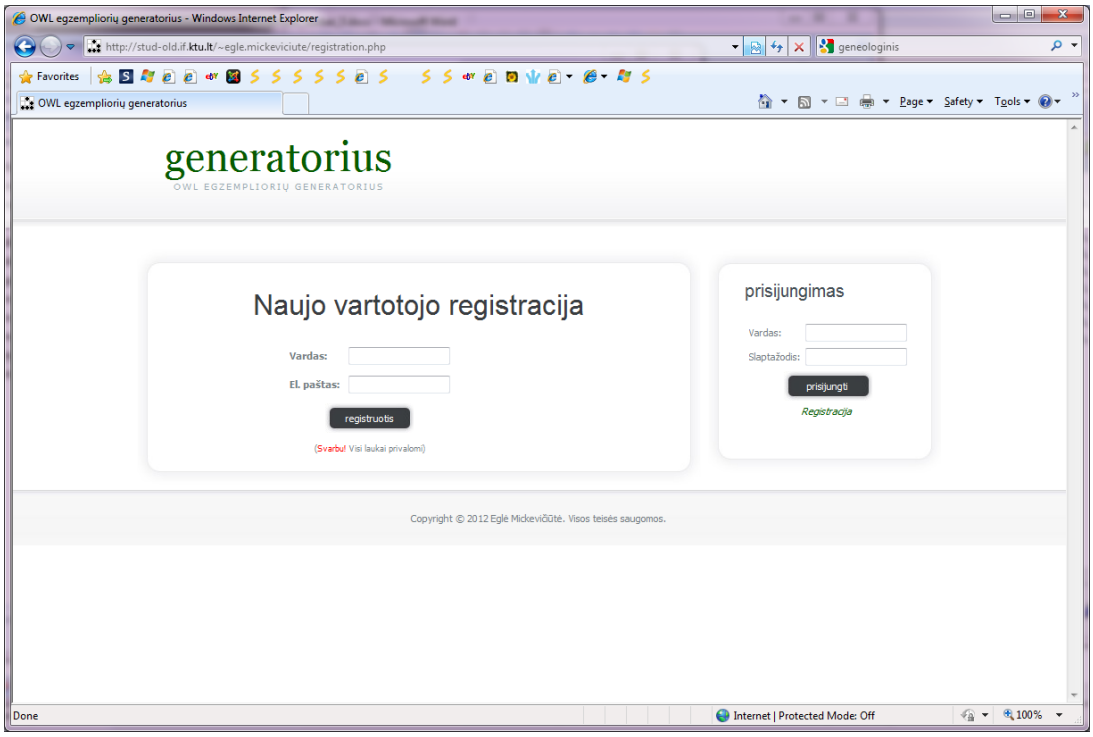
Vartotojui suvedus reikalingus parametrus, sistema sugeneruoja ontologijos su jos egzemplioriais failą ir jį išsaugo. Iš pradžių sistema nuskaito ontologijos (naudojama genealoginio medžio ontologija) antraštę su ontologijos klasėmis ir įrašo į sugeneruoto pavadinimo rezultato failą (jo vardas yra „result_“ + sugeneruotas 6 skaitmenų skaičius_egzempliorių skaičius), vėliau pagal įvestus parametrus sugeneruoja ontologijos individus ir įrašo į rezultato failą, toliau – nuskaito ontologijos poraštę ir įrašo į rezultato failą. Sugeneruoti individų vardai yra sudaromi taip: jei individas yra medžio viršūnė – „Ancestor“ + numeris, kuris nusako, kelinto medžio viršūnėje jis yra; jei individas yra šakos narys, esantis ne pačioje viršūnėje – „Person_name_“ + numeris,

kuriame medyje jis yra + „_“ + numeris, kurioje medžio šakoje jis yra + „_“ + kelintas individas šakoje jis yra. Tokia vardų formuluotė padeda lengvai atskirti ir identifikuoti individus testavimo metu. Testavimo pavyzdys pateiktas 4.3.3 skyriuje.

Sugeneruoti medžiai turi būti sujungti tarpusavyje, todėl neturi likti atskirų medžių. Sujungiant juos tarpusavyje pakeičiama antrojo tėvo lytis, sujungiamas paskutinis individas iš vieno medžio paskutinės šakos su priešpaskutiniu sekančio medžio pirmos šakos individu. Taip daroma todėl, kad nebūtų atsitiktinai sujungti individai iš labai nutolusių kartų.

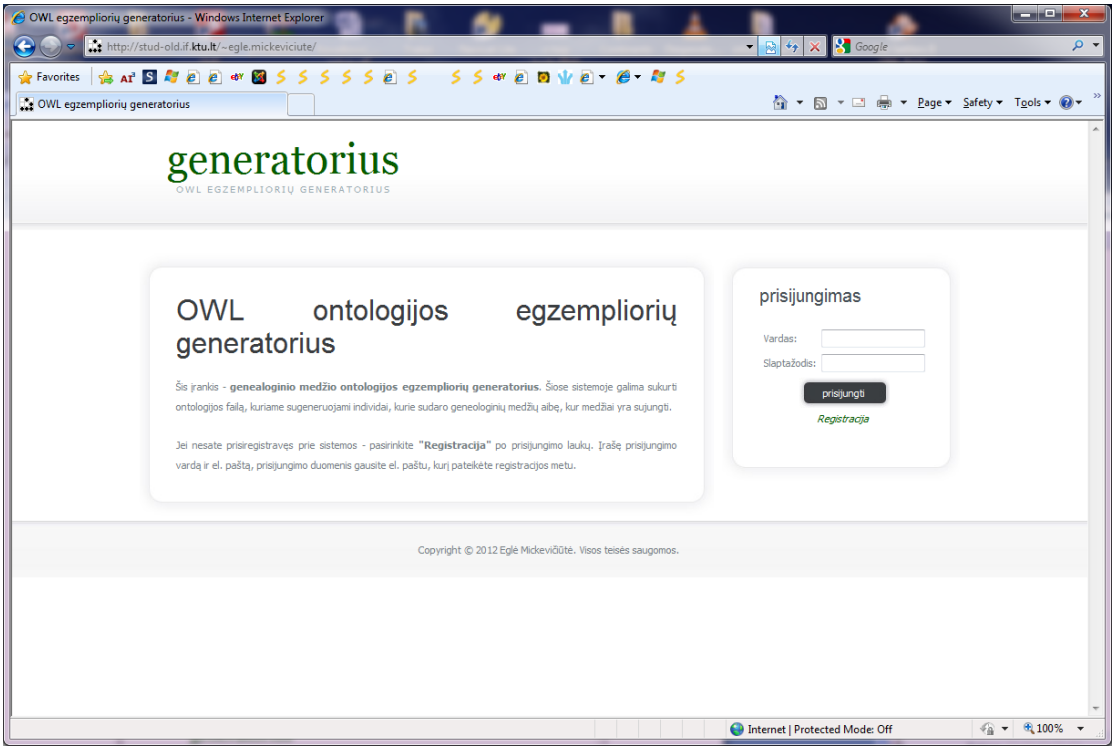
Funkcijas, kurias vartotojas gali atlikti prisijungęs prie sistemos, iliustruoja 4.34 – 4.38 paveikslai.

Vartotojo registracijos langas pateiktas 4.34 paveiksle. Vartotojui reikia pateikti prisijungimo vardą ir savo elektroninį paštą, į kurį patvirtinus registraciją yra išsiunčiamas laiškas su naujo vartotojo prisijungimo duomenimis.



4.34 pav. Registracijos prie sistemos langas

Vartotojo prisijungimo langas pateiktas 4.35 paveiksle. Pirmame, prisijungimo, lange trumpai pateikiama informacija, kam yra skirta sistema. Taip pat pateikiama informacija vartotojams apie registraciją, kurie dar neprisiregistravę prie sistemos.

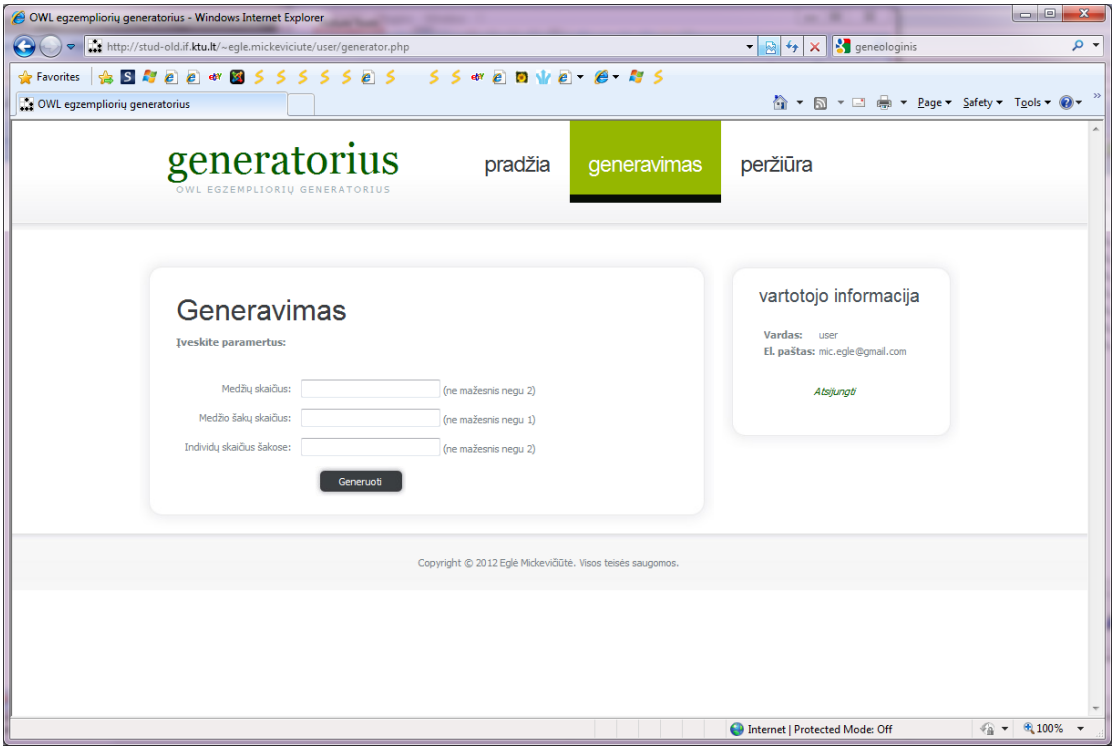


4.35 pav. Prisijungimo prie generavimo sistemos langas

Vartotojui prisijungus prie sistemos yra pateikiamas pradinis langas, kuris pateiktas 4.36 paveiksle. Prisijungęs vartotojas gali susipažinti su įrankiu bei jo funkcijomis bei atsisiųsti užklausų rinkinį. Dešinėje pusėje pateikiama trumpa informacija apie sistemą ir prisijungusio vartotojo duomenys.

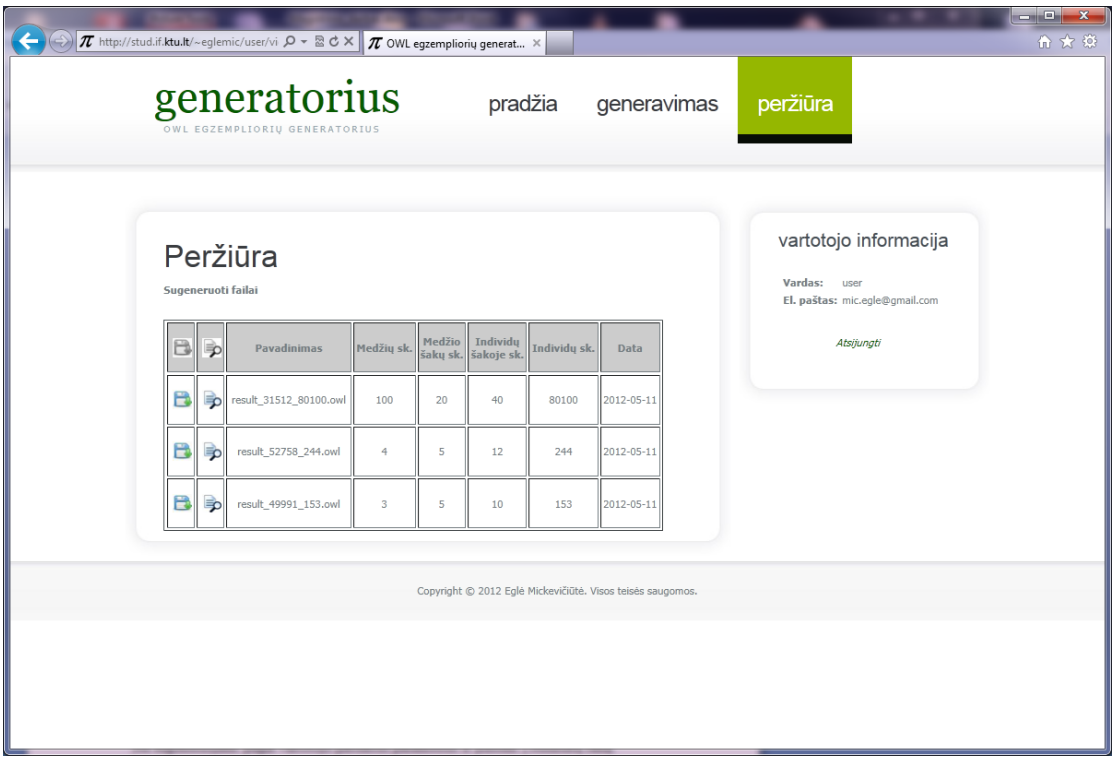
4.36 pav. Pagrindinis generatoriaus langas

Vartotojui pasirinkus „Generavimas“ iš kairėje esančio meniu yra patenkama į ontologijos su jos egzemplioriais generavimo langą, kuriame reikia suvesti medžių, medžių šakų ir individų šakose skaičius. Generavimo langas pateiktas 4.37 paveiksle.



4.37 pav. Generavimo langas

Vartotojui pasirinkus „Peržiūra“ iš kairėje esančio meniu yra patenkama į sugeneruotų failų peržiūrą. Peržiūros langas pateiktas 4.38 paveiksle. Vartotojas gali matyti visus savo ontologijos su jos egzemplioriais failus sukurtus pagal vartotojo įvestus kriterijus, kurie taip pat pateikiami peržiūroje. Papildomai yra paskaičiuojamas kiekvienos ontologijos sugeneruotų egzempliorių bendras skaičius.



4.38 pav. Sugeneruotų failų peržiūros langas

4.3.2. Testavimo modelis

Kadangi sistemos pagrindinė funkcija yra ontologijos su jos egzemplioriais generavimas, svarbu ištestuoti ar teisingai yra sugeneruojami ontologijos individai. Pasirinkta ontologija – genealoginis medis, kurios antraštę, klases ir poraštę generavimo algoritmas nuskaito ir įrašo į rezultato failą generavimo metu. *SPARQL* užklausos bus atliekamos tarp ontologijos individų, kurie yra sugeneruojami pagal vartotojo pateiktus parametrus ir įrašomi į rezultatų failą.

Generavimo algoritmo testavimas susideda iš 3 dalių:

1. pradinių reikšmių parinkimo;
2. failo peržiūros, kuris yra sugeneruojamas;
3. individų, kurie yra sugeneruojami, medžio nubraižymu.

4.3.3. Testavimo duomenys ir rezultatai

Pradinių reikšmių parinkimas

Testavimui parenkame tokias reikšmes:

Medžių skaičius: 3

Šakų skaičius: 2

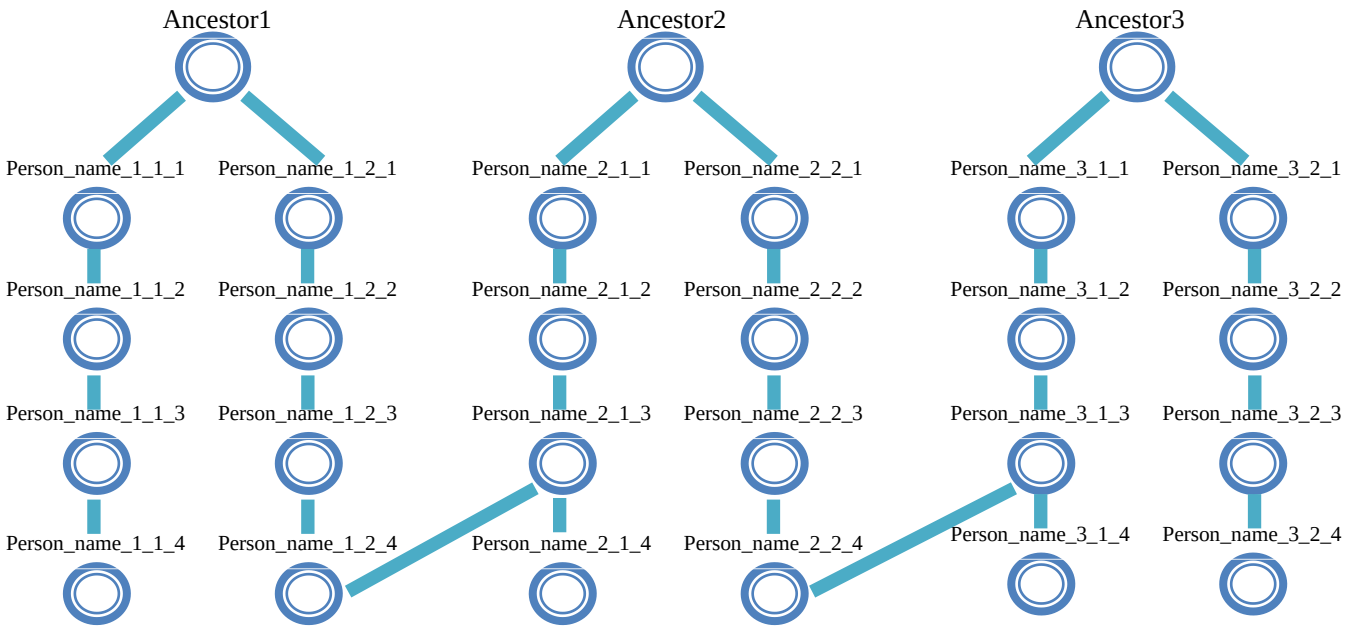
Individų skaičius šakoje: 4

Sugeneruoto failo turinys peržiūrai:

Kadangi failas labai ilgas, pateikiami tik sugeneruoti individai 2 šio darbo priede.

Individų medis:

Pagal pateiktus sugeneruotus individus buvo sudarytas medis, kuris pateiktas 4.39 paveiksle.



4.39 pav. Sugeneruotų ontologijos egzempliorių genealoginių medžių miškas

5. Eksperimentinis sistemos tyrimas

5.1. Eksperimento planas

Apibrėžimas. Išanalizuoti *SPARQL* užklausų kalbos naujas galimybes, didžiausią dėmesį skiriant rekursinėms užklausoms, siekiant ištirti šių užklausų galimybes, vykdymo laiką susidorojant su egzempliorių gausa, užklausų ypatumais ir ontologijos charakteristikomis pasitelkiant šiame darbe sudarytu užklausų rinkiniu ir suprojektuotu ir realizuotu ontologijos egzempliorių generavimo įrankiu, kuriuo sugeneruojami genealoginio medžio *OWL 2* ontologijos failai su skirtingu egzempliorių skaičiumi. Užklausoms realizuoti pasirinktas *Eclipse* įrankis su *Pellet OWL Reasoner* išvedimo ir *SPARQL* vykdymo mechanizmu.

Planavimas. Eksperimentas vykdomas pasinaudojus šiame darbe sukurtu ontologijos egzempliorių generavimo įrankiu, sugeneruojami genealoginio medžio ontologijos su skirtingu egzempliorių skaičiumi, jose vykdomos *SPARQL* užklausų rinkinio užklausos, įvertinamas šių užklausų vykdymo laikas bei ontologijos užkrovimo laikas. Eksperimentas kartojamas su skirtingus parametrus turinčia kompiuterine įranga, atsižvelgiant į operatyviąją atmintį.

Kintamųjų pasirinkimas – pasirenkami patogiausi subjektai iš įvairių populiacijos elementų. Eksperimentas susideda iš dviejų dalių: tiriamas ontologijos užkrovimo laikas ir užklausų vykdymo laikas su mažesniu egzempliorių skaičių turinčiomis sugeneruotomis ontologijomis (šiuo atveju tiriamos užklausos, kuriose naudojamos *SWRL* taisyklės) ir ontologijos užkrovimo ir užklausų vykdymo laiko su didesniu egzempliorių skaičių turinčiomis sugeneruotomis ontologijomis (be užklausų, kuriose naudojamos *SWRL* taisyklės). Primoje eksperimento dalyje pasirinkti kintamieji pateikti 5.1 lentelėje.

5.1 lentelė Kintamųjų reikšmės pirmoje eksperimento dalyje

Nr.	Medžių skaičius	Medžio šakų skaičius	Individų šakoje skaičius	Bendras egzempliorių skaičius
1.	2	3	4	26
2.	3	4	5	63
3.	4	5	5	104
4.	5	5	6	155
5.	5	6	6	185

Antroje eksperimento dalyje pasirinkti kintamieji pateikti 5.2 paveiksle.

5.2 lentelė Kintamųjų reikšmės antroje eksperimento dalyje

Nr.	Medžių skaičius	Medžio šakų skaičius	Individų šakoje skaičius	Bendras egzempliorių skaičius
1.	5	6	7	215
2.	5	9	9	410
3.	8	10	10	808
4.	9	11	13	1296
5.	10	12	13	1570
6.	10	15	15	2034

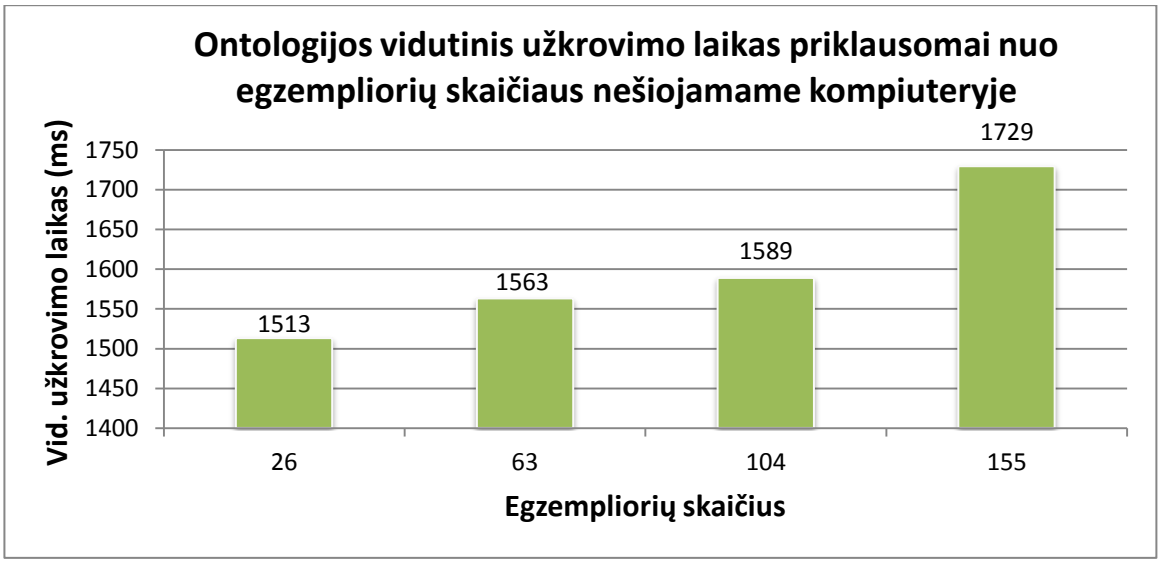
Kiekviena užklausa vykdoma 10 kartų, rezultatas imamas šių matavimų vidurkis (kaip buvo vykdomos *SPARQL* užklausos *LUBM* etalone [8]). Ontologijos užkrovimas taip pat matuojamas tai pačiai ontologijai 10 kartų ir imamas matavimų vidurkis, tokiu būdų stengiantis kuo tiksliau išmatuoti šias vykdymo charakteristikas.

Eksperimentas vykdomas dvejuose skirtingus parametrus turinčiuose kompiuteriuose: nešiojamame kompiuteryje (charakteristikos: 4 GB RAM, Intel(R) Core(TM) i3 CPU M330 @ 2.13 GHz) ir serveryje (charakteristikos: 6 GB RAM, Intel(R) Xeon(R) CPU X5570 @ 2.93 GHz). Nešiojamame kompiuteryje atliktas abiejų dalių eksperimentas, serveryje – tik antros dalies, norint sužinoti ar atminties padidinimas išsprendžia tam tikrų užklausų vykdymo problemas (kai trūksta atminties) ir išmatuoti šių užklausų vykdymo laiką. Nešiojamame kompiuteryje skirtas atminties kiekis *Eclipse* įrankiui – 1300 MB, serveryje – 4096 MB.

5.2. Eksperimento rezultatas

Atlikus eksperimentą rezultatų duomenys pateikti grafiniu būdu – grafikuose. Eksperimentas, atliktas nešiojamame kompiuteryje, grafiniu būdu pateiktas 5.1 – 5.12 paveiksluose, o serveryje – 5.13 – 5.18 paveiksluose.

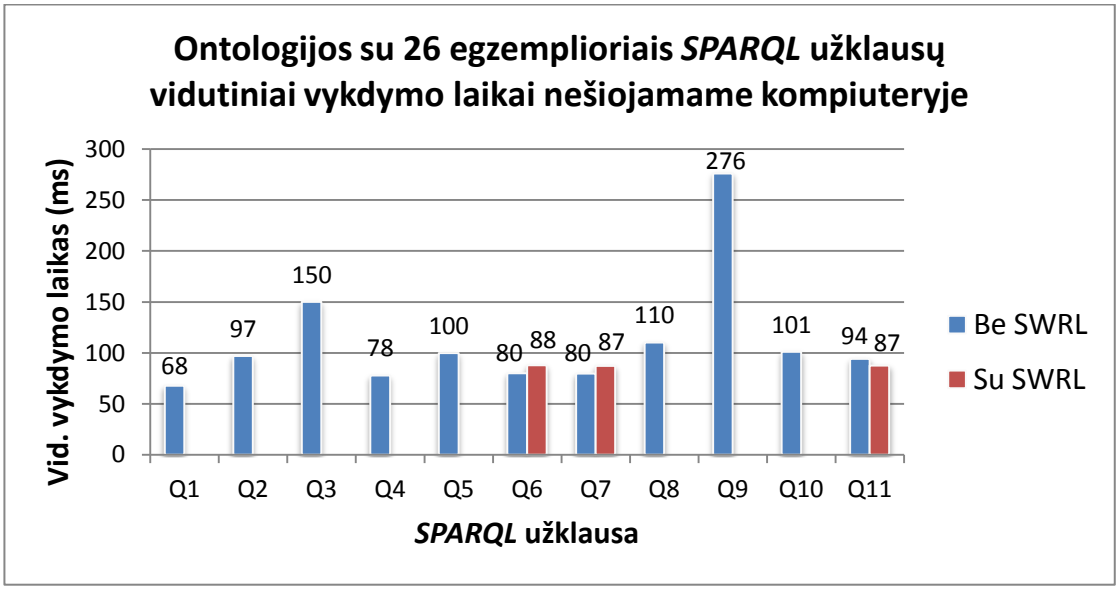
Ontologijos vidutinio užkrovimo laikas priklausomai nuo egzempliorių skaičiaus nešiojamame kompiuteryje su *AllDifferent* savybe pateiktas 5.1 paveiksle. Kuo didesnis egzempliorių skaičius ontologijoje – tuo vidutinis užkrovimo laikas yra didesnis. Ta pati tendencija matoma ir matuojant vidutinį užrovimo laiką ontologijai be *AllDifferent* savybės, kuris pateiktas grafike 5.7 paveiksle.



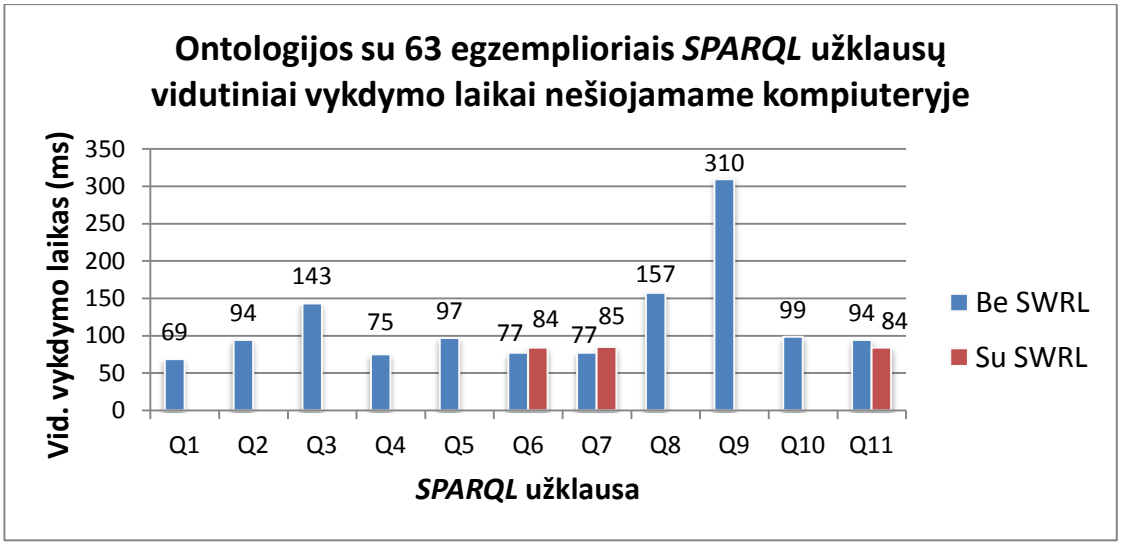
5.1 pav. Ontologijos vidutinio užkrovimo laikas priklausomai nuo egzempliorių skaičius nešiojamame kompiuteryje su *AllDifferent* savybe

Ilgiausiai iš visų užklausų buvo vykdomos Q3, Q8 ir Q9 užklausos. Q3 užklausoje naudojamos vidinės užklausos, agregavimo funkcijos `COUNT` ir rekursijos, Q8 – agregavimo

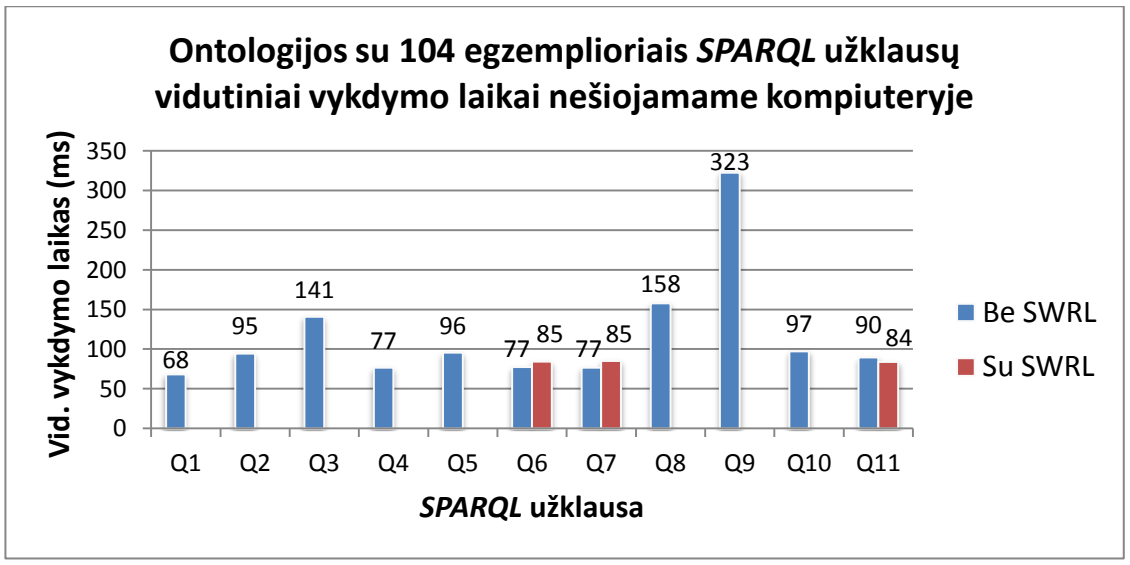
funkcija `COUNT` su `GROUP BY` ir `HAVING`, Q9 – kaip ir Q8 užklausoje naudoti užklausos elementai bei rekursija, pastaroji užklausa vyko ilgiausiai iš visų kitų užklausų. Visos užklausos pateiktos 3.3 skyriuje 3.2 lentelėje. Tai pastebima ir vykdant užklausas su vis didesniu egzempliorių skaičiumi ontologijoje (5.2 – 5.5 paveikslai). Užklausos Q6 ir Q7 parodo, kad naudojant *SWRL* taisykles užklausos vyksta ilgiau, tačiau Q11 rodo priešingai. Taip yra todėl, kad Q6 ir Q7 užklausose naudojamos nurodomo dydžio rekursijos, todėl nėra tarpinių rezultatų, kaip Q11 užklausoje be rekursijos.



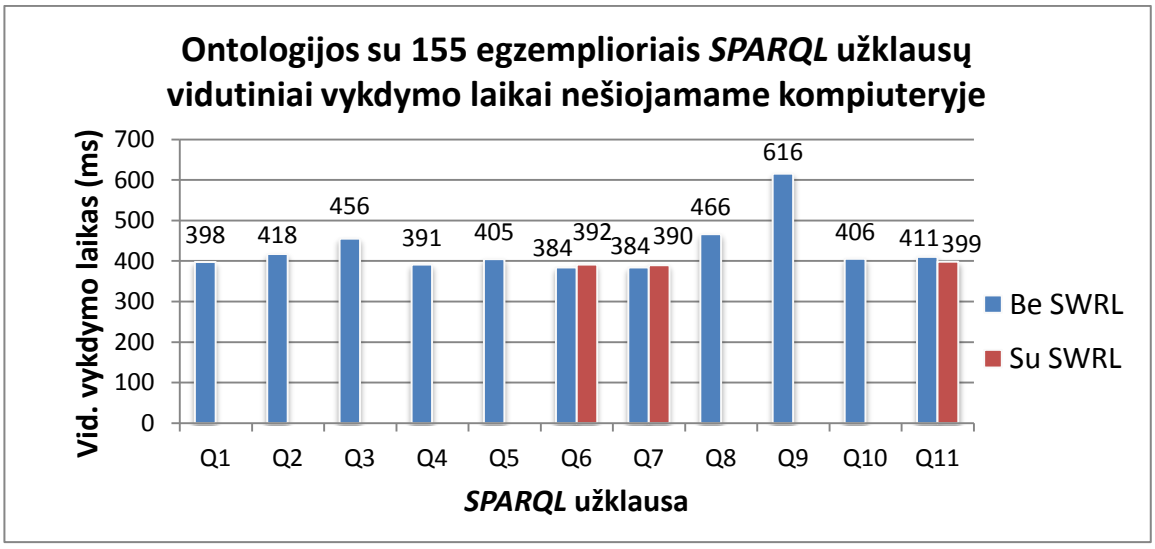
5.2 pav. Ontologijos su 26 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai nešiojamame kompiuteryje su *AllDifferent* savybe



5.3 pav. Ontologijos su 63 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai nešiojamame kompiuteryje su *AllDifferent* savybe

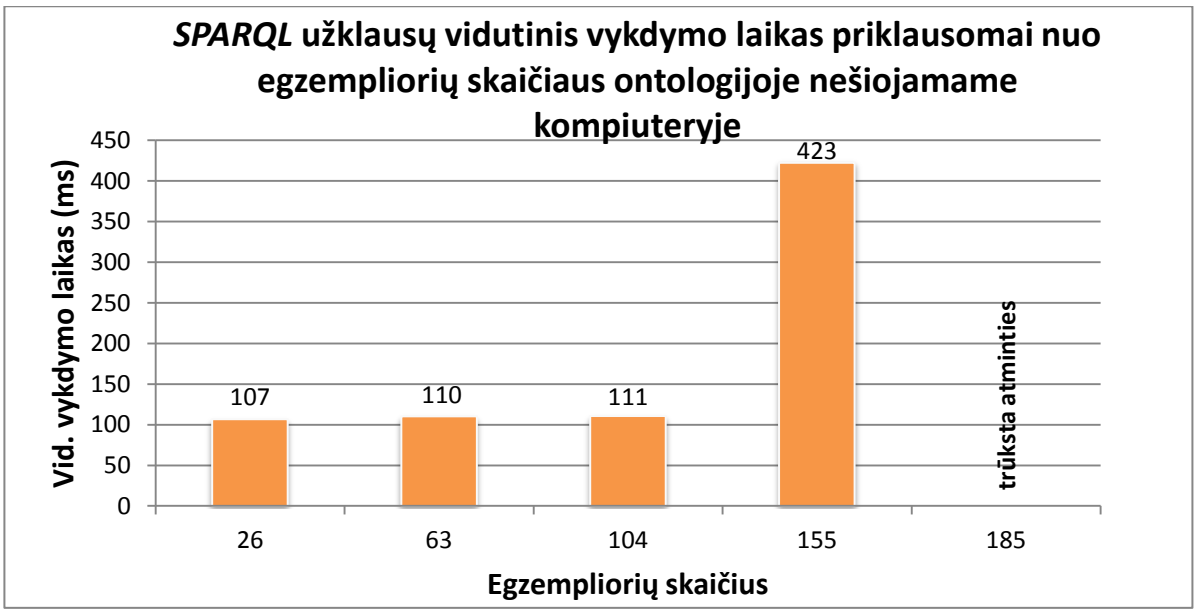


5.4 pav. Ontologijos su 104 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai nešiojamame kompiuteryje su *AllDifferent* savybe



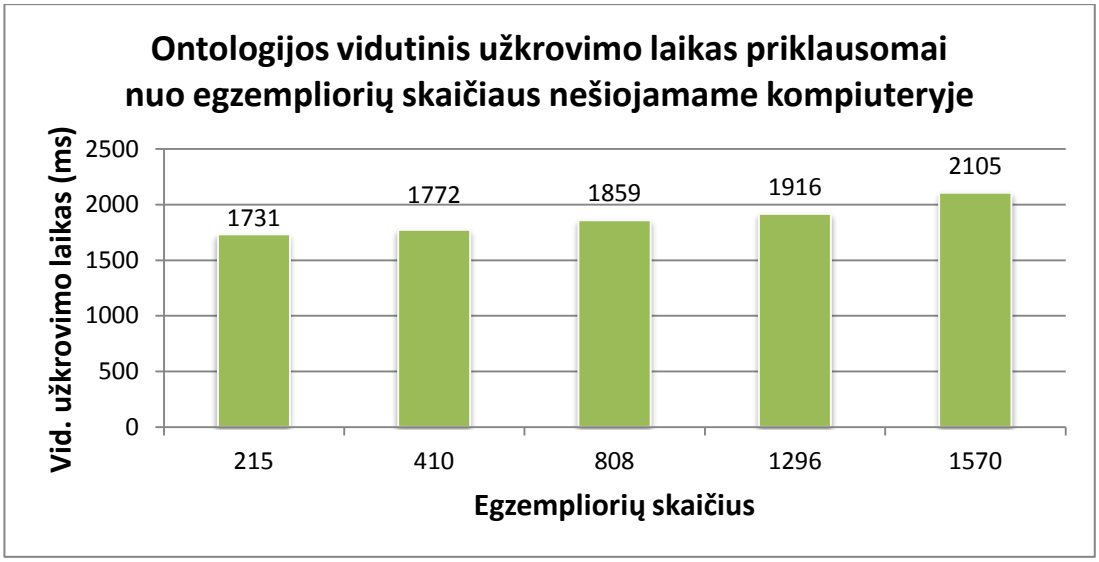
5.5 pav. Ontologijos su 155 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai nešiojamame kompiuteryje su *AllDifferent* savybe

SPARQL užklausų vidutinis vykdymo laikas priklausomai nuo egzempliorių skaičiaus ontologijoje nešiojamame kompiuteryje su *AllDifferent* savybe pateiktas 5.6 paveiksle. Pateiktas grafikas parodo, kad užklausų vidutinis vykdymo laikas ženkliai išauga vykdant užklausas su 155 egzemplioriais turinčia ontologija, tačiau užklausų vykdymas ontologijoje su 185 egzemplioriais yra nesėkmingas, nes gaunamas klaidos pranešimas dėl atminties trūkumo ir užklausos nėra įvykdomos.



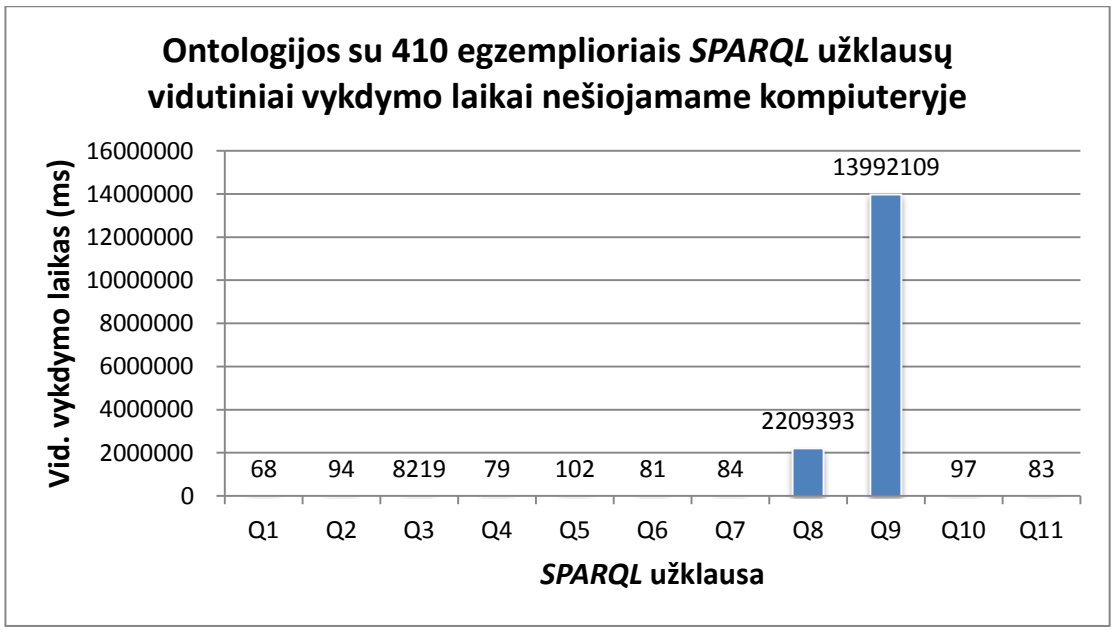
5.6 pav. Vidutinis užklausų vykdymo laikas priklausomai nuo egzempliorių skaičiaus ontologijoje nešiojamame kompiuteryje su *AllDifferent* savybe

Ontologijos užkrovimo vidutinis laikas priklausomai nuo egzempliorių skaičiaus nešiojamame kompiuteryje be *AllDifferent* savybės pateiktas 5.7 paveiksle. Šiuo atveju ontologija neturi savybės *AllDifferent,* kurios reikia *SWRL* taisyklėms, kad būtų galima pažymėti individus, kaip visiškai atskirus egzempliorius.

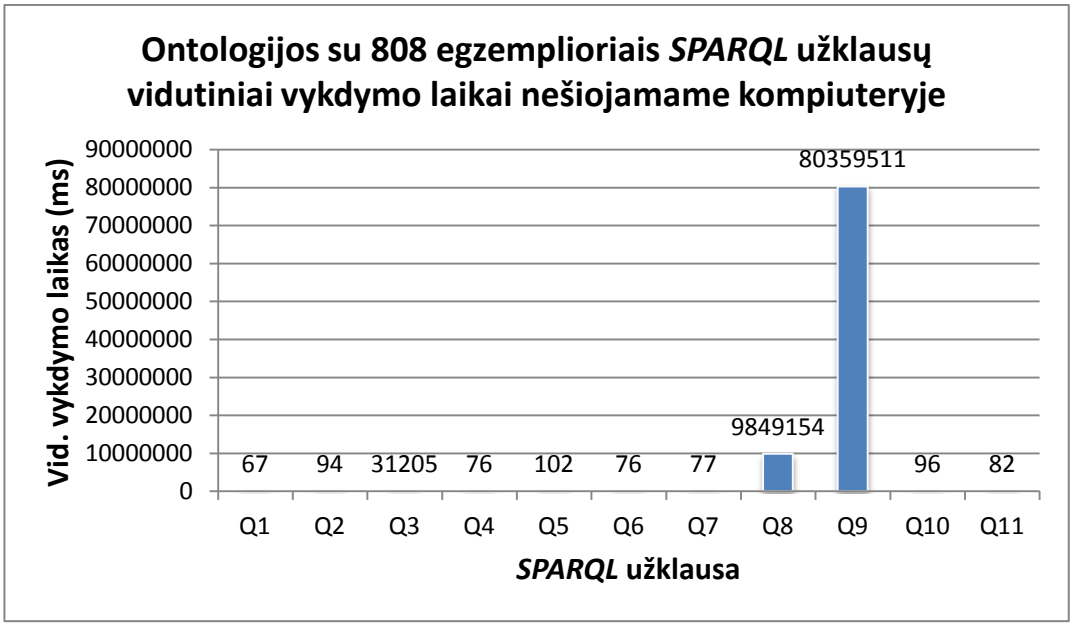


5.7 pav. Ontologijos vidutinis užkrovimo laikas priklausomai nuo egzempliorių skaičius nešiojamame kompiuteryje be *AllDifferent* savybės

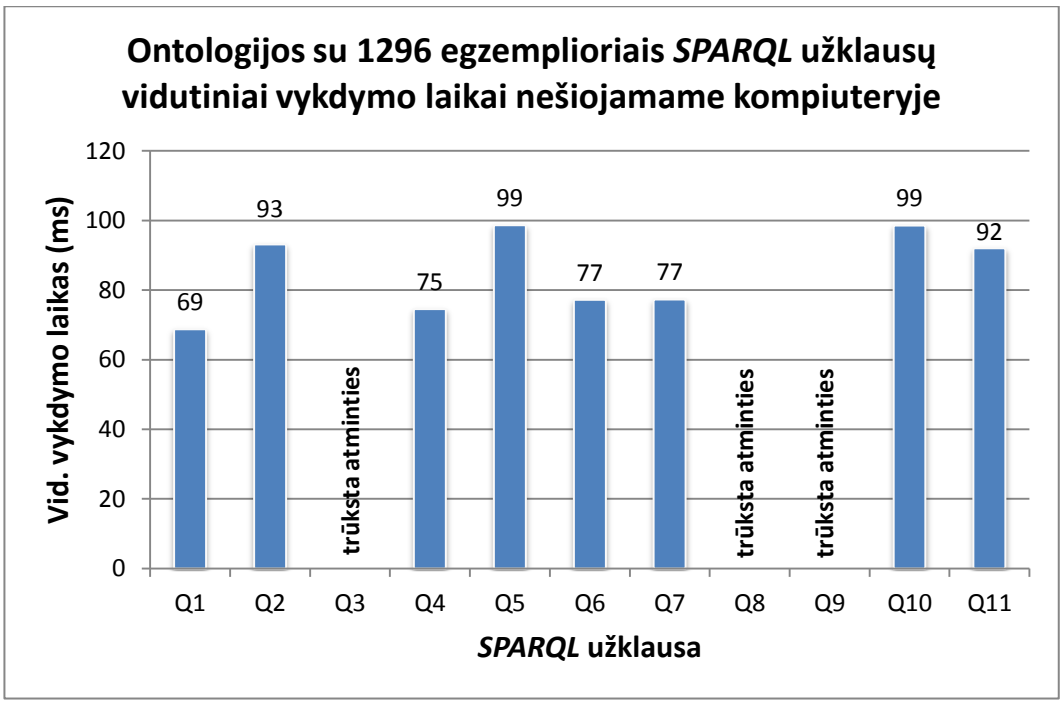
Ilgiausiai iš visų užklausų buvo vykdomos Q3, Q8 ir Q9 užklausos, kaip ir su *AllDifferent* savybe atveju. Ši tendencija matoma vykdant užklausas su skirtingu egzempliorių skaičiumi ontologijose, kurių vykdymo grafikai pateikti 5.8 ir 5.9 paveiksluose. Tačiau 5.10 ir 5.11 paveiksluose matoma, kad tik šioms trims užklausoms iš visų vykdytų užklausų neužtenka atminties iki galo įvykdyti užklausas.



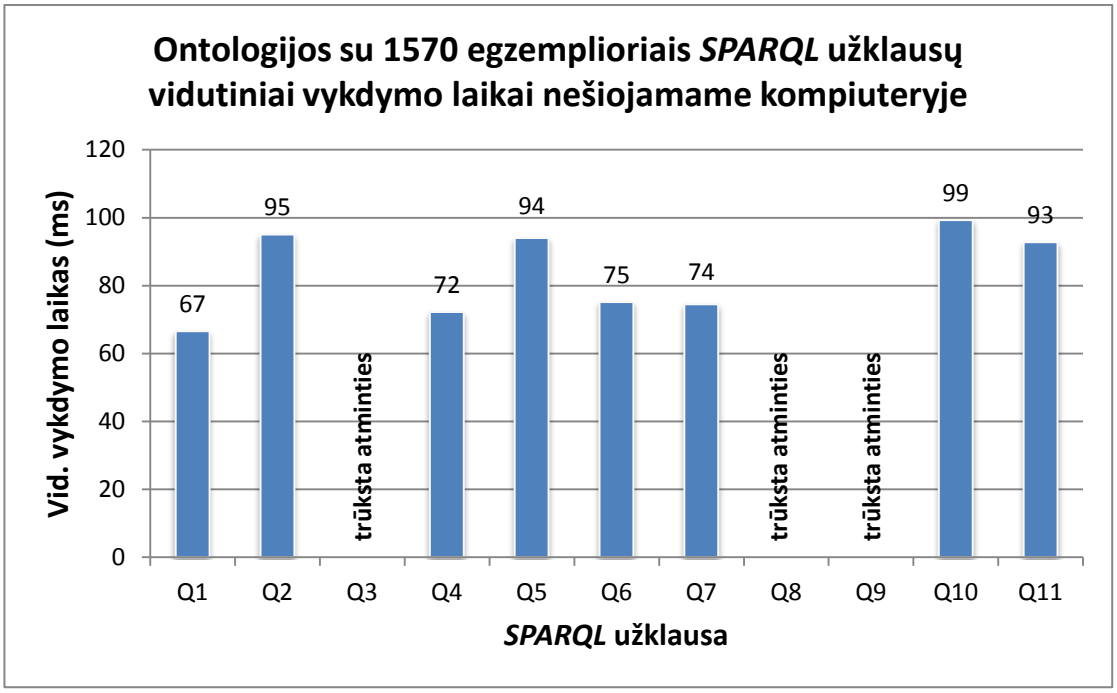
5.8 pav. Ontologijos su 410 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai nešiojamame kompiuteryje be *AllDifferent* savybės



5.9 pav. Ontologijos su 808 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai nešiojamame kompiuteryje be *AllDifferent* savybės



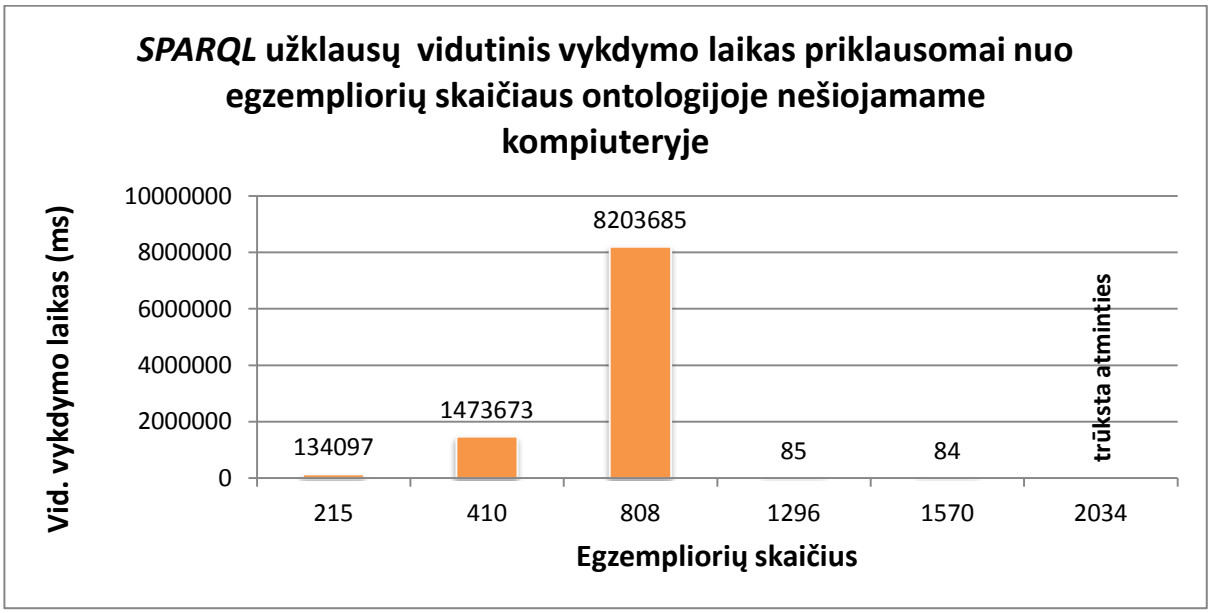
5.10 pav. Ontologijos su 1296 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai nešiojamame kompiuteryje be *AllDifferent* savybės



5.11 pav. Ontologijos su 1570 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai nešiojamame kompiuteryje be *AllDifferent* savybės

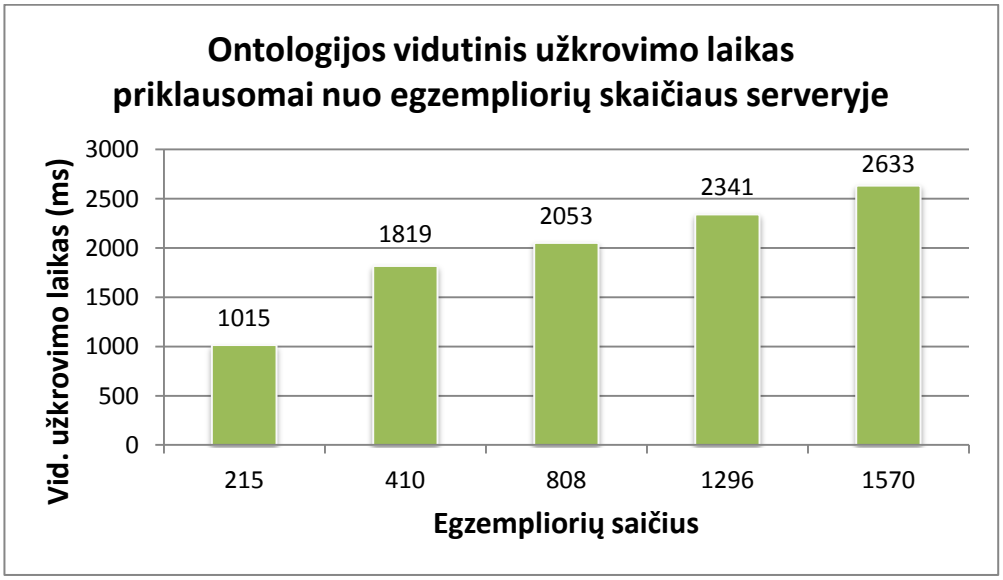
SPARQL užklausų vidutinis vykdymo laikas priklausomai nuo egzempliorių skaičiaus ontologijoje nešiojamame kompiuteryje be *AllDifferent* savybės pateiktas 5.12 paveiksle. Pateiktame grafike matoma, kad vykdant užklausas su skirtingu egzempliorių skaičiumi jau su 2034 egzempliorius turinčia ontologija vykdymo įrankis nepajėgia vykdyti užklausų dėl atminties trūkumo. Ontologijos su 1296 ir 1570 egzemplioriais vidutinis vykdymo laikas yra toks mažas

palyginti su 808 egzempliorius turinčia ontologija dėl to, kad nėra įvertintos Q3, Q8 ir Q9 užklausos, kurių įrankis nesugebėjo įvykdyti, todėl jų vykdymo laikai turi skaitinę reikšmę 0.



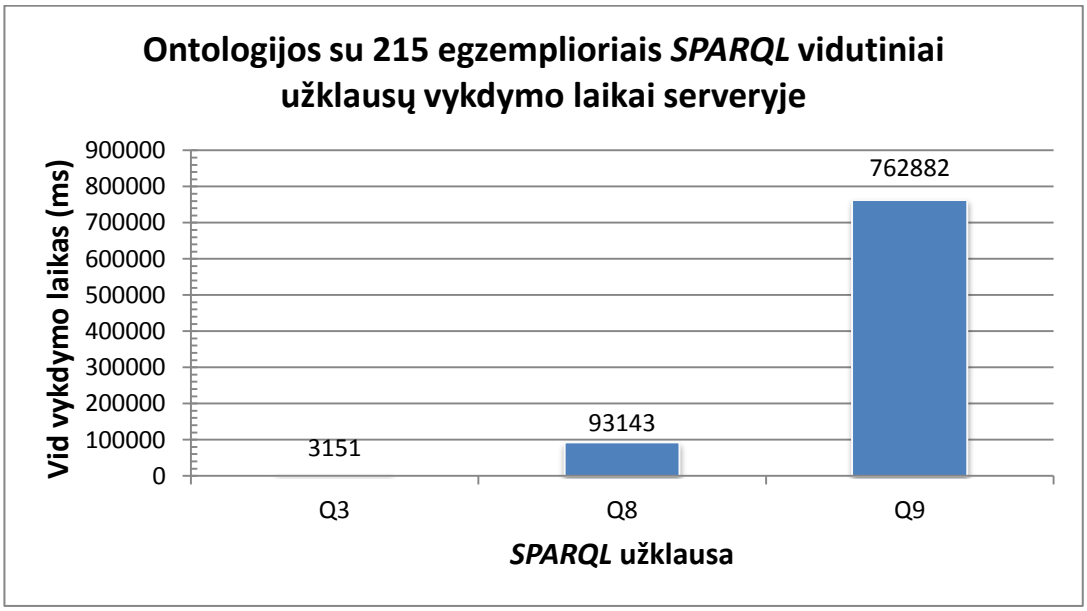
5.12 pav. SPARQL užklausų vidutinis vykdymo laikas priklausomai nuo egzempliorių skaičiaus ontologijoje nešiojamame kompiuteryje be *AllDifferent* savybės

Ontologijos vidutinis užkrovimo laikas priklausomai nuo egzempliorių skaičiaus nešiojamame kompiuteryje be *AllDifferent* savybės pateiktas 5.13 paveiksle. Eksperimento antra dalis pakartota serveryje, kur *Eclipse* įrankiui buvo suteikta daugiau atminties. Buvo tiriamos tik tos užklausos, kurių vykdymo nepavyko gauti tiriant su nešiojamuoju kompiuteriu. Ontologijos užkrovimo vidutinis laikas parodo, kad ontologija su 215 ir 410 egzemplioriais užkraunami greičiau, likę – lėčiau, tačiau skirtumas nėra didelis, todėl galima daryti išvadą, kad atminties padidinimas neturi didelės reikšmės ontologijos užkrovimui į atmintį laiko atžvilgiu.

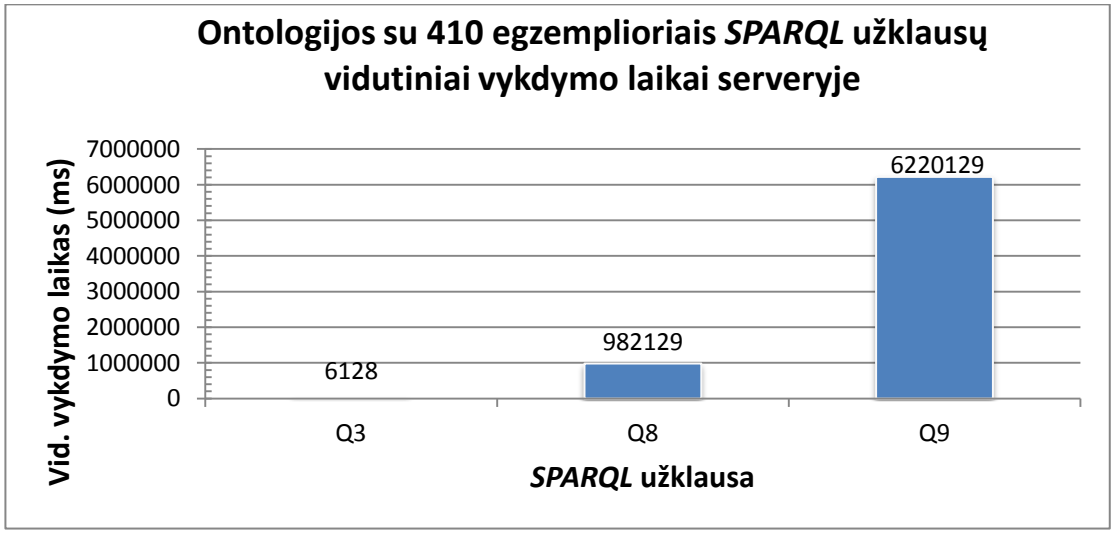


5.13 pav. Ontologijos vidutinis užkrovimo laikas priklausomai nuo egzempliorių skaičius serveryje be *AllDifferent* savybės

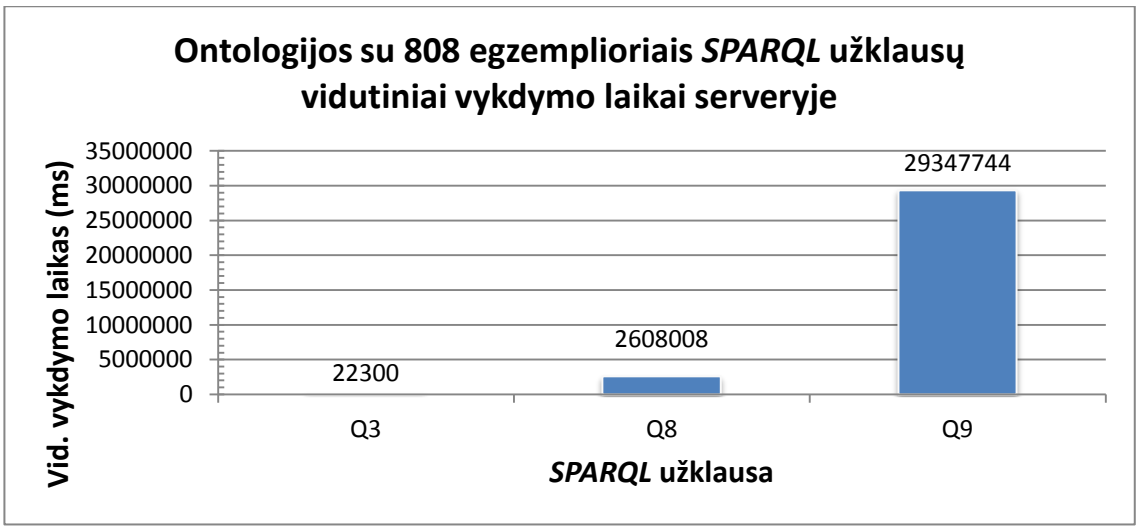
Kuo didesnis egzempliorių skaičius ontologijoje – tuo užklausos yra atliekamos ilgiau. Visi vykdymo rezultatai su skirtingu egzempliorių skaičiumi pateikti 5.14 – 5.18 paveiksluose.



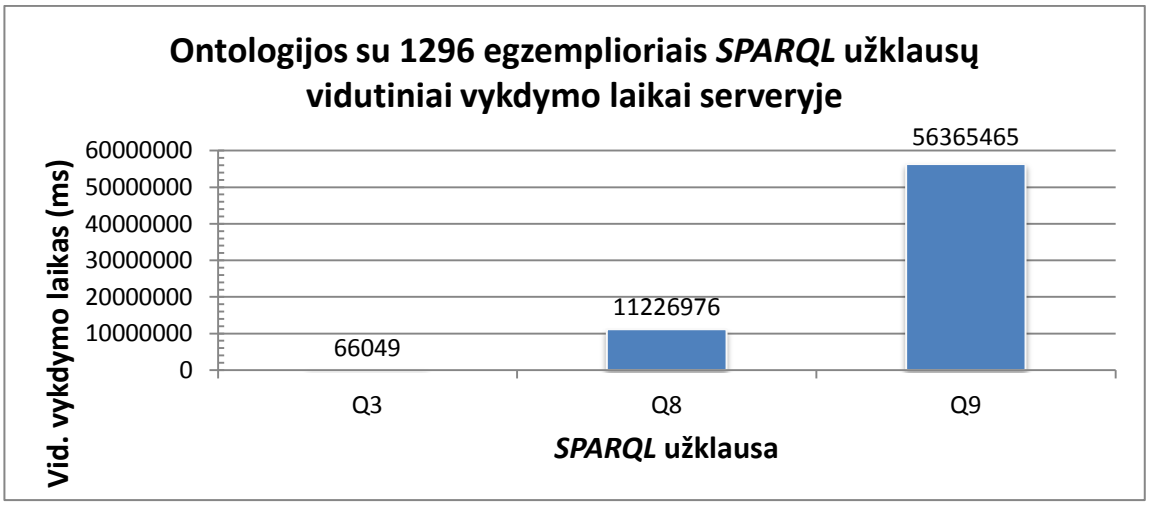
5.14 pav. Ontologijos su 215 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai serveryje be *AllDifferent* savybės



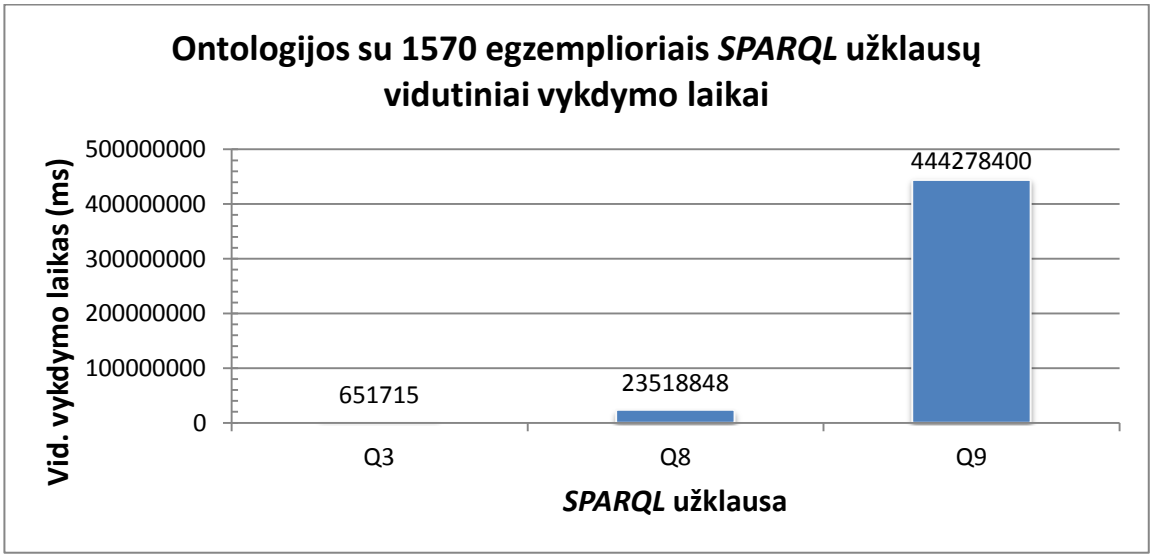
5.15 pav. Ontologijos su 410 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai serveryje be *AllDifferent* savybės



5.16 pav. Ontologijos su 808 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai serveryje be *AllDifferent* savybės



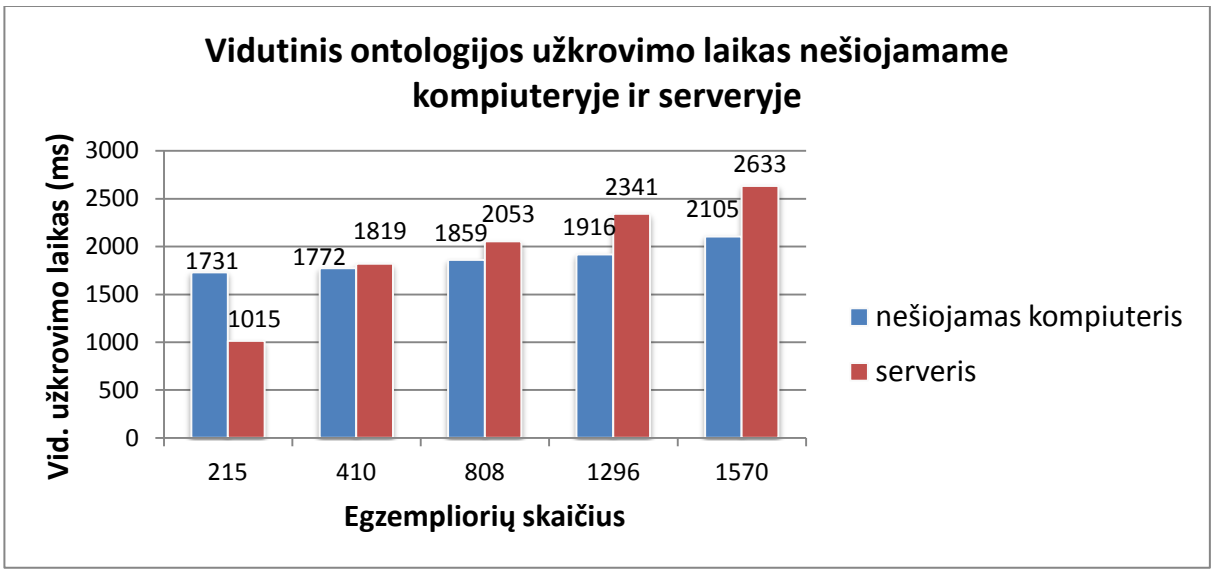
5.17 pav. Ontologijos su 1296 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai serveryje be *AllDifferent* savybės



5.18 pav. Ontologijos su 1570 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai serveryje be *AllDifferent* savybės

5.3. Sistemos veikimo ir savybių analizė, kokybės kriterijų įvertinimas

Atliekant eksperimentą buvo matuojamas ontologijos užkrovimo laikas – ontologijos išvedimas ir įkėlimas į atmintį, vidutinis ontologijos užkrovimo laikas nešiojamame kompiuteryje ir serveryje pateiktas 5.19 paveiksle. Didėjant egzempliorių skaičiui ontologijoje – didėja ir vidutinis ontologijos užkrovimo laikas. Ontologijos vidutinis užkrovimo laikas serveryje, kur *Eclipse* įrankiui buvo suteikta daugiau atminties, buvo vykdomas per ilgesnį laiką (išskyrus pirmą atvejį) negu nešiojamame kompiuteryje, tačiau skirtumas nėra didelis. Vidutinis užkrovimo laikas gali būti didesnis dėl atliekamo ontologijos išvedimo ir užkrovimo į atmintį, po kurio yra vykdomos užklausos, kadangi serveryje vykdytos Q3, Q8 ir Q9 užklausos buvo įvykdytos, tačiau nešiojamame kompiuteryje šioms užklausoms vykdyti neužteko atminties.

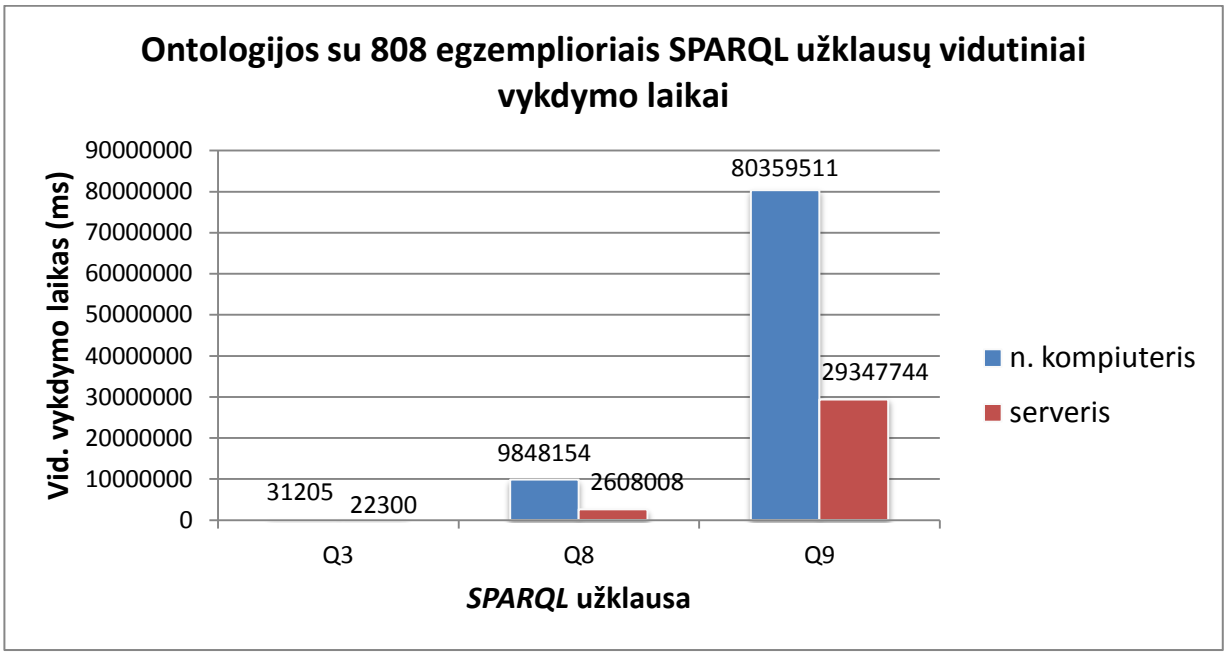


5.19 pav. Vidutinis ontologijos užkrovimo laikas nešiojamame kompiuteryje ir serveryje

Atliekant eksperimentinį užklausų vykdymo tyrimą buvo nustatyta, kad užklausų vykdymas taikant *Pellet* įrankį priklauso nuo:

1. **Egzempliorių skaičiaus ontologijoje.** Kuo didesnis egzempliorių skaičius ontologijoje – tuo ilgesnis yra užklausos vidutinis vykdymo laikas. Pirmoje eksperimento dalyje užklausų vidutinis vykdymo laikas smarkiai išauga ontologijoje su 155 egzemplioriais lyginant su ontologija, kuri turi 104 egzempliorius. Antroje eksperimento dalyje užklausos vykdomos su 1570 egzempliorius turinčia ontologija, nors kai kurioms užklausoms vykdyti neužtenka atminties.
2. **Ontologijos savybės *AllDifferent*.** Taikant šią savybę buvo tiriamas užklausų vykdymas su *SWRL* taisyklėmis, kur *AllDifferent* savybė reikalinga tam, kad būtų galima identifikuoti egzempliorius kaip atskirus individus. Užklausų tyrimas su *SWRL* taisyklėmis buvo atliekamas pirmoje eksperimento dalyje su ontologija turinčia iki 155 egzempliorių, o antroje eksperimento dalyje be *SWRL* taisyklių su ontologija, turinčia iki 1570 egzempliorių.

3. Skiriamos operatyviosios atminties *Eclipse* įrankiui. Antroje eksperimento dalyje nešiojamame kompiuteryje su ontologijomis, turinčiomis 1296 ir 1570 egzempliorių, nebuvo vykdomos Q3, Q8 ir Q9 užklausos dėl atminties trūkumo, kur atminties *Eclipse* įrankiui buvo galima skirti iki 1300 MB, todėl antra eksperimento dalis buvo pakartota serveryje, tiriant tik tas užklausas, kurioms įvykdyti neužteko atminties, padidinant skiriamą atminties kiekį *Eclipse* įrankiui iki 4096 MB. Ontologijos su 808 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai pateikti 5.20 paveiksle.



5.20 pav. Ontologijos su 808 egzemplioriais *SPARQL* užklausų vidutiniai vykdymo laikai

4. Užklausų tipų. Tam tikrų užklausų vidutinis vykdymo laikas didėjant egzempliorių skaičiui labai išauga, pasiekus tam tikrą egzempliorių skaičių – jų vykdymas tampa neįmanomas dėl atminties trūkumo. Šios užklausos – Q3, Q8 ir Q9 – pasižymi tuo, kad jose yra šių naujų *SPARQL 1.1* galimybių deriniai:

Q3: vidinės užklausos, agregavimo funkcijos `COUNT`, rekursijos;

Q8: agregavimo funkcija `COUNT` su `GROUP BY`, filtravimas `HAVING`;

Q9: agregavimo funkcija `COUNT` su `GROUP BY`, filtravimas `HAVING`, rekursija.

5. Ontologijos išvedimo. Atliekant užklausų vykdymą be ontologijos išvedimo, užklausų Q3, Q8 ir Q9 vykdymo laikai gaunami ženkliai mažesni, neiškyla problemų dėl atminties trūkumo, tačiau ontologijos išvedimas yra būtinas norint pasinaudoti visomis jos teikiamomis galimybėmis, kurių gali prireikti vykdant paiešką ontologijoje.

Eksperimentas parodė, kad *Pellet* išvedimo mechanizmas turi būti tobulinamas vykdant užklausas su tam tikrais *SPARQL* užklausų tipais, kadangi užklausos su *SWRL* taisyklėmis yra vykdomos su mažu egzempliorių skaičiumi, o be *SWRL* – esant mažesniam atminties kiekiui – susiduriama su atminties trūkumo problemomis. *SWRL* taisykles yra tikslinga vartoti tose

užklausose, kur yra išvedama daug tarpinių rezultatų. *Pellet* išvedimo biblioteka nėra pritaikyta vykdyti išvedimą ir užklausas su didesniu egzempliorių skaičiumi, be to norint taikyti *SWRL* taisykles užklausų vykdymas galimas tik su dar mažesniu egzempliorių skaičiumi ontologijoje (iki 155 egzempliorius turinčia ontologija) negu šių taisyklių nenaudojant (iki 1570 egzempliorius turinčia ontologija, tačiau ne su visomis užklausomis). Padidinus skiriamos operatyvios atminties kiekį *Eclipse* įrankiui gauti vidutiniai užklausų vykdymo laikai serveryje didėjant egzempliorių skaičiui nėra korektiški paieškos užklausoms.

Reiktų paminėti, kad egzempliorių skaičius nusako tik individų skaičių ontologijoje, tačiau jie gali įgyti skirtingus vaidmenis (būti tėvu, būti seneliu ir t.t.), ontologijose saugomi individai tripletais, todėl tripletų atžvilgiu jų yra daugiau negu egzempliorių, kadangi vienas ir tas pats individas gali dalyvauti keliuose vaidmenyse, t.y. turėti ne vieną savybę, o kelias.

Taip pat buvo bandoma išvedimą atlikti ontologijų redaktoriuje, o užklausą be išvedimo atlikti *Eclipse* aplinkoje, tačiau toks variantas nepasitvirtino, nes išvedimo mechanizmas *HermiT* ontologiją su 2034 egzemplioriais išvedė per dvi paras. *HermiT* buvo pasirinktas todėl, kad *Protégé* 4.1 versijoje *Pellet* išvedimo mechanizmo neįtraukė, tačiau vykdyti užklausų *Eclipse* aplinkoje *HermiT* neturi tam reikalingų bibliotekų (2.5.1 skyrius 2.11 lentelė).

Eksperimentą ar jo dalį būtų galima pakartoti ir su kitais įrankiais, išbandant jų galimybes susidorojant su išvedimu ir užklausomis priklausomai nuo egzempliorių skaičiaus ontologijoje.

6. Išvados

1. Semantinio tinklo užklausų vykdymo priemonių vertinimo etalonų analizė parodė, kad esami vertinimo etalonai neapima visų ontologijų savybių, užklausų tipų ir *SWRL* taisyklių.

2. Todėl buvo nuspręsta sukurti naują etaloną, kurio ontologijas būtų galima naudoti naujų tipų užklausų vykdymo charakteristikų tyrimui. Etalono ontologija pasirinktas genealoginis medis, nes jis leidžia tirti iki šiol netirtas rekursines užklausas.

3. Atliktas eksperimentas su *Pellet* įrankiu *Eclipse* aplinkoje parodė, kad *Pellet* išvedimo biblioteka nėra optimizuota tam tikriems *SPARQL 1.1* užklausų tipams esant didesniam ontologijos egzempliorių skaičiui. *Pellet* išvedimo mechanizmo atliktas ontologijos išvedimas daro neigiamą įtaką užklausų vykdymo laikui.

4. Eksperimento metu buvo nustatyta, kad *SWRL* taisyklių taikymas leidžia supaprastinti užklausų kūrimą, tačiau *Pellet* įrankis kai kuriais atvejais (pavyzdžiui, taikant *SWRL* individus atskiriančią savybę) nevykdo užklausų net esant nedideliam egzempliorių skaičiui ontologijoje.

5. Tyrimas parodė, kad sukurtas etalonas yra vertingas, nes leido įvertinti iki šiol nenagrinėtų užklausų tipų veikimą ir atskleidė, kokių tipų užklausoms ir *SWRL* taisyklėms tikslinga tobulinti *Pellet* įrankį.

6. Sukurtą įrankį ir tyrimo metodiką galima pritaikyti kitų *SPARQL 1.1* užklausų vykdymo priemonių tyrimui.

7. Remiantis šiuo darbu tema „Įrankis *SPARQL 1.1* užklausų vykdymo galimybėms tirti“ atspausdintas straipsnis ir perskaitytas pranešimas XVII tarpuniversitetinėje magistrantų ir doktorantų konferencijoje „Informacinės technologijos 2012“.

7. Literatūra

1. Antoniuo, G.; and Harmelen, F. Web Ontology Language: OWL. *In Staab, S., Studer, R. (Eds.) Handbook on Ontologies.* Second edition, New York, 2009, pp. 91-111. ISBN 978-3-540-70999-2.
2. Balani, N. The future of the Web is Semantic. Ontologies form the backbone of a whole new way to understand online data. *developerWorks* [interaktyvus]. 2005, [žiūrėta 2010-10-02]. Prieiga per internetą: <http://www.ibm.com/developerworks/web/library/wa-semweb/>.
3. Berners-Lee T. Semantic Web Road map. *The World Wide Web Consortium (W3C).* 1998, [žiūrėta 2010.11.06]. Prieiga per internetą: <http://www.w3.org/DesignIssues/Semantic.html/>
4. Bizer, C., Schultz, A. The Berlin SPARQL Benchmark. *International Journal on Semantic Web & Information Systems*, 2009, Vol. 5, (2), pp. 1-24.
5. Cardoso, J. Semantic *Web Services: Theory, Tools, and Applications*. Portugal, 2007, pp. 72-90. ISBN 978-1-59904-045-5.
6. Chebotko, A., Lu, S., Fotouhi, F. Semantics preserving SPARQL-to-SQL translation. *Data & Knowledge Engineering*, 2009, Vol. 68 (10), pp. 973-1000.
7. Dodds, L. Introducing SPARQL: Querying the Semantic Web. *O'Reilly xml.com* [interaktyvus]. 2005, [žiūrėta 2010-11-09]. Prieiga per internetą: <http://www.xml.com/pub/a/2005/11/16/introducing-sparql-querying-semantic-web-tutorial.html>.
8. Guo, Y., Pan, Z., Heflin, J. LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics*, 2005, Vol. 3 (2-3), pp. 158-182.
9. Kapoor, B., Sharma, S. A Comparative Study Ontology Building Tools for Semantic Web Applications. *International journal of Web & Semantic Technology*, 2010, Vol. 1, Nr. 3, pp. 1-13.
10. Li, M., Yang, Y., Qui, Z., Xie, G., Pan, Y., Liu, S. Towards A Complete OWL Ontology Benchmark. *Proceedings of the 3rd European conference on The Semantic Web*, Berlin, 2006.
11. Melton, J. SQL, XQuery, and SPARQL - what is wrong with this picture [interaktyvus]. 2005, [žiūrėta 2010-12-05]. Prieiga per internetą: <http://www.wiscorp.com/H2-2006-006-SQL-XQuery-and-SPARQL-What_s-Wrong-With-This-Picture.pdf>.
12. Mickevičiūtė E., Nemuraitė L. Įrankis SPARQL 1.1 užklausų vykdymo galimybėms tirti. XVII tarpuniversitetinė magistrantų ir doktorantų konferencijos „Informacinės technologijos 2012“ pranešimų medžiaga. Vilniaus universitetas, Vytauto Didžiojo universitetas, Kauno technologijos universitetas. [Kaunas] : [Vilniaus universitetas Kauno humanitarinis fakultetas]. ISSN 2029-249X, pp. 139-142., 2012

13. Motik, B., Patel-Schneider, P.F., Grau, B.C. OWL 2 Web Ontology Language Direct Semantics. *W3C* [interaktyvus]. 2009, [žiūrėta 2010-10-02]. Prieiga per internetą: <http://www.w3.org/TR/2009/REC-owl2-direct-semantics-20091027/>.
14. Pellet OWL Reasoner. *mindswap* [interaktyvus]. 2006, [žiūrėta 2011-12-06]. Prieiga per internetą: <http://www.mindswap.org/2003/pellet/>.
15. Prudhommeaux, E., Seaborne, A. SPARQL Query Language for RDF. *W3C* [interaktyvus]. 2008, [žiūrėta 2010-10-12] . Prieiga per internetą: <http://www.w3.org/TR/rdf-sparql-query/>.
16. Rahamatullah Khondoker, M., Mueller, P. Comparing Ontology Development Tools Based on an online Survey. *Proceedings of the World Congress on Engineering*, London, UK, 2010, Vol. I.
17. Schmidt M., Hornung T., Lausen G., Pinkel C. SP^2Bench: A SPARQL Performance Benchmark. *2009 IEEE 25th International Conference on Data Engineering*, Shanghai, 2009, pp. 222-233.
18. Sirin, E., Bulka, B., Smith, M. Terp: Syntax for OWL-friendly SPARQL queries. *Seventh International Workshop*, 2010, San Francisco, California, USA.
19. SPARQL 1.1 Query Language. *W3C* [interaktyvus]. 2011, [žiūrėta 2011-05-14]. Prieiga per internetą: <http://www.w3.org/TR/sparql11-query/>.
20. SPARQL 1.1 Update. *W3C* [interaktyvus]. 2011, [žiūrėta 2011-06-10]. Prieiga per internetą: <http://www.w3.org/TR/2009/WD-sparql11-update-20091022/>.
21. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. *W3C* [interaktyvus]. 2004, [žiūrėta 2012-02-05]. Prieiga per internetą: <http://www.w3.org/Submission/SWRL/#1>.
22. Weithoner, T., Liebig, T., Luther, M., Bohm, S. What‘s Wrong with OWL Benchmarks? *Proceedings of the Second InternationalWorkshop on Scalable Semantic Web Knowledge Base Systems*, Athens, GA, USA, 2006, pp. 101-114.

8. Priedai

1 priedas. Straipsnis

TURINYS

I sekcija. Formali analizė ir projektavimo metodai

Dalia Čalnerytė
Dvimatės struktūros medžiagos parametrų nustatymas iš trimatės kompozitinės struktūros 7

Laura Šeškutė
Dialogo valdymo modelių analizė 11

Tomas Rasymas
Lietuviškų skaitmenų atpažinimas naudojant anglų kalbos atpažinimo variklį 15

Tomas Rasymas
Balso komandų aptikimas triukšmingame kalbos signale skirtas automatinei kalbos atpažinimo sistemai 19

Mindaugas Vasiljevas, Rūtenis Turčinas, Ignas Martišius
Netiesinių operatorių taikymas EEG duomenų apdorojimui 23

Vytautas Simanaitis, Agnius Liutkevičius, Arūnas Vrubliauskas, Egidijus Kazanavičius
MPEG-2 transporto srauto dalinio šifravimo metodas 27

Justina Čenytė, Raimundas Jasinevičius
Kontekstas ir kontekstinis panašumas 31

Ieva Paužaitė, Vytautas Ašeris
Kompiuterinis biojutiklių modeliavimas taikant kintamą diskrečiosios gardelės erdvės žingsnį 35

Kęstutis Šidlauskas, Ignas Martišius
EEG duomenų klasifikavimas naudojant balsavimo ir daugiasluoksnius perceptronus 39

II sekcija. Informacinių technologijų taikymai

Vigintas Šakys, Edvardas Pranckevičius
Universiteto akademinių duomenų švieslenčių kūrimo metodikos ir priemonių tyrimas 45

Darius Ašeriškis, Justas Tamošaitis
Kitokia programinės įrangos įmonė 49

Egidijus Babenskas, Šarūnas Packevičius
Automatinis testų generavimas testuojant Android OS aplikacijas 53

Vytautas Valaitis
Judesiai gamtoje ir dirbtinėse sistemose 57

Gita Burbaitė
Aukšto lygio kompiuterizuoto testavimo sistemos modelio modifikavimas e-Guardian v2.0 programos realizavimui 61

Giedrė Česonytė, Vladislav V.Fomin
Elektroninio dienyno vaidmuo mokykloje 65

Egidijus Galkus, Paulius Danėnas, Gintautas Garšva
Kolektyvinių klasifikavimo metodų taikymas kredito rizikos vertinime 70

III sekcija. Programinės įrangos inžinerija

Šarūnas Dargelis
Modified Scrum project management method 77

Edminas Vrubliauskas, Lina Tutkutė
Veiklos proceso modelio praplėsto veiklos taisyklėmis transformavimas į veiklos paslaugas 81

Birutė Kaminskaitė, Lina Tutkutė
Veiklos taisyklių integracijos į veiklos procesų modelį metodas 85

Stasys Peldžius
Tarpinio programų kūrimo proceso modelio formalizuotas aprašymas 89

Milda Mickevičiūtė, Lina Tutkutė
Automatizuotas veiklos žodyno išgavimo iš veiklos procesų diagramos metodas ..93

Andrius Kondratas, Lina Nemuraitė
Įrankis OWL 2 ontologijoms transformuoti į reliacines duomenų bazes..97

Živilė Misevičiūtė
Žinių inžinerijos principų taikymas Hendersono veiklos ir IT suderinimo modelyje..101

Dovilė Šukytė
Mobiliųjų informacinių sistemų vartotojo reikalavimų specifikavimo metodas..105

Ilona Veitaitė, Audrius Lopata
Veiklos modeliu grindžiamas UML diagramų generavimas ..109

IV sekcija. Duomenų bazės ir informacinės sistemos

Arūnas Bareišis, Aušra Kentraitė
Deployment of data mining and machine learning tools to identify factors influencing alcohol consumption: A cross-country anglysis..115

Aušra Kentraitė, Arūnas Bareišis
Evaluating companies interoperability using DEXi..119

Vigintas Šakys, Tomas Skroblas
Duomenų gavybos priemonių taikymas studentų priėmimo į universitetą tyrimui ..123

Vaidas Mikalauskas, Gintautas Garšva
Tinklalapių semantikos tyrimas Google paieškos sistemoje..127

Aistė Dabrilaitė, Auksė Stravinskienė, Saulius Gudas
Sprendimų medžio algoritmų panaudojimas duomenų gavyboje ..131

Gintarė Sukarevičienė, Vladislav V.Fomin
Business model analysis: The case of TV hite space databases..135

Eglė Mickevičiūtė, Lina Nemuraitė
Įrankis SPARQL 1.1 užklausų vykdymo galimybėms tirti..139

Gintarė Bernotaitytė, Lina Nemuraitė
Teminių tinklų kūrimo metodika grindžiama ontologijų sudarymo principais..143

Kęstutis Spudys, Lina Nemuraitė
Dinaminio semantinių užklausų formavimo sąsaja ..147

V sekcija. Kompiuterių sistemos

Nerijus Jusas, Algimantas Venčkauskas
Programos apsaugos metodas, naudojant sistemos parašą..153

Giedrė Jankauskienė, Algimantas Venčkauskas
Informacijos saugos audito sprendimų paramos sistema ..157

Andrius Čepaitis, Vytautas Ašeris
Efektyvi virtualizacija skaičiavimams debesyse..161

Marius Liutvinavičius, Donatas Mažeika, Arvydas Dudavičius, Tomas Janulevičius
Socialinių tinklų decentralizavimo galimybių tyrimas ..165

Julius Kriukas, Rimantas Kavaliūnas
Laisvai pasirenkamo mazgo identifikatoriaus įtakos DHT tinklo saugumui analizė..169

Andrius Randis, Klaidas Gudžiūnas, Eimantas Misevičius
Socialinių tinklų ir verslo sąveika..173

Laurynas Dovydaitis
Vienetinio prisijungimo galimybės hibridinių debesų kompiuterijoje..177

Aurelijus Jakas
Socialinių tinklų įtakos elektroninei komercijai vertinimo sistemos modelis..181

ĮRANKIS SPARQL 1.1 UŽKLAUSŲ VYKDYMO GALIMYBĖMS TIRTI

Eglė Mickevičiūtė[1], Lina Nemuraitė[2]

[1,2]Kauno Technologijos Universitetas, Informacijos sistemų katedra, Studentų g. 50, Kaunas, e.mickeviciute@stud.if.ktu.lt[1], lina.nemuraite@ktu.lt[2]

Santrauka. Besivystančiame Semantiniame pasauliniame tinkle nuolat atsiranda naujų kalbų ir technologijų, kurių galimybes nori žinoti jas besirenkantys vartotojai. Norint pasirinkti tinkamą technologiją, joms palyginti taikomi taip vadinami etalonai (ang. *Benchmark*). Šio darbo tikslas – sukurti įrankį, leidžiantį palyginti ontologijų užklausų kalbos *SPARQL* vykdymo priemones naujų *SPARQL 1.1* versijos galimybių, kurių neapima esami *SPARQL* vykdymo priemonių tyrimo etalonai, atžvilgiu. Straipsnyje analizuojami tyrimo principai, sukurtas įrankis, pasirinkta ontologija ir SPARQL 1.1 užklausų rinkinys.

Raktiniai žodžiai: Semantinis tinklas, ontologija, etalonas, *OWL 2*, *RDF*, *SPARQL 1.1*.

1 Įžanga

Semantinis tinklas – tai terminas, kurį sugalvojo *Tim Berners-Lee*, ir kuris reiškia ateities interneto tinklą, kuris yra suvokiamas kaip globali duomenų bazė [1]. Semantinio tinklo technologijos padeda išskirti reikšmes iš duomenų (informacijos), dokumento turinio, ar taikomosios programos kodo, naudojant technologijas, kurios paremtos atviraisiais standartais. Jei kompiuteris suprastų dokumento semantiką, o ne tik interpretuotų jį kaip simbolių rinkinį, tuomet jis suprastų dokumento esmę, kam jis yra skirtas. Semantinės technologijos naudojasi ontologijomis ir stengiasi pateikti argumentus per ryšius, taisykles, logiką ir sąlygas, kurios yra apibrėžtos tose ontologijose.

Besivystančio Semantinio pasaulinio tinklo ontologijų ir jų užklausų kalbos atsirado siekiant įgyvendinti Semantinio tinklo viziją – pateikti ir surasti pagal prasmę susietus duomenis, esančius bet kuriame tinklo taške. Šiuo metu yra daug semantinės paieškos technologijų, tačiau jų gaunama informacija dažnai yra perteklinė ir netiksli. Šiuo metu pažengusios semantinės technologijos siejamos su ontologijų kalba *OWL 2*, išteklių aprašymo kalba *RDF*, bei *OWL 2* užklausų kalba *SPARQL 1.1*.

Daugėjant informacijos internete susiduriama su reikiamos informacijos radimo ir laiko, per kurį galima surasti patikimą informaciją, problemomis. Vartotojai, besirenkantys semantines technologijas, nori žinoti jų galimybes. Norint turėti vieningą matą palyginti skirtingas technologijas, kuriami tam skirti etalonai. Šiuo metu ypač aktualu žinoti naujos semantinio tinklo užklausų kalbos *SPARQL 1.1* versijos užklausų vykdymo galimybes. *SPARQL* etalonas apima tam tikrą ontologiją, užklausų rinkinį ir ontologijos egzempliorių generavimo įrankį, kuris leidžia sugeneruoti įvairaus dydžio egzempliorių rinkinius. Vykdant etalonines užklausas su įvairaus dydžio ontologijomis, galima įvertinti semantinio tinklo pajėgumą susidorojant su egzempliorių gausa, užklausų ypatumais ir ontologijos charakteristikomis. Nors yra keletas nuo seno naudojamų *SPARQL* etalonų, tačiau šiuo metu jų jau neužtenka, todėl šiame darbe siekiama tokį etaloną sukurti.

Straipsnio struktūra. Antrame skyriuje analizuojami esami *SPARQL* etalonai, trečiame – ontologija ir jos egzempliorius generuojantis įrankis, ketvirtame pateikiamas užklausų rinkinys, pabaigoje – išvados.

2 Semantinio tinklo įrankių įvertinimo etalonų analizė

Naujoje *SPARQL* užklausų kalbos versijoje *SPARQL 1.1* įtrauktos naujos galimybės [6], [7]: iteracijos, agregavimo funkcijos (*MIN*, *MAX*, *COUNT*, *AVG*, *SUM*), sub-užklausos, neigimas, filtravimas, duomenų manipuliavimo operacijos *INSERT*, *UPDATE*, *DELETE*. Vartotojams aktualu žinoti, kokios šių funkcijų veikimo charakteristikos įvairiuose užklausų vykdymo įrankiuose.

Tiriant įrankių savybes *SPARQL* etalonu, tyrimai atliekami su tos pačios ontologijos pavyzdžiais, turinčiais skirtingą sugeneruotų egzempliorių skaičių. Tiriamas - ontologijų užkrovimo laikas, galimybės išvesti rezultatus priklausomai nuo *OWL* sub-kalbų (*OWL-Lite*, *OWL-DL*), užklausų vykdymo laikas, užbaigtumas (grąžinamas sistemos atsakymas į užklausą) bei tikslumas (užbaigtumo tikslumas), be to, semantinio tinklo įrankiai ir jų charakteristikos, lyginamos su kitomis technologijomis (pvz.: reliacinėmis duomenų bazių valdymo sistemomis(DBVS)). Kadangi kiekvienas etalonas turi tik vieną ontologiją ir užklausų rinkinį, vieno etalono nepakanka, kad būtų galima ištirti visas *SPARQL* užklausų kalbos ir įrankių charakteristikas.

Geriausiai žinomi ir taikomi yra seniausias *LUBM* (*Lehigh University Benchmark*) etalonas, kuris buvo patobulintas į *OUBM* (*University Ontology Benchmark*) ir *Berlin SPARQL* etalonas.

LUBM etalone naudojama universiteto ontologija ir 14 užklausų. Jis leidžia palyginti ontologijų užkrovimo ir užklausų vykdymo greitį, bei įvertinti gaunamų atsakymų teisingumą priklausomai nuo ontologijos egzempliorių skaičiaus. Tačiau *LUBM* negali pilnai ištestuoti visų ar bent jau daugelio semantinio tinklo įrankių

galimybių [3]: neapima visų *OWL – Lite* ir *OWL – DL* konstrukcijų, o jo sugeneruojami egzemplioriai, kurie sudaro medžius, tarpusavyje nesusieti. Šie trūkumai buvo pataisyti patobulintame *LUBM* variante *OUBM*.

Berlin SPARQL etalonas naudoja e-komercijos ontologiją aprašančią produktus, kuriuos siūlo įvairūs pardavėjai ir pirkėjus, kurie gali rašyti atsiliepimus apie siūlomus produktus. *Berlin SPARQL* etalonas turi 12 užklausų ir matuoja šias metrikas: užklausų įvairovę per valandą, užklausas per sekundę, duomenų užkrovimą. Šiuo etalonu buvo tiriamos, pavyzdžiui, *RDF* saugyklų (*Sesame*, *Virtuaso*, *Jena TDB* ir *Jena SDB*), *SPARQL-to-SQL* transliatorių (*D2R Server* ir *Virtuoso RDF Views*), reliacinių DBVS (*MySQL* ir *Virtuaso RDBMS*) galimybės [2].

Iš esamų etalonų dar galima paminėti SP^{2}Bench, kurio autoriai giriasi, kad jų etalonas sugeneruoja itin tikslius duomenis, atitinkančius realaus pasaulio reikalavimus. Jų pasirinkta ontologija - publikacijos. Šis etalonas skiriasi nuo jau paminėtų tuo, kad norima išanalizuoti užklausų veikimą bendruoju atveju, netaikant konkrečiam įrankiui. SP^{2}Bench etalone yra 17 užklausų [4], tačiau jos neapima rekursinių bei kitų naujų *SPARQL 1.1* užklausų tipų.

Minėti etalonai didžiausią dėmesį skiria į semantinio tinklo užklausų įrankių galimybes, tačiau nevertina loginio išvedimo [5] ir pačios *SPARQL* užklausų kalbos. Paprastai jie gali parinkti geriausius įrankius atskiriems taikymo scenarijams, bet ne įvertinti *SPARQL* galimybes bendruoju atveju.

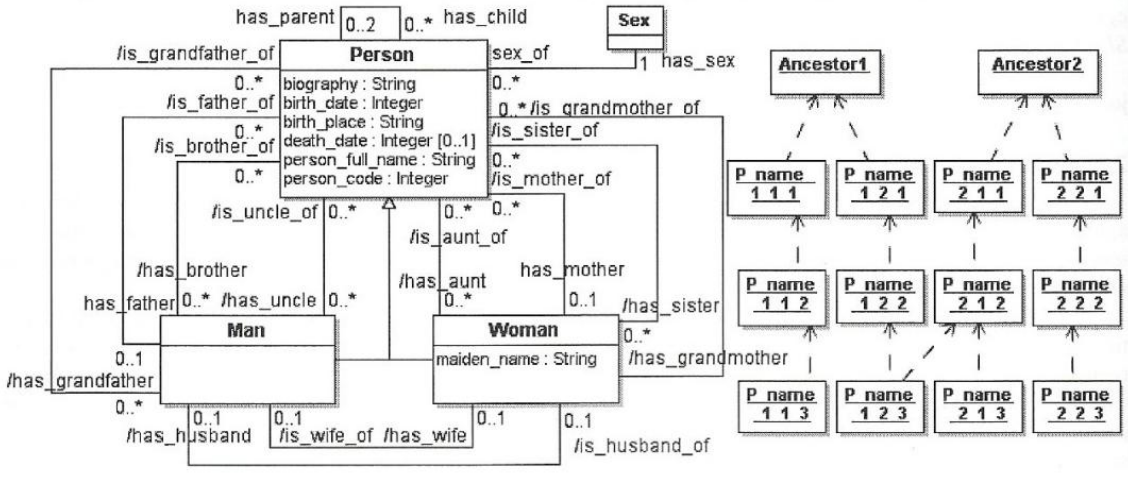
Nei vienas etalonas neatsižvelgia į užklausų optimizavimą taikant *SWRL* taisykles, kurios leidžia išvesti papildomų savybių ir supaprastinti *SPARQL* užklausas.

Lentelė Nr. 1 Etalonų analizė

Įrankis / savybė	*LUBM*	*Berlin Benchmark*	*SP^{2}Bench*	Siekiamas įrankis
Ontologijų kalba	*OWL*	*RDF*	*OWL*	*OWL 2*
Dalykinė sritis	Universitetų veikla	E-komercija	Publikacijos	Genealoginis medis
Tiriami parametrai	Egzempliorių skaičius	Egzempliorių skaičius	Egzempliorių skaičius ir užklausų tipai	Hierarchijos gylis ir plotis, ryšių skaičius, *SWRL* taikymas
Vertinimo objektas	Užklausų vykdymo galimybės ir saugojimo mechanizmai	Užklausų vykdymo galimybės ir saugojimo mechanizmai	Užklausų vykdymo galimybės	Užklausų vykdymo galimybės
Užklausų tipai	Savybėmis (*OUBM* ir ryšiais) grindžiamos užklausos	Savybėmis grindžiamos užklausos	Visos *SPARQL* užklausos	Rekursinės ir naujų tipų *SPARQL 1.1* užklausos

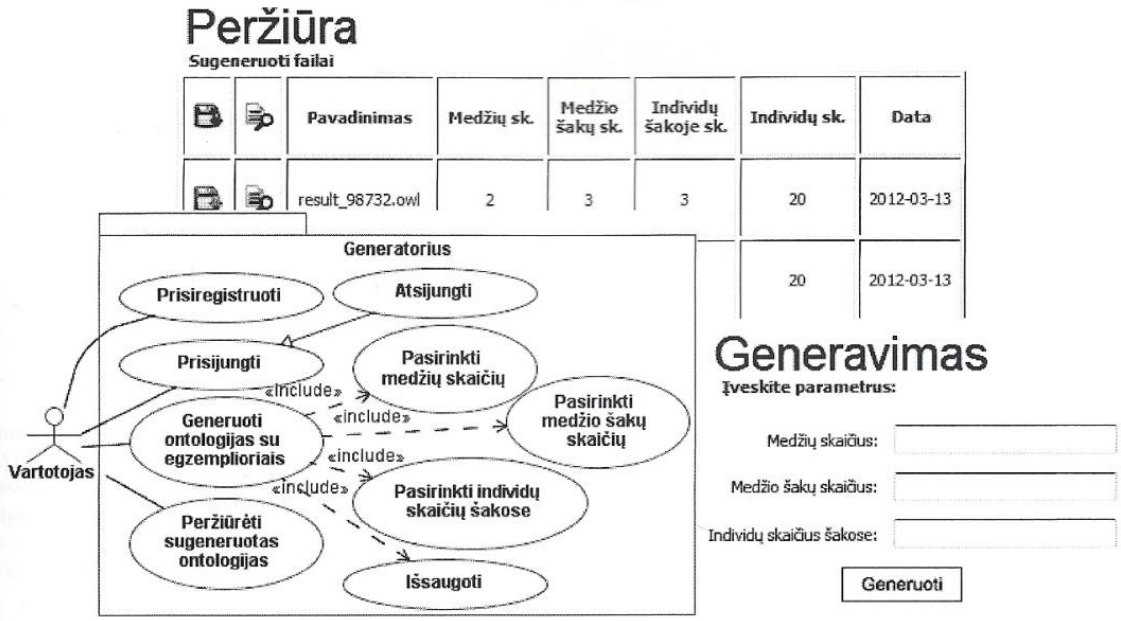
3 Ontologijos egzempliorių generavimo įrankis

Kadangi esantys sprendimai turi trūkumų, buvo nuspręsta sukurti naują etaloną, kuriam pasirinkta genealoginio medžio ontologija, kurioje galima vykdyti rekursines bei kitas naujas *SPARQL* užklausas. Genealoginio medžio *OWL 2* ontologijos fragmentas ir sugeneruoti egzemplioriai pateikti 2 paveiksle.



1 pav. Genealoginio medžio *OWL 2* ontologijos fragmentas ir sugeneruoti egzemplioriai

Sukurtame įrankyje vartotojas gali nurodyti generavimo parametrus, generuoti ontologijos egzempliorius, išsaugoti juos failuose, peržiūrėti savo sugeneruotus ontologijos failus. Ontologijos generavimo įrankio panaudojimo atvejų modelis ir langų fragmentai pateikti 2 paveiksle.



2 pav. Panaudojimo atvejų diagrama ir ontologijos generavimo įrankio langų fragmentai

4 Užklausų rinkinys tyrimui

Įvertinant *SPARQL 1.1* naujas galimybes buvo sudarytas užklausų rinkinys, kuris pateiktas 2 lentelėje. Tripletų ar įrašų modifikavimas, redagavimas ar šalinimas turėtų būti atliekamas ne tiesiogiai užklausomis, bet su atitinkama vartotojo sąsaja, todėl į rinkinį šios *SPARQL* užklausos neįtrauktos.

Lentelė Nr. 2. Užklausų rinkinys

Nr.	Užklausa	Kodas
1.	Surasti visus vieno asmens gimines	`SELECT ?kin WHERE {gen:Person_name_1_2_2(gen:has_child|gen:has_parent)* ?kin}`
2.	Surasti visų asmenų visas gimines	`SELECT ?person ?kin WHERE {?person rdf:type gen:Person . ?person (gen:has_child|gen:has_parent)* ?kin FILTER ( ?person != ?kin )}`
3.	Surasti kartų skirtumą tarp dviejų asmenų	`SELECT (fn:abs((?count1 - ?count2)) AS ?range) WHERE {{SELECT (count(?x) AS ?count1) WHERE {gen:Person_name_1_1_1 (gen:has_parent)* ?x}} {SELECT (count(?x) AS ?count2) WHERE {gen:Person_name_1_1_2 (gen:has_parent)* ?x}}}`
4.	Surasti asmens pirmtaką	`SELECT ?ancestor WHERE {gen:Person_name_2_2_2 gen:has_parent* ?ancestor. not exists {?ancestor gen:has_parent ?y}}`
5.	Surasti visų asmenų pirmtakus	`SELECT ?person ?ancestor WHERE {?person rdf:type gen:Person . ?person (gen:has_parent)* ?ancestor NOT EXISTS {?ancestor gen:has_parent ?y}}`
6.	Surasti asmens senelį (1) be *SWRL* ir (2) su *SWRL*	`(1) SELECT ?antecedent WHERE {gen:Person_name_1_2_3 (gen:has_parent){2,2} ?antecedent}` `(2) SELECT ?antecedent WHERE {gen:Person_name_1_2_3 (gen:has_grandparent) ?antecedent}`
7.	Surasti visų asmenų senelius (1) be *SWRL* ir (2) su *SWRL*	`(1) SELECT ?person ?antecedent WHERE {?person (gen:has_parent){2,2} ?antecedent}` `(2) SELECT ?person ?antecedent WHERE {?person (gen:has_grandparent){2,2} ?antecedent}`
8.	Surasti asmenis, kurie turi ne mažiau negu 2 vaikus	`SELECT ?parent (count(?child) AS ?child_count) WHERE {?parent gen:has_child ?child} GROUP BY ?parent HAVING (?child_count > 1)`
9.	Surasti nurodytos kartos asmenis	`SELECT ?people (count(?ancestor) AS ?range) WHERE {?people rdf:type gen:Person . ?people (gen:has_parent)* ?ancestor} GROUP BY ?people HAVING (?range = 2)`

Nr.	Užklausa	Kodas
10.	Surasti visus asmenis, kurie yra paskutiniai kartoje	`SELECT DISTINCT ?last WHERE {?people rdf:type gen:Person . ?people (gen:has_child)* ?last NOT EXISTS {?last gen:has_child ?x}}`
11.	Surasti visų asmenų dėdes (1) be *SWRL* ir (2) su *SWRL*	`(1) SELECT ?person ?uncle WHERE {?person gen:has_parent ?x . ?x gen:has_parent ?y . ?y gen:has_child ?uncle . ?uncle gen:has_sex gen:male_sex FILTER (?x != ?uncle)} (2) SELECT ?person ?uncle WHERE {?person (gen:has_uncle) ?uncle}`
12.	*SWRL*, papildomai aprašomos *SWRL* taisyklės	`has_parent(?x, ?y), has_parent(?y,?z) -> has_grandparent(?x, ?z); has_grandparent(?x, ?y), has_child(?y, ?z), has_sex(?z, male_sex) -> has_uncle(?x, ?z)`

Pasinaudojus šiame darbe sukurtu ontologijos egzempliorių generatoriumi, bus sugeneruojami genealoginio medžio ontologijos pavyzdžiai su skirtingu egzempliorių skaičiumi, jiems vykdomos *SPARQL* užklausų rinkinys, įvertinamas šių užklausų vykdymo laikas, gautų rezultatų išbaigtumas ir teisingumas. Eksperimentas bus kartojamas su skirtingus parametrus turinčia kompiuterine įranga, atsižvelgiant į operatyviąją atmintį (*RAM*), procesoriaus dažnį, branduolių skaičių ar kietojo disko charakteristikas.

5 Išvados

Atsirandant naujiems semantinio tinklo įrankiams, kalboms, jiems tobulėjant reikia ištirti jų galimybes. Atlikus esamų etalonų analizę paaiškėjo, kad esami etalonai neturi užklausų, kurios apimtų naujas *SPARQL 1.1* užklausų kalbos galimybes, neapima visų ontologijų ir užklausų savybių, nėra panaudojamos *SWRL* taisyklės bei didžiausią dėmesį skiria semantinio tinklo įrankių tyrimui, o ne *SPARQL* užklausų kalbos vykdymo tyrimui. Todėl buvo nuspręsta, kad reikia kurti naują generavimo įrankį, su kuriuo būtų galima atlikti *SPARQL* užklausų vykdymo tyrimą, sudaryti *SPARQL* užklausų klasifikaciją, tyrimui sudaryti užklausų šablonų rinkinį, atsižvelgiant į naujas *SPARQL 1.1* galimybes. Tripletų ar įrašų modifikavimas, redagavimas ar šalinimas nėra įtraukiamas.

Užklausų rinkinys buvo sukurtas rekursinėms ir kitoms naujoms *SPARQL 1.1* užklausoms testuoti. Pagrindinis akcentas - rekursinės užklausos, kurių galimybių nenagrinėja nei vienas žinomas etalonas. Šioms užklausoms realizuoti buvo pasirinktas *Eclipse* įrankis su *Pellet OWL Reasoner* išvedimo ir *SPARQL* vykdymo mechanizmu, pasirinkta etaloninė ontologija - genealoginio medžio ontologija ir sukurtas šios ontologijos egzempliorių generavimo įrankis ontologijos failų kūrimui su skirtingu egzempliorių skaičiumi.

Literatūros sąrašas

[1] **Berners-Lee T.** Semaptic Web Road map. *The World Wide Web Consortium (W3C).* 1998 [žiūrėta 2010.11.06]. access via the Internet: http://www.w3.org/DesignIssues/Semantic.html

[2] **Bizer C., Schultz A.** The Berlin SRARQL Benchmark. *International Journal on Semantic Web & Information Systems.* 2009, volume 5, issue 2, 1-24.

[3] **Li M., Yang Y., Qui Z., Xie G., Pan Y., Liu S.** Towards A Complete OWL Ontology Benchmark. *Proceedings of the 3rd European conference on The Semantic Web.* 2006.

[4] **Schmidt M., Hornung T., Lausen G., Pinkel C.** SP^{2}Bench: A SPARQL Performance Benchmark. *2009 IEEE 25th International Conference on Data Engineering.* 2009, 222-233.

[5] **Weithoner T., Liebig T., Luther M., Bohm S.** A What's Wrong with OWL Benchmarks? *Second InternationalWorkshop on Scalable Semantic Web Knowledge Base Systems.* 2006, 101-104.

[6] **W3C.** SPARQL 1.1 Query Language. *The World Wide Web Consortium (W3C).* 2011 [žiūrėta 2011.05.14]. access via the Internet: http://www.w3.org/TR/sparql11-query/

[7] **W3C.** SPARQL 1.1 Update. *The World Wide Web Consortium (W3C).* 2011 [žiūrėta 2011.06.10]. access via the Internet: http://www.w3.org/TR/2009/WD-sparql11-update-20091022/

A tool for exploring capabilities for executing SPARQL 1.1 queries

With the development of Semantic Web new languages and tools emerge whose capabilities are interesting for users. In order to select a suitable technology so-called benchmarks are used. The aim of this research is to create a tool for evaluating ontology query executing tools in respect to capabilities of new SPARQL 1.1 version, which is not analyzed by existing benchmarks. This paper presents the principles of the benchmarking, developed tool, chosen ontology and created set of SPARQL 1.1 queries.

INFORMACINĖS TECHNOLOGIJOS 2012

17-OJI TARPUNIVERSITETINĖ MAGISTRANTŲ IR DOKTORANTŲ KONFERENCIJA

SERTIFIKATAS

Šis sertifikatas liudija, kad XVII Tarpuniversitetinės magistrantų ir doktorantų konferencijos

INFORMACINĖS TECHNOLOGIJOS 2012

Sekcijoje *Duomenų bazės ir informacinės sistemos* išrinktas

GERIAUSIU

Autorių Eglė Mickevičiūtė

Straipsnis Įrankis SPARQL 1.1 užklausų vykdymo galimybėms tirti

Konferencijos pirmininkas doc.dr. A.Lopata

Vilniaus universiteto Kauno humanitarinis fakultetas
Kaunas
2012

2 priedas. Sugeneruoti ontologijos individai

```
<!-- http://www.w3.org/2002/07/owl#Thing -->
<owl:Class rdf:about="http://www.w3.org/2002/07/owl#Thing" />
<owl:Thing rdf:about="http://Data.Genealogy_tree/Ancestor1">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_1_1_1">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Ancestor1" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_1_1_2">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_1_1_1" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_1_1_3">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_1_1_2" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_1_1_4">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_1_1_3" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_1_2_1">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Ancestor1" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_1_2_2">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_1_2_1" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_1_2_3">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_1_2_2" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_1_2_4">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_1_2_3" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_2_1_3" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Ancestor2">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
```

```
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_2_1_1">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Ancestor2" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_2_1_2">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_2_1_1" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_2_1_3">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_2_1_2" />
<has_sex rdf:resource="http://Data.Genealogy_tree/female_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_2_1_4">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_2_1_3" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_2_2_1">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Ancestor2" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_2_2_2">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_2_2_1" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_2_2_3">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_2_2_2" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_2_2_4">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_2_2_3" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_3_1_3" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Ancestor3">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_3_1_1">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
<has_parent rdf:resource="http://Data.Genealogy_tree/Ancestor3" />
<has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
  </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_3_1_2">
<rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
```

```
 <has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_3_1_1" />
 <has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
   </owl:Thing>
<owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_3_1_3">
 <rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
 <has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_3_1_2" />
 <has_sex rdf:resource="http://Data.Genealogy_tree/female_sex" />
   </owl:Thing>
 <owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_3_1_4">
 <rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
 <has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_3_1_3" />
 <has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
   </owl:Thing>
 <owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_3_2_1">
 <rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
 <has_parent rdf:resource="http://Data.Genealogy_tree/Ancestor3" />
 <has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
   </owl:Thing>
 <owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_3_2_2">
 <rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
 <has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_3_2_1" />
 <has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
   </owl:Thing>
 <owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_3_2_3">
 <rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
 <has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_3_2_2" />
 <has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
   </owl:Thing>
 <owl:Thing rdf:about="http://Data.Genealogy_tree/Person_name_3_2_4">
 <rdf:type rdf:resource="http://Data.Genealogy_tree/Person" />
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual" />
 <has_parent rdf:resource="http://Data.Genealogy_tree/Person_name_3_2_3" />
 <has_sex rdf:resource="http://Data.Genealogy_tree/male_sex" />
   </owl:Thing>
```