KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Giedrius Petrikas

OWL transformavimas į reliacinių duomenų bazių schemas

Magistro darbas

Darbo vadovė
prof. Lina Nemuraitė

Kaunas, 2010

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Giedrius Petrikas

OWL transformavimas į reliacinių duomenų bazių schemas

Magistro darbas

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

Vadovė
prof. Lina Nemuraitė
2010-05-31

Atliko
IFM-4/2 gr. stud.
Giedrius Petrikas
2010-05-31

Kaunas, 2010

Turinys

1. Įvadas 6
2. Ontologijų vaizdavimo reliacinėse duomenų bazėse metodų analizė 8
   2.1. OWL kalbos analizė 8
      2.1.1. Trys OWL kalbos poaibiai 8
      2.1.2. Kalbos dariniai ir sąvokos 9
   2.2. OWL vaizdavimo į reliacines duomenų bazes metodų analizė 10
      2.2.1. Ontologijų taikymas panaudojant duomenų bazes 10
      2.2.2. Ontologijų transformavimas į koncepcinius duomenų modelius 11
      2.2.3. Duomenų bazės struktūros ir jos duomenų vaizdavimas ontologijoje 11
      2.2.4. Pagrindiniai skirtumai tarp OWL ir reliacinių duomenų bazių 12
      2.2.5. Reliacinių duomenų bazių išplėtimas ontologijų palaikymui 13
      2.2.6. Universalios duomenų bazės schemos naudojimas 14
      2.2.7. Didelio masto informacinių sistemų ontologijų valdymas 14
      2.2.8. Ontologijos pavertimas duomenų bazės schema 15
      2.2.9. Ontologijų vaizdavimo transformavimas iš OWL į reliacinių duomenų bazių schemas 16
      2.2.10. Transformavimo metodų palyginimas 16
   2.3. Analizės išvados 17
3. Ontologijų transformavimo Algoritmas 18
   3.1. Algoritmo statinis vaizdas 18
   3.2. Algoritmo dinaminis vaizdas 22
      3.2.1. Apibendrintas dinaminis vaizdas 22
      3.2.2. Klasių transformavimo dinaminis vaizdas 23
      3.2.3. Objektų savybių transformavimo dinaminis vaizdas 24
      3.2.4. Duomenų tipų savybių transformavimo dinaminis vaizdas 26
      3.2.5. Apribojimų transformavimo dinaminis vaizdas 26
4. Ontologijų transformavimo į RDB Projektas 28
   4.1. Reikalavimų specifikacija 28
   4.1.1. Funkciniai reikalavimai 28
      4.1.1.1. Panaudojimo atvejų sąrašas 31
      4.1.1.2. Funkciniai reikalavimai ir reikalavimai duomenims 32
      4.1.1.2.1. Funkciniai reikalavimai 32
   4.1.1.3. Nefunkciniai reikalavimai 38
   4.2. Projekto modelis 39
   4.2.1. Architektūros pateikimas 39
   4.2.2. Architektūros tikslai ir apribojimai 39
   4.2.3. Sistemos statinis vaizdas 40
      4.2.3.1. Apžvalga 40
      4.2.3.2. Paketų detalizavimas 40
      4.2.3.2.1. Transformavimo algoritmas 40
      4.2.3.2.2. RDB valdymas 41
      4.2.3.2.3. Vartotojo sąsaja 42
      4.2.3.2.4. OWL2RDB 42
   4.2.4. Sistemos dinaminis vaizdas 43
      4.2.4.1. Apibendrintas dinaminis vaizdas 43
      4.2.4.2. Konfigūravimo dinaminis vaizdas 44
      4.2.4.3. Transformavimo dinaminis vaizdas 44
      4.2.4.4. Veiksmų RDB dinaminis vaizdas 47
   4.2.5. Įrankio kokybė 49
5. Realizacija ir eksperimentas 50
   5.1. Įrankio išdėstymas Protege sistemoje 50
   5.2. Realizacija 50
   5.3. Eksperimentas 58
   5.3.1. Internete patalpintų ontologijų transformavimo tyrimas 58
   5.3.2. Paieškos greitaveikos tyrimas 60
6. Išvados 64
7. Literatūra 65
8. Terminų ir santrumpų žodynas 67
9. Priedai 68
   9.1. Užklausų trukmės eksperimento rezultatai 68

Paveikslėlių sąrašas

Paveikslėlis Nr. 1. SnoBase architektūra 15
Paveikslėlis Nr. 2. Ontologijose taikomas klasių metamodelis 18
Paveikslėlis Nr. 3. Ontologijose taikomas savybių metamodelis 18
Paveikslėlis Nr. 4. Ontologijose taikomas apribojimų metamodelis 19
Paveikslėlis Nr. 5. Lentelių, atributų ir duomenų tipų metamodelis 19
Paveikslėlis Nr. 6. Unikalumo apribojimų ir išorinių raktų metamodelis 20
Paveikslėlis Nr. 7. Klasių transformacija ir pavyzdys 20
Paveikslėlis Nr. 8. Poklasių transformacija ir pavyzdys 21
Paveikslėlis Nr. 9. Objektų savybių transformacija ir pavyzdys 21
Paveikslėlis Nr. 10. Duomenų tipų savybių transformacija ir pavyzdys 21
Paveikslėlis Nr. 11. Apribojimų transformacija 22
Paveikslėlis Nr. 12. Kardinalumo apribojimų transformacija 22
Paveikslėlis Nr. 13. Transformavimo veiklos schema 23
Paveikslėlis Nr. 14. Klasių transformavimo veiklos diagrama 24
Paveikslėlis Nr. 15. Objektų savybių transformavimo veiklos diagrama 25
Paveikslėlis Nr. 16. Duomenų tipų savybių transformavimo veiklos diagrama 26
Paveikslėlis Nr. 17. Apribojimų transformavimo veiklos diagrama 27
Paveikslėlis Nr. 18. Konteksto diagrama 29
Paveikslėlis Nr. 19. Panaudojimo atvejų diagrama 30
Paveikslėlis Nr. 20. Įrankio išskaidymas į paketus 40
Paveikslėlis Nr. 21. Transformavimo algoritmo paketo klasių diagrama 41
Paveikslėlis Nr. 22. RDB valdymo paketo klasių diagrama 42
Paveikslėlis Nr. 23. Vartotojo sąsajos paketo klasių diagrama 42
Paveikslėlis Nr. 24. OWL2RDB paketo klasių diagrama 43
Paveikslėlis Nr. 25. Apibendrinta įrankio bendradarbiavimo diagrama 43
Paveikslėlis Nr. 26. Konfigūravimo sekos diagrama 44
Paveikslėlis Nr. 27. Įrankio transformavimo veiklos schema 45
Paveikslėlis Nr. 28. Įrankio transformavimo proceso būsenų diagrama 45
Paveikslėlis Nr. 29. Transformavimo sekos schema 46
Paveikslėlis Nr. 30. Veiksmų RDB sekos diagrama 47
Paveikslėlis Nr. 31. Veiksmų RDB veiklos diagrama 48
Paveikslėlis Nr. 32. Veiksmų RDB būsenų diagrama 48
Paveikslėlis Nr. 33. Įrankio išdėstymo diagrama 50
Paveikslėlis Nr. 34. Įrankio pagrindinis langas 51
Paveikslėlis Nr. 35. OWL schema 52
Paveikslėlis Nr. 36. RDB lentelių struktūra, kuria išreiškiama hierarchija 54
Paveikslėlis Nr. 37. RDB lentelių, kurių klasės susijusios hierarchiškai, turinys 55
Paveikslėlis Nr. 38. RDB lentelių struktūra, kuria išreiškiamos funkcinės objektų savybės 55
Paveikslėlis Nr. 39. RDB lentelių struktūra, kuria išreiškiamos nefunkcinės objektų savybės 56
Paveikslėlis Nr. 40. RDB lentelių, kurios išreiškia nefunkcines objektų savybes, turinys 56
Paveikslėlis Nr. 41. RDB lentelių struktūra, kuria išreiškiamos duomenų tipų savybės 56
Paveikslėlis Nr. 42. RDB lentelių struktūra, kurioje saugojami metaduomenys 57
Paveikslėlis Nr. 43. RDB metaduomenų lentelių turinys 57
Paveikslėlis Nr. 44. Delegation OWL ontologijos schema 58
Paveikslėlis Nr. 45. Identifies ir isIdentifiedBy ryšiai po transformacijos 59
Paveikslėlis Nr. 46. Authorizes ir isAuthorizedBy ryšiai po transformacijos 59
Paveikslėlis Nr. 47. Naudojama Statinė ontologijos klasių struktūra 61
Paveikslėlis Nr. 48. Protege SPARQL ir Oracle SQL palyginimas 62
Paveikslėlis Nr. 49. Protege SPARQL su duomenų baze palyginimas su kitais būdais 63

Lentelių sąrašas

Lentelė Nr. 1. Transformavimo algoritmų palyginimas 16
Lentelė Nr. 2. Veiklos įvykių sąrašas 29
Lentelė Nr. 3. OWL apribojimai 53
Lentelė Nr. 4. Naudojamos paieškos užklausos, kurių rezultatai sutampa 61
Lentelė Nr. 5. Terminų ir santrumpų žodynas 67
Lentelė Nr. 6. SPARQL užklausų vykdymo rezultatai 68
Lentelė Nr. 7. SQL užklausų vykdymo rezultatai 69

Summary

Ontology descriptions are typically used in Semantic Web/Web2.0, but nowadays they find more and more adaptability in everyday Information Systems. Well-formed ontology must have correct syntax and unambiguous machine-understandable interpretation, so it is capable to clearly defining fundamental concepts and relationships of the problem domain.

Ontologies are increasingly used in many applications: business process and information integration, search and navigation. Such applications require scalability and performance, efficient storage and manipulation of large scale ontological data. In such circumstances, storing ontologies in relational databases are becoming the relevant needs for Semantic Web and enterprises.

For ontology development, Semantic Web languages are dedicated: Resource Description Framework (RDF) and schema RDFS, and Web Ontology Language (OWL) that consists of three sublanguages – OWL Lite, OWL Description Logic (DL) and OWL Full. When ontology based systems are growing in scope and volume, reasoners of expert systems are becoming unsuitable.

In this work an algorithm which fully automatically transforms ontologies, represented in OWL, to RDB schemas is proposed. Some concepts, e.g. ontology classes and properties are mapped to relational tables, relations and attributes, other (constraints) are stored like metadata in special tables. Using both direct mapping and metadata, it is possible to obtain appropriate relational structures and not to lose the ontological data.

1. ĮVADAS

Ontologijų aprašymai yra dažniausiai naudojami semantiniame žiniatinklyje (Semantic Web/Web 2.0), tačiau pastaruoju metu jie randa vis daugiau ir daugiau pritaikymo kasdienėms informacijos sistemoms. Puikiai suformuota ontologija privalo turėti teisingą sintaksę ir nedviprasmišką mašinai suprantamą interpretaciją, tokiu būdu ji gali aiškiai apibrėžti fundamentalias sąvokas ir ryšius probleminėje srityje.

Ontologijos vis plačiau naudojamos įvairiuose taikymuose: verslo procesų ir informacijos integravime, paieškoje ir žvalgyme. Tokie taikymai reikalauja geros greitaveikos, efektyvaus saugojimo ir didelio mąsto ontologinių duomenų manipuliavimo. Kai ontologijomis paremtos sistemos auga tiek akiračiu, tiek apimtimi, specialistų sistemose naudojami samprotavimo varikliai tampa nebetinkami. Tokiomis aplinkybėmis, ontologijų saugojimas reliacinėse duomenų bazėse tampa būtinas semantiniame žiniatinklyje ir įmonėse.

Šiame darbe atsakoma į klausimą kokiu būdu OWL ontologijas galima efektyviai transformuoti į reliacinių duomenų bazių schemas.

Taigi, šio darbo tyrimo sritis yra OWL ontologijų saugojimas ir naudojimas taikant reliacinių duomenų bazių technologijas, o tyrimo objektas – OWL ontologijos struktūros transformavimo į RDB struktūrą procesas.

Šio darbo tikslas – padidinti didelių ontologijų užklausų vykdymo galimybes ir veikimo greitį, sukuriant įrankį, leidžiantį transformuoti ontologijas į reliacines duomenų bazes.

Darbo uždaviniai:

- Ištirti OWL ontologijų transformavimo į RDB schemas algoritmą.
- Suprojektuoti, realizuoti ir ištestuoti įrankį, kuris realizuotų transformavimo algoritmą Protégé aplinkoje.
- Atlikti eksperimentą ir įvertinti darbo rezultatus.

Tiriant OWL ontologijų specifiką, buvo remtasi literatūros šaltiniu [1] . Jame pateikiamas pilnas OWL kalbos aprašymas. Analizuojant egzistuojančius ontologijų saugojimo ir apdorojimo sprendimus buvo nagrinėjami [2] - [14] šaltiniai. Darbo tikslai ir uždavinių formuluotė susiję su [15] ir [16] šaltiniais, kuriuose išdėstytas OWL ontologijų transformavimo į RDB schemas algoritmas.

Šiame darbe pateikiamas algoritmas, kuris pilnai transformuojama ontologijas, kurios yra atvaizduotos OWL, į reliacinių duomenų bazių schemas. Vienos sąvokos (pvz., ontologijų klasės ir savybės) yra išsaugojamos kaip reliacinės lentelės, ryšiai ir atributai, o kitos yra saugojamos (pvz., apribojimai) yra išsaugojami kaip metaduomenys specialiose lentelėse.

Tokiu būdu panaudojant tiek tiesioginį saugojimą, tiek metaduomenis, tampa įmanoma sudaryti atitinkamas reliacines struktūras neprarandant ontologinių duomenų.
Realizuotas įrankis, kuris šį algoritmą įgyvendina Protégé sistemoje kaip įskiepis, todėl jo teikiama nauda galės pasinaudoti projektuotojai. Šis įrankis atlieka visą algoritmą automatiškai, todėl projektuotojas, sukūręs OWL kalba aprašytą ontologiją, iškarto galės ją transformuoti į reliacinės duomenų bazės schemą – sukurti veikiančioje DBVS arba išsaugoti schemos kūrimo skripte.
Atliktas eksperimentas parodė, kad sukurtu įrankiu transformavus OWL ontologiją į RDB schemą ontologijų užklausos vykdomos efektyviau, nei nenaudojant esamą Protégé sistemos saugojamo būdą vienos lentelės RDB schemoje arba faile ir informacijos paieškos šiose priemonėse algoritmą. Taip pat taikant RDB galima efektyviai vykdyti užklausas didelėse ontologijose, kurios netelpa į užklausų procesoriaus atmintį.
Darbo struktūra:

- Antrame skyriuje atlikta OWL ontologijų kalbos ir OWL vaizdavimo į reliacines duomenų bazes analizė. Pateikti ontologijų ir OWL apibrėžimai, OWL kalbos specifika. Apžvelgti egzistuojantys sprendimai valdant dideles ontologijas ir jas atvaizduojant į reliacines duomenų bazes.
- Trečiame skyriuje pateiktas tiriamas algoritmas. Pateikiama OWL ontologijų ir reliacinių duomenų bazių apibendrinta statinė struktūra metamodeliais ir atliekamos transformacijos pagal ją. Veiklos schemomis pateikiamas algoritmo modelis.
- Ketvirtame skyriuje pateikiamas transformavimo algoritmą realizuojančio įrankio projektas. Suformuluoti reikalavimai įrankiui, suprojektuota jo architektūra panaudojant statinius ir dinaminius vaizdus.
- Penktame skyriuje pateikta įrankio realizacija, transformavimo algoritmo darbo pavyzdys ir eksperimentas.
- Šeštame skyriuje suformuluotos darbo išvados.

2. ONTOLOGIJŲ VAIZDAVIMO RELIACINĖSE DUOMENŲ BAZĖSE METODŲ ANALIZĖ

2.1. OWL kalbos analizė

OWL ontologijos kalba [1] buvo sukurta naudoti tokiuose taikymuose, kurie turi dirbti su informacijos turiniu, o ne vien tik jį pavaizduoti žmonėms. Ontologija yra vadinamas terminų ir jų tarpusavio ryšių pateikimas. OWL palengvina mašininį turinio interpretavimą palyginus su XML, RDF ar RDF-S, nes praplečia žodyną ir formalią semantiką. OWL taip pat daug sukauptos patirties įtraukė iš DAML+OIL ontologijos kalbos kūrimo ir taikymo.

2.1.1. Trys OWL kalbos poaibiai

OWL kalba yra išskaidyta į tris poaibius: OWL Lite, OWL DL ir OWL Full, kurių kiekviena turi vis didesnę išraiškų laisvę.

- OWL Lite yra naudojama tokioms reikmėms, kaip klasifikavimo hierarchijos ir paprasti apribojimai. Pavyzdžiui, nors ji palaiko kardinalumo ribojimus, bet suteikia jiems 0 arba 1 reikšmę. Dėl to šiai kalbos rūšiai yra žymiai lengviau kurti įrankius nei kitoms, ir ji turi mažesnį formalų sudėtingumą.
- OWL DL suteikia maksimalų ekspresyvumą, išlaikant skaičiavimų baigtumą (užtikrinama, kad visos išvados yra garantuotai suskaičiuojamos) ir sprendimo baigtumą (visi skaičiavimai bus atlikti per baigtinį laiką). OWL DL gali būti naudojami visi OWL kalbos dariniai, bet juos galima naudoti su kai kuriais ribojimais (pavyzdžiui, nors klasė gali būti kelių klasių poklasė, bet klasė negali būti kitos klasės atskiras atvejis). OWL DL pavadinimas išplaukia iš jo suderinamumo su apibūdinimo logika (angl. description logics) – tyrinėjimų srities, kuri studijavo logiką nuo pat formalaus OWL įkūrimo.
- OWL Full suteikia maksimalų ekspresyvumą, bet nesuteikia garantijų dėl skaičiavimų baigtumo. Deja, panašu, kad jokia samprotaujanti programinė įranga galėtų palaikyti pilną pagrindimą kiekvienai OWL Full ypatybei.

Kiekviena kalbos rūšis yra ankstesnės praplėtimas, todėl paprastesnėje kalbos rūšyje aprašyta ontologija taip pat bus galiojanti sudėtingesnėje.

Šiuo metu OWL Full realizacijų nėra, dėl to palaikymas ir aiškinimas yra menkai išplėtotas palyginus su OWL DL.

2.1.2. Kalbos dariniai ir sąvokos

Visą kalbos pagrindą sudaro OWL Lite, kurioje yra sukaupta daugelis kalbos darinių, nors ir daugelis jų turi griežtus ribojimus.

- Iš RDF Schema paveldėtos savybės:

  Class (Thing, Nothing) – tai yra grupė individų, kurie priklauso būti kartu, nes turi kokią nors bendrą savybę

  rdfs:subClassOf – nurodo hierarchinį ryšį, kuris reiškia vienos klasės buvimą kitos poklase

  rdf:Property – savybė nurodo ryšius tarp individų arba jų duomenų reikšmių

  rdfs:subPropertyOf – nurodo hierarchinį ryšį, kuris reiškia vienos savybės buvimą kitos posavybe

  rdfs:domain – nurodo ribojimą, kuriems individams savybė gali būti taikoma

  rdfs:range – nurodo ribojimą, kuriuos individus savybė gali turėti kaip reikšmę

  Individual – individai yra klasių egzemplioriai

- Lygybės:

  equivalentClass – ekvivalenčios klasės, kurios turi tuos pačius egzempliorius

  equivalentProperty – ekvivalenčios savybės, kurios tuos pačius individus susieja tarpusavyje

  sameAs – nurodo, kad du individai yra vienodi

  differentFrom – nurodo, kad individas skiriasi nuo kito

  AllDifferent – nurodo kiek gali būti visiškai skirtingų individų

- Savybių charakteristikos:

  inverseOf – viena savybė yra deklaruojama kaip priešinga kitai

  TransitiveProperty – nurodo, kad savybės yra tranzityvios

  SymmetricProperty – nurodo, kad savybės yra simetriškos

  FunctionalProperty – nurodo, kad savybė gali turėti tik vieną unikalią reikšmę kiekvienam individui

  InverseFunctionalProperty – nurodo priešingą funkcinę savybę

- Savybių ribojimai:

  allValuesFrom – nurodo, kad savybės turi visas klasės reikšmes

  someValuesFrom - nurodo, kad savybės apima dalį klasės reikšmių

- Ribotas kardinalumas:

  minCardinality – nurodo kiek mažiausiai kiekvienas egzempliorius turės susietų egzempliorių

  maxCardinality – nurodo kiek daugiausiai kiekvienas egzempliorius turės susietų egzempliorių

  cardinality – nurodomas, kai ir minCardinality, ir maxCardinality reikšmės sutampa

2.2. OWL vaizdavimo į reliacines duomenų bazes metodų analizė

2.2.1. Ontologijų taikymas panaudojant duomenų bazes

OWL kalba yra vis plačiau ir plačiau naudojama semantiniame tinkle aprašant ontologijas. Deja, kol kas nėra pakankamai įrankių, kurie leistų jomis manipuliuoti, saugoti ir ieškoti. Kai semantinio tinklo taikymai pasiekia ontologijų dydį, kai jas sudaro milijonai egzempliorių (pavyzdžiui, biologijoje), aprašomosios logikos įrankiai (palaikantys OWL-DL ontologijas) pradeda susidurti su greitaveikos problemomis tvarkyti jas. Visų pirma, daugelis įrankių ontologijos aiškinimąsi atlieka naudodami pagrindinę kompiuterio atmintį, o algoritmai nėra pritaikyti dideliems duomenų mastams. Taip yra dėl to, kad sudarant algoritmus didžiausias dėmesys buvo kreipiamas į ontologijos struktūros aiškinimosi optimizavimą, bet kai reikia aiškintis ją su milijonais egzempliorių arba ieškoti tarp jų informacijos, atsiranda sunkumų. Ir tai visuomet yra tik konjunktyvios užklausos. Logiška manyti, kad kuo toliau – tuo labiau semantiniame žiniatinklyje įgys didesnę reikšmę paieškos optimizavimas egzempliorių lygyje, ir dėl to reikia optimizuoti šią sritį. Ne tik ištobulinti greitaveiką, bet ir suteikti galimybes vykdyti sudėtingesnes užklausas. Kai užklausos tampa sudėtingos – joms užrašyti būtina kalba, kuri taip pat galėtų remtis ir aprašymo logika (DL).

Todėl buvo pasiūlytas sprendimas ECQ (Extended Conjunctive Queries) [2] . Šiuo būdu parašytas užklausas, panaudojant trijų žingsnių algoritmą, galima jas paversti į SQL užklausas, kurios išpildytų tiek aprašymo logikos galvojimą, tiek egzempliorių paiešką – iš pradžių sudaromas pagrindinis SELECT sakinys, kuris įgyvendina paiešką ontologijos struktūros, tuomet jam pridedamos ribojimų sąlygos (konstantos), ir galiausiai nustatomi egzempliorių paieškos WHERE sąlygos.

Nors ECQ kalba yra formali ir skirta aprašomosios logikos tyrėjams, o be to, ji yra vis dar kuriama ir tobulinama, galima daug architektūrinių sprendimų panaudoti sudarant algoritmą kaip egzistuojanti ontologija gali būti išsaugota duomenų bazėje, išsaugant jos struktūrą ir duomenis ir kaip vėliau ieškoti joje duomenų.

2.2.2. Ontologijų transformavimas į koncepcinius duomenų modelius

Koncepciniai duomenų modeliai informacijos sistemose yra naudojami suprasti taikymo sritį. Kadangi ontologijų meta modeliai turi sukaupę daug taikymo srities informacijos, automatinis ar pusiau automatinis jų pavertimas koncepciniais duomenų modeliais turėtų didelę naudą siekiant suprasti taikymo sritį.

Šaltinyje [8] aptariamas būdas, kaip ontologiją paversti ER modeliu. Tai gali būti padaryta aprašant ontologiją matematiškai. Čia geriausiai tinka grafų formalizmas. Tokiu pavidalu ji būtų aprašyta taip:

1. Koncepcijų sąrašas – grafo viršūnės.
2. Koncepcijų ryšiai – grafo orientuotos briaunos.
3. Egzempliorių, kurie priskirti koncepcijoms – duomenų įrašai, kurie priskiriami arba pačioms koncepcijoms, arba jų ryšiams.

Ontologijos pavertimą ER modeliu galima vaizduoti kaip jų grafų transformavimą:

$$G_O(\text{OWLElement}) \rightarrow G_{ER}(\text{ERElement})$$

Elementariosios transformacijos yra:

1. OWL ontologijos elementas yra keičiamas į ER modelio elementą.
2. OWL klasės yra paverčiamos ER esybėmis.
3. OWL duomenų tipo savybės yra paverčiamos atributais.
4. OWL objektų savybės yra paverčiamos sąryšius.
5. OWL duomenų ribos yra paverčiamos atomine sritimi, t.y. ribojimu.

Kitų elementų neįmanoma paversti tiesiogiai. Kadangi pagal ER modelį yra sudaromos reliacinės duomenų bazės, šį būdą galima laikyti vienu iš būdų OWL ontologijas transformuoti į reliacines duomenų bazių schemas.

2.2.3. Duomenų bazės struktūros ir jos duomenų vaizdavimas ontologijoje

Priešingas procesas ontologijų vaizdavimui duomenų bazėse yra ontologijų kūrimas pagal reliacines duomenų bazes [7] . Daugeliu atveju egzistuojančioje informacinėse sistemose duomenys yra saugojami reliacinėse duomenų bazėse ir norint, kad ontologijų pagrindu veikiančių pasaulinio tinklo taikymų pradėtų veikti pagal esamus duomenys, reikia kažkokiu būdu tuos duomenis vaizduoti ontologijose.

Čia pagrindinė problema yra kokiu būdu galima būtų susieti duomenų bazėje esantį egzempliorių su OWL ontologijoje esančiu. Ir svarstoma pagrindinė mintis yra rasti būdą, kaip SQL užklausą vaizduoti kaip RDF/OWL XML šabloną.

Tai yra sudėtingumu palyginus paprastesnė operacija, nei OWL vaizdavimas duomenų bazėje, ir jai reikalingi trys žingsniai – SQL užklausos įvykdymas, šablono užpildymas gautais duomenimis ir gautų RDF/OWL duomenų išsaugojimas. Jau yra sukurta įrankių, kurie jas atlieka kiekvienas vis su didesne greitaveika. Deja, pilnas procesas lieka neautomatizuotas – šabloną šiaip ar taip turi paruošti žmogus.

Straipsnyje [14] pastebima, kad reliacinės duomenų bazės ne visada būna tinkamos vaizdavimui ontologijoje – jose gali būti prarasta nemažai semantinės dalykinės srities informacijos, jos yra optimizuotos ir de-normalizuotos siekiant geresnės greitaveikos, dažnai plečiamos tik pradedančiųjų duomenų bazių specialistų, būna sunku suprasti atributų prasmę matant tik jų vardus. Dėl to būtina analizuoti ne tik pačia reliacinės duomenų bazės schema, o ir taikomąsias programas, kurios ja naudojasi, pavyzdžiui stipriai pagelbėja HTML puslapių struktūros analizė, jeigu sistema yra sukurta šios technologijos pagrindu.

2.2.4. Pagrindiniai skirtumai tarp OWL ir reliacinių duomenų bazių

Pagrindinis skirtumas tarp OWL ontologijų ir duomenų bazių yra tas [3] , kad ontologijos palyginti su duomenų bazėmis yra nepilnos, joms negalioja vientisumo ribojimai, jų modeliai gali būti begaliniai, o duomenų bazės yra pilnos – jose užtikrinamas duomenų vientisumas.

Duomenų bazėse iš pradžių yra analizuojami dalykinėje srityje egzistuojantys duomenų egzemplioriai ir sukuriama duomenų bazės struktūra, o ontologijose atvirkščiai – iš pradžių yra kuriama meta duomenų struktūra ir kai kurie struktūros elementai gali net neturėti jokių numatytų egzempliorių [4] . Dėl to ontologijai sukurtas koncepcinis duomenų modelis ne tik aprašo kokie duomenys bus naudojami sistemoje, bet tuo pačiu aprašo ir visą naudojamą taikymo sritį, dėl to vizualizavus pačią ontologiją, ji yra priimtinesnė vartotojui bandant suprasti kas ir kodėl yra naudojama. Visgi, ne visas ontologijų galimybes galima pavaizduoti koncepciniu duomenų modeliu, dėl to jis ne visada tinka kuriant sudėtingesnes ontologijas.

Taip yra dėl to, kad ontologijose vykdant operacijas yra priimama, kad duomenys gali būti nepilni taikymo srities atžvilgiu, kai tuo metu duomenų bazėse priimama kad duomenys pilnai aprašo tai ką vaizduoja. Pavyzdžiui, jeigu ontologijoje įdedame žmogų ir nurodome, kad jis parašė knygą, automatiškai po struktūros analizės nustatoma, kad jis iš tikro yra autorius (nors tokia informacija nebuvo suteikta). Taip pat, ontologijose kai kurie ryšiai nėra griežti. Pavyzdžiui, jeigu įvedama knyga, kuri privalo turėti autorių, pakeitimas yra

padaromas net jei autorius nenurodytas, nes priimama, kad autorius egzistuoja, bet šiuo metu nėra žinomas.

2.2.5. Reliacinių duomenų bazių išplėtimas ontologijų palaikymui

Duomenų bazėse galima saugoti ontologijas keletu būdų [9] :

- Horizontali duomenų bazė: naudojama tik viena universali lentelė duomenų bazėje. Kiekvienas egzempliorius saugojamas viename įraše. Nors toks modelis yra nesudėtingas, jo trūkumai aiškūs – didelis laukų kiekis, savybių reikšmės ribotos, duomenų išsisklaidymas ir pan.
- Vertikali lentelė: čia taip pat naudojama tik viena lentelė, o kiekvienas įrašas atitinka RDF trejetą. Deja, šis atvejis taip pat nėra be trūkumų – kiekviena užklausa ieškos duomenų bazėje pakankamai ilgai dėl brangių sujungimų.
- Horizontali klasė: kiekvienai ontologijos klasei sukuriama lentelė ir egzemplioriai surašomi į jas. Tai yra labai panašus metodas, kai kiekvienai esybei yra priskiriama lentelė, kai yra kuriama duomenų bazė.
- Lentelė kiekvienai savybei: duomenų bazių projektavime tokia alternatyva taip pat žinoma kaip išskaidymo saugojimo modelis. Deja, kaip ir vertikaliame būde, užklausos tampa labai brangios, kai reikia išrinkti pilnus kurios nors klasės egzempliorius.
- Hibridinis būdas: įtraukiama horizontalios klasės ir lentelių kiekvienai savybei metodika. Tokiu būdu kiekviena lentelė atitinka arba klasės, arba savybės aprašymą ontologijoje. Toks būdas tinkamas, kai ontologiją sudaro vidutinis klasių kiekis (daugiausiai kelių šimtų).

Todėl sukurtas naujas būdas – klasių hierarchija yra saugojama vaizde (angl. view). Tai yra tam tikra užklausos forma duomenų bazėse. Čia klasės vaizdas yra aprašomas rekursyviai. Tai yra sąjunga klasės lentelės ir visų jos tiesioginių poklasių vaizdų. Tokiu būdu, klasės vaizde bus tiek klasės egzemplioriai, tiek poklasių egzemplioriai. Savybių sąryšiai gali būti aprašomi panašiu būdu.

Panaudojus tokią paprastą duomenų bazę kaip Microsoft Access ir FaCT samprotavimą, eksperimentai parodė, kad rezultatai buvo daug išsamesni kai kurioms užklausoms, o jų laikas buvo žymiai mažesnis ar visai nežymus palyginus su kitais būdais (kai egzempliorių kiekis yra apie milijoną).

Taigi, panaudojus dideliems duomenų kiekiams daugiau pritaikytus duomenų bazių produktus – greitaveikos rezultatai turėtų būti tinkami ir didesnėse ontologijose.

2.2.6. Universalios duomenų bazės schemos naudojimas

Vienas iš galimų OWL ontologijų vaizdavimo į duomenų bazes variantų yra naudojant universalią duomenų bazės schemą [6] . Tokiu būdu, vietoje įprastinio bandymo kiekvieną lentelę susieti su klase ar savybe, yra sukuriama fiksuota ir palyginus nesudėtinga schema, kuri nepriklauso nei nuo ontologijos struktūros, nei nuo atskirų egzempliorių.

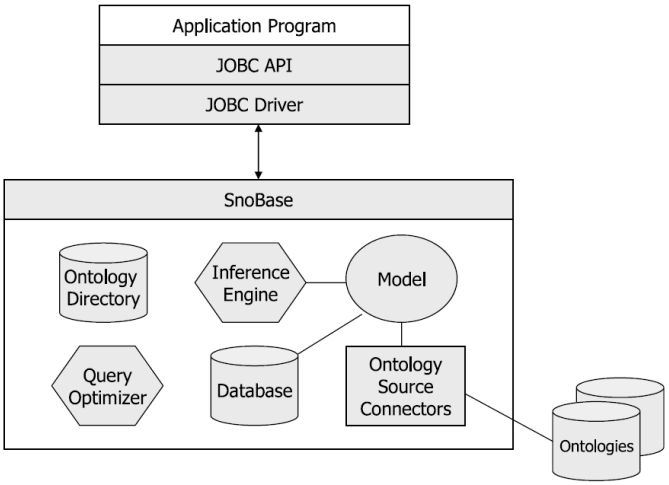
Tokios sistemos pagrindas ir schemos centras yra aprašomosios logikos elementas, su kuriuo susiejama bet kokia galimai reikalinga informacija – lentelė teiginiams, ekvivalentams, vaikams ir tėvams. Norint, kad tokiu būdu saugoma ontologija turėtų didelę greitaveiką (kai saugoma milijonai egzempliorių), visiems aprašomosios logikos elementams taip pat yra saugoma tarpinė informacija tam skirtoje lentelėje, kuri leidžia vėliau atliekant samprotavimą sumažinti atskirų SQL užklausų skaičių, sudarant sudėtingesnes užklausas vietoje jų.

Nors ir toks būdas pasižymi greitaveika kai reikia ieškoti duomenų, deja, šioje sistemoje yra vienas svarbus trūkumas – jeigu atsiranda struktūrinių ontologijos pasikeitimų, pavyzdžiui atsiradus naujam klasės lygiui, laukia ilgas ir sudėtingas visos struktūros atnaujinimo procesas.

Nors ši sistema negali būti naudojama visur – dėl savo architektūros turi būti pasverta ar ji yra tinkamas sprendimas, bet ją kuriant yra iškeltas labai svarbus teiginys – kad reikia atskirti samprotavimą nuo didelių duomenų kiekių valdymo, kad yra jau optimizuota duomenų bazėse vykdant tokias operacijas, kaip sąjungos ar susikirtimai.

2.2.7. Didelio masto informacinių sistemų ontologijų valdymas

Kadangi ontologijos vis dažniau yra taikomos įmonių masto informacinėse sistemose, yra reikalingas tiek vartotojui, tiek kūrėjui draugiškas jų panaudojimas, išlaikant greitaveiką ir užtikrintą darbą, kai duomenų yra labai daug. Todėl buvo sukurta SnoBase ontologijų valdymo sistema [10] , kurios pagrindas yra Java bei IBM DB2 technologijos.



Paveikslėlis Nr. 1. SnoBase architektūra

Paveikslėlis Nr. 1 parodo SnoBase architektūros pagrindus. Čia JOBC API yra Java kalbai pritaikyta sąsaja su JOBC Driver, kuris įgyvendina prieigą prie SnoBase ir yra ekvivalentus JDBC IBM DB2 tvarkyklei. Visos užklausos atliekamos OWL-QL kalba [11] , kuri yra panaši į SQL, tik pritaikyta ontologijoms.

Pereiti prie pilno ontologijų saugojimo duomenų bazėse nesiūloma, todėl pateikiamas tarpinis variantas – duomenys yra saugojami duomenų bazėje (Database elementas), o ontologijų struktūra yra saugojama joms pritaikytose saugyklose, ir yra pasiekiamos Ontology Source Connectors elementu.

2.2.8. Ontologijos pavertimas duomenų bazės schema

Šaltinyje [5] parodoma, kaip buvo konkreti ontologija pusiau automatiškai paversta į duomenų bazės schemą:

1. Paverčiama koncepcijos hierarchija: kiekviena koncepcija yra paverčiama lentele. Jei super-koncepcija yra parenkama – kuriamos dvi lenteles su tuo pačiu pirminiu raktu.
2. Paverčiamos duomenų tipų savybės: kiekviena savybė yra paverčiama lentelės atributu atitinkamoje lentelėje.
3. Paverčiamos objektų savybės: remiantis ribojimais, kurie nurodyti objekto savybėse, sukuriami ryšiai. Jei savybė nurodo 1:n ryšį, sukuriamas ryšys tarp lentelių panaudojant išorinį raktą. Jei savybė nurodo n:m ryšį, tarpinė lentelė yra kuriama, kurios laukai yra išoriniai raktai. Kai nurodomas 1:1 ryšys – atributai yra įdedami į koncepcijos lentelę, kuri vaizduoja srities savybę.
4. Egzemplioriai yra paverčiami duomenų eilutėmis, kurios įdedamos į naują duomenų bazę.

2.2.9. Ontologijų vaizdavimo transformavimas iš OWL į reliacinių duomenų bazių schemas

Reliacinių duomenų bazių panaudojimas vaizduojant ontologijas yra propaguojama dėl to, kad RDB yra vystomos ilgą laiką ir jų galimybės apdoroti didelius duomenų kiekius [13] buvo patikrintos ir nuolat optimizuojamos, o tokios savybės kaip plėtojamos automatizuoto atkūrimo ar optimizavimo galimybės suteikia stipresnį pagrindą panaudoti šias technologijas.

Straipsnyje [15] pateiktas ontologijų vaizdavimo transformavimo iš OWL į reliacinių duomenų bazių schemas algoritmas bus tiriamas ir plečiamas šiame darbe.

Algoritmas atlieka skirtingų OWL elementų transformaciją į RDB:

- Klasės vaizduojamos kaip reliacinės duomenų bazės lenteles.
- Objektų savybės vaizduojamos kaip ryšiai.
- Ribojimai vaizduojami kaip metaduomenys.

2.2.10. Transformavimo metodų palyginimas

Apžvelgtų transformavimo algoritmų santrauka pateikta lentelėje:

Lentelė Nr. 1. Transformavimo algoritmų palyginimas

Transformavimo algoritmas	Teigiamos savybės	Neigiamos savybės
Horizontali lentelė	Nesudėtinga struktūra.	Didelis laukų kiekis. Savybių reikšmės ribotos. Duomenų išsisklaidymas. Neišnaudojamos RDB savybės.
Vertikali lentelė	Nesudėtinga struktūra	Užklausos, kurios naudoja sujungimus, yra labai lėtos Neišnaudojamos RDB savybės
Horizontali klasė	Išlaikomas struktūrinis panašumas - lentelės atitinka klases Panašus į duomenų bazės kūrimo pagal esybes metodą	Prarandama objektų savybių ir apribojimų informacija

Lentelė kiekvienai savybei	Išlaikomas struktūrinis panašumas - lentelės atitinka savybes	Užklausos, kurios naudoja numanomus duomenis, yra labai lėtos
Hibridinis metodas	Išlaikomas struktūrinis panašumas – lentelės atitinka klases ir savybes, sudaromi ryšiai	Jaučiamas sulėtėjimas pasiekus 200 klasių dydžio ontologiją
Vaizdų panaudojimas	Užklausos vyksta palyginus greitai pasiekus milijoną egzempliorių	Rekursyvus vaizdų naudojimas gali sukelti greitaveikos problemų, kai poklasių medis didėja ir į plotį, ir į gylį
Universali duomenų bazė	Universali struktūra Pakankamai greitas užklausų vykdymas	Sudėtinga atlikti struktūrinius pakeitimus po sukūrimo Negali būti panaudota absoliučiai visais atvejais
Klasių, savybių ir ribojimų transformavimo algoritmas	Išlaikomas struktūrinis panašumas, neprarandama informacija	

2.3. Analizės išvados

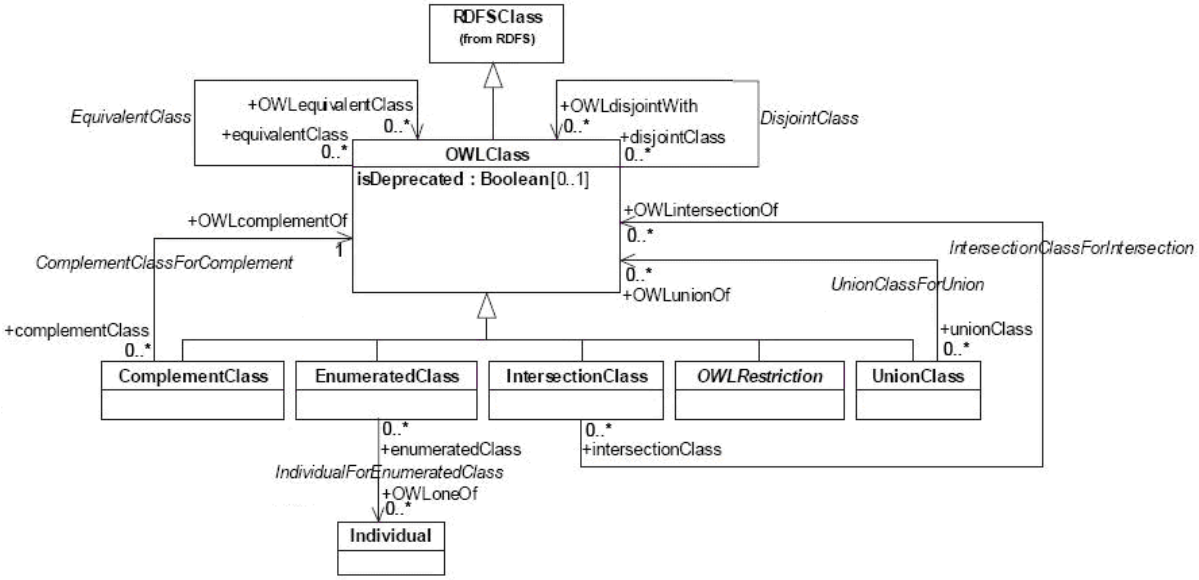
Literatūros analizė rodo, kad didelių ontologijų saugojimas yra problema, o DBVS panaudojimas leidžia saugoti, ieškoti ir manipuliuoti OWL ontologijų duomenimis efektyviau Esamų ontologijų saugojimo RDB metodų analizė parodė, kad jie turi trūkumų ir neišnaudoja visų RDB galimybių.

3. ONTOLOGIJŲ TRANSFORMAVIMO ALGORITMAS

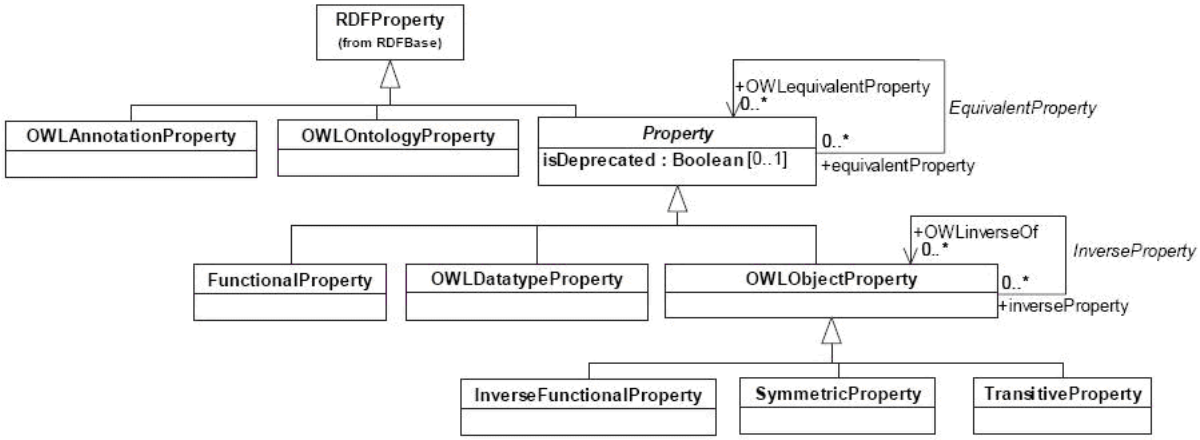
Šiame skyriuje aprašyti OWL ontologijų transformavimo į reliacinių duomenų bazių schemas algoritmo principinis modelis.

3.1. Algoritmo statinis vaizdas

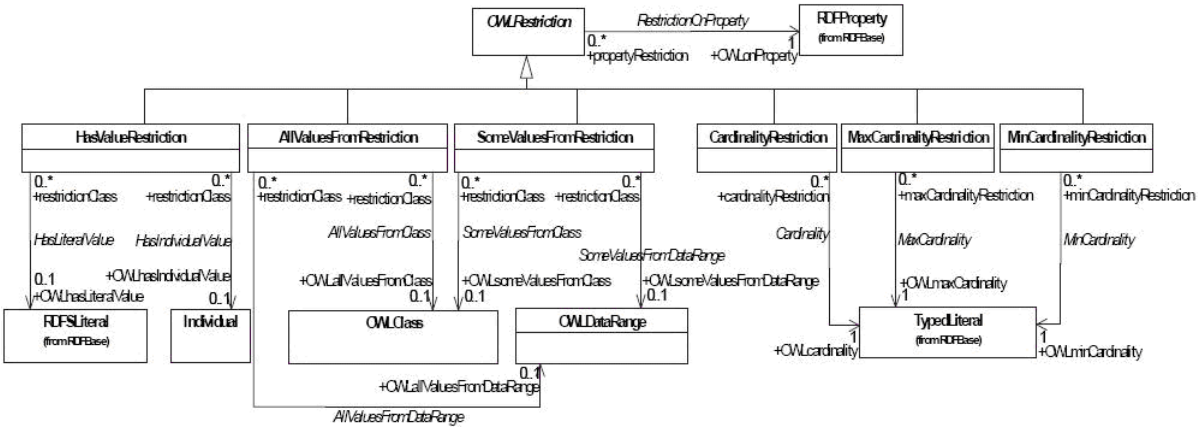
Ontologijos yra kuriamos daugybei dalykinių sričių, dėl to dalykinės srities modelis algoritmo ribose yra dinaminis. Tačiau visoms ontologijoms būdinga bendra struktūra [16] , kurią galima atvaizduoti šiais metamodeliais:



Paveikslėlis Nr. 2. Ontologijose taikomas klasių metamodelis

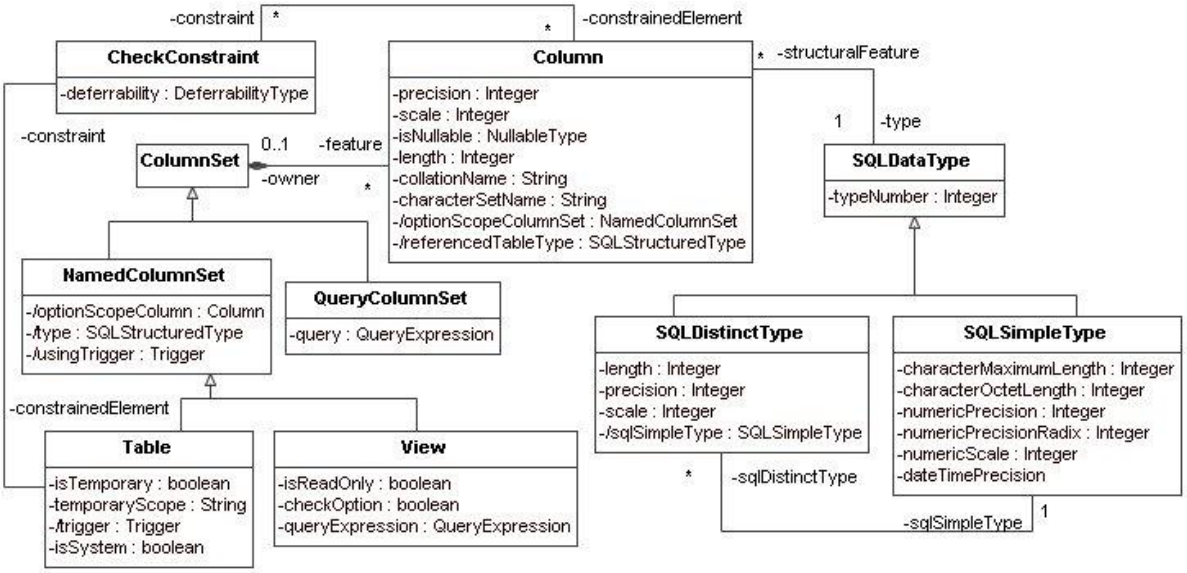


Paveikslėlis Nr. 3. Ontologijose taikomas savybių metamodelis

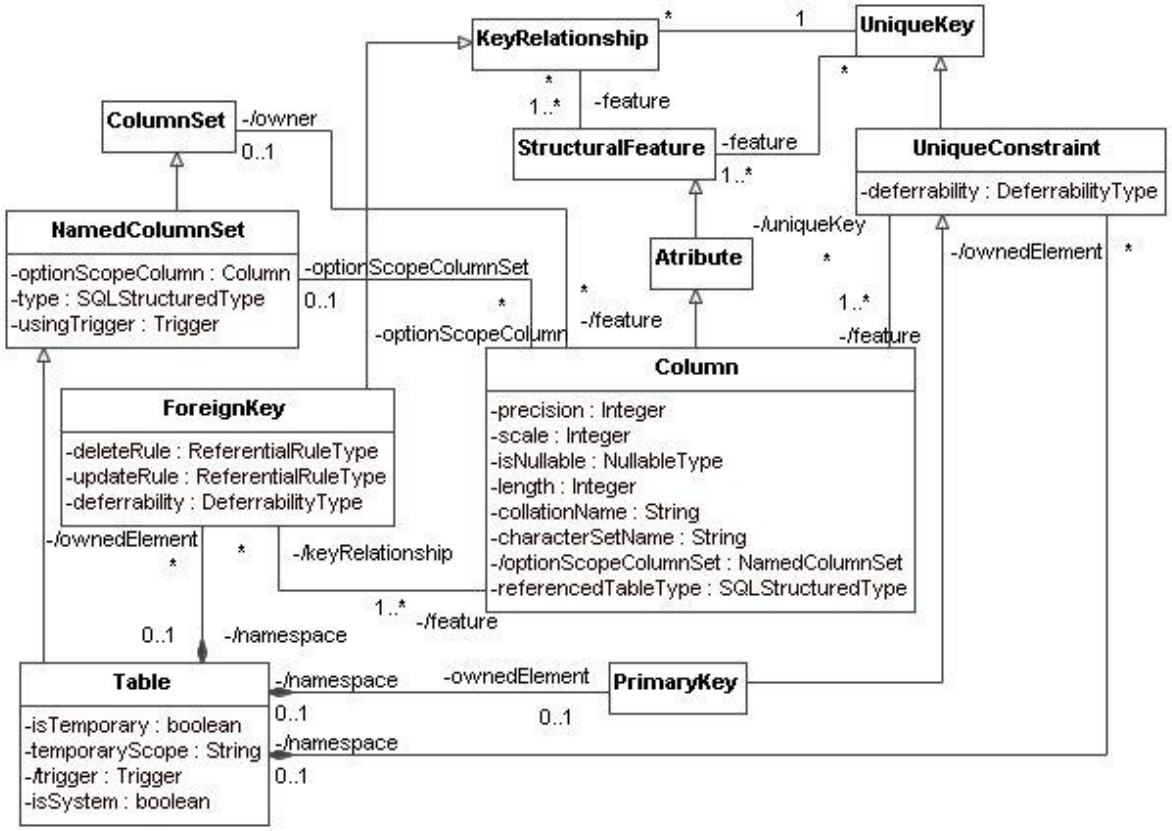


Paveikslėlis Nr. 4. Ontologijose taikomas apribojimų metamodelis

Reliacinėse duomenų bazėse taip pat būdinga bendra struktūra:



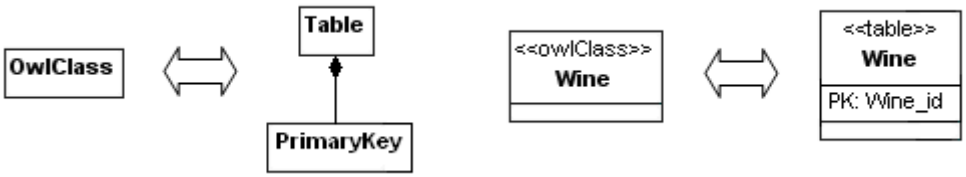
Paveikslėlis Nr. 5. Lentelių, atributų ir duomenų tipų metamodelis



Paveikslėlis Nr. 6. Unikalumo apribojimų ir išorinių raktų metamodelis

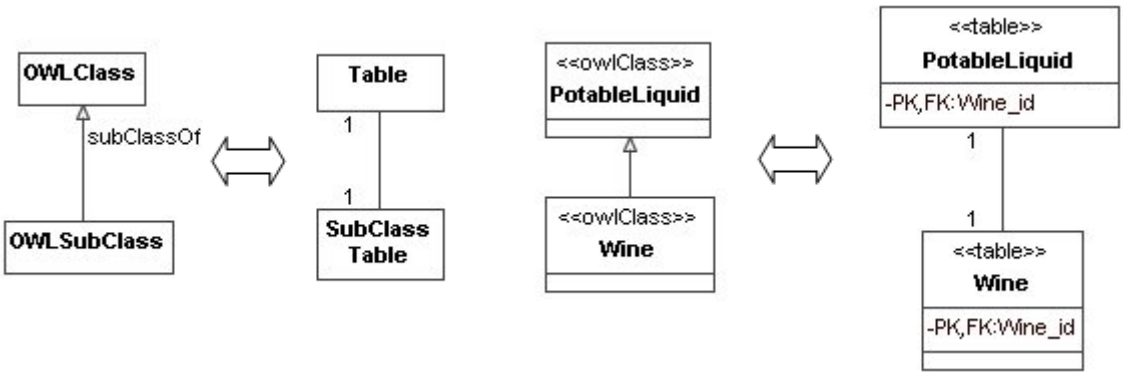
Atliekant transformaciją, išlaikomas semantinis ontologijos ir reliacinės duomenų bazės elementų panašumas.

Tokiu būdu OWL ontologijos klasės yra transformuojamos į lenteles duomenų bazėje (Paveikslėlis Nr. 7). Taip pat išlaikomas vienodas (arba panašus) pavadinimas, nes klasės ontologijose yra unikalios. Lentelės užpildomos klasių egzemplioriais. Kadangi klasių egzempliorių pavadinimai yra unikalūs, pirminiu raktu paskelbiamas vienas stulpelis, kuriame ir išsaugojami egzempliorių pavadinimai.



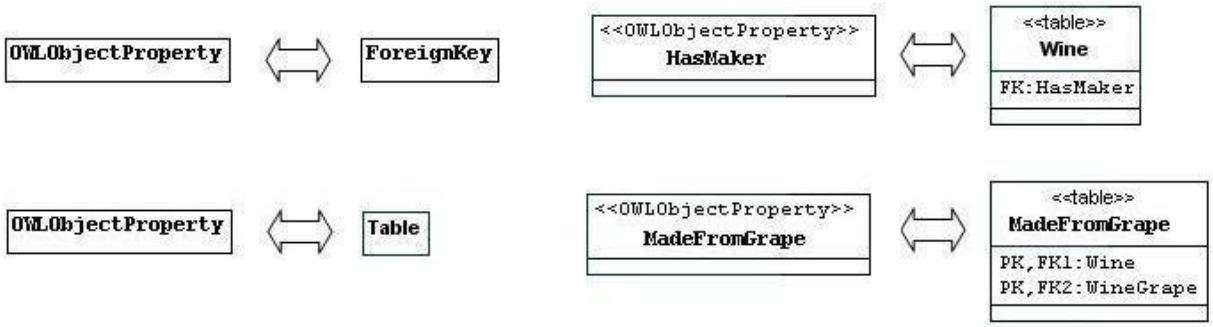
Paveikslėlis Nr. 7. Klasių transformacija ir pavyzdys

Klasių hierarchijai išreikšti naudojami 1:1 ryšiai iš poklasių lentelių pirminių raktų į aukštesnių klasių lentelių pirminius raktus, pakeičiant poklasių lentelių pirminio rakto stulpelio pavadinimą pagal aukštesnės klasės lentelės pirminio rakto stulpelio pavadinimą (Paveikslėlis Nr. 8).



Paveikslėlis Nr. 8. Poklasių transformacija ir pavyzdys

OWL objektų savybės yra paverčiamos į ryšius tarp klasių lentelių. Priklausomai nuo jų kardinalumo, naudojamos tarpinės lentelės su išoriniais raktais, kai kardinalumas daug su daug, arba kuriami stulpeliai, kai kardinalumas yra vienas su daug (Paveikslėlis Nr. 9).

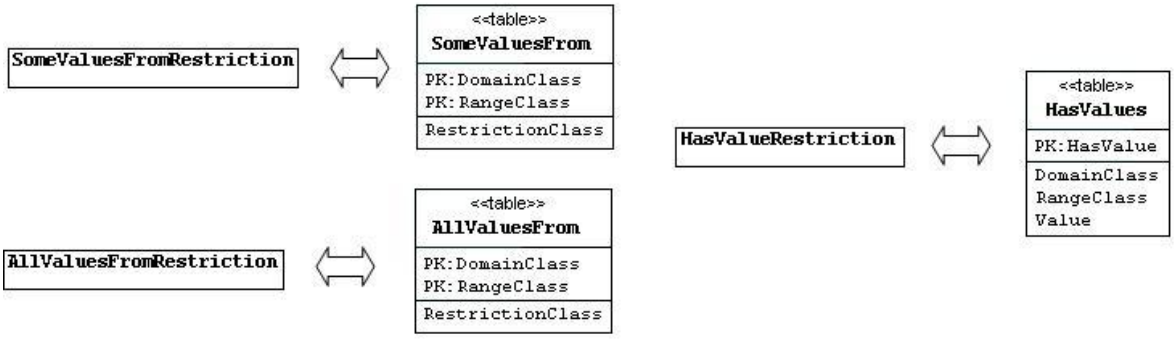


Paveikslėlis Nr. 9. Objektų savybių transformacija ir pavyzdys

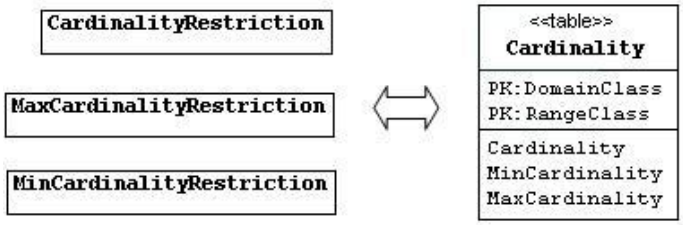
OWL duomenų tipų savybės yra paverčiamos naujais stulpeliai lentelėse, kurių klasėms jos taikomos, ir užpildomos atitinkama kiekvieno egzemplioriaus turima reikšme (Paveikslėlis Nr. 10).

Paveikslėlis Nr. 10. Duomenų tipų savybių transformacija ir pavyzdys

OWL apribojimai neturi struktūrinio atitikmens reliacinėse duomenų bazėse, todėl transformavimas atliekamas kuriant metaduomenų lenteles. Tokiu būdu sukuriamos trys lentelės SomeValuesFrom, AllValuesFrom ir HasValues (Paveikslėlis Nr. 11). Visi kardinalumo apribojimai (Cardinality, MinCardinality, MaxCardinality) išsaugojami Cardinality lentelėje (Paveikslėlis Nr. 12).



Paveikslėlis Nr. 11. Apribojimų transformacija



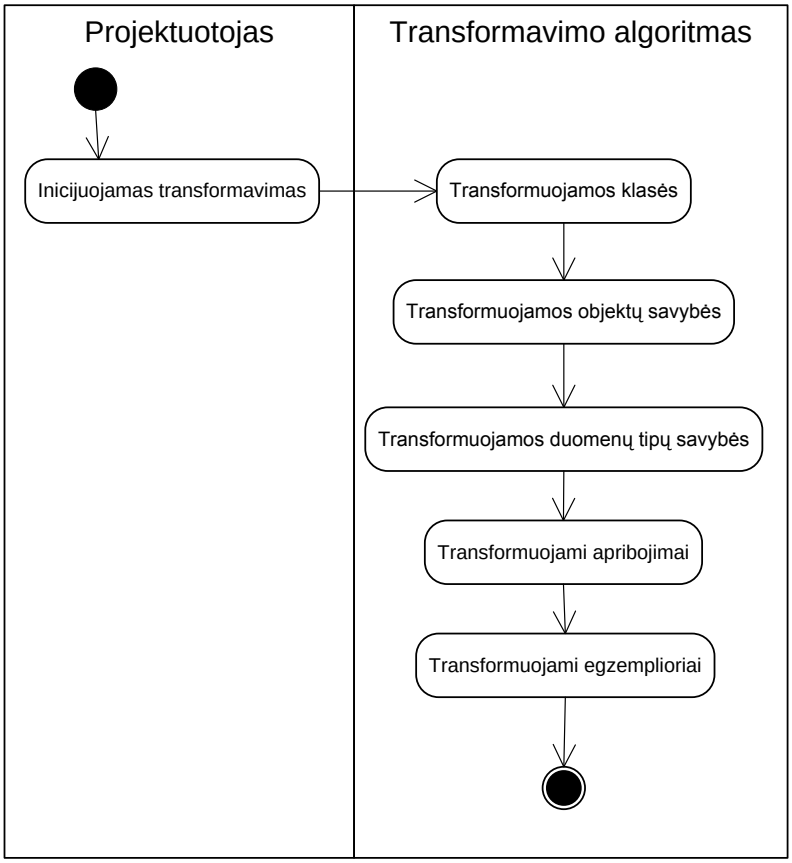
Paveikslėlis Nr. 12. Kardinalumo apribojimų transformacija

3.2. Algoritmo dinaminis vaizdas

3.2.1. Apibendrintas dinaminis vaizdas

OWL ontologijos transformavimas atliekamas nuosekliai vykdant veiksmus, skirtus skirtingiems elementams (Paveikslėlis Nr. 13).

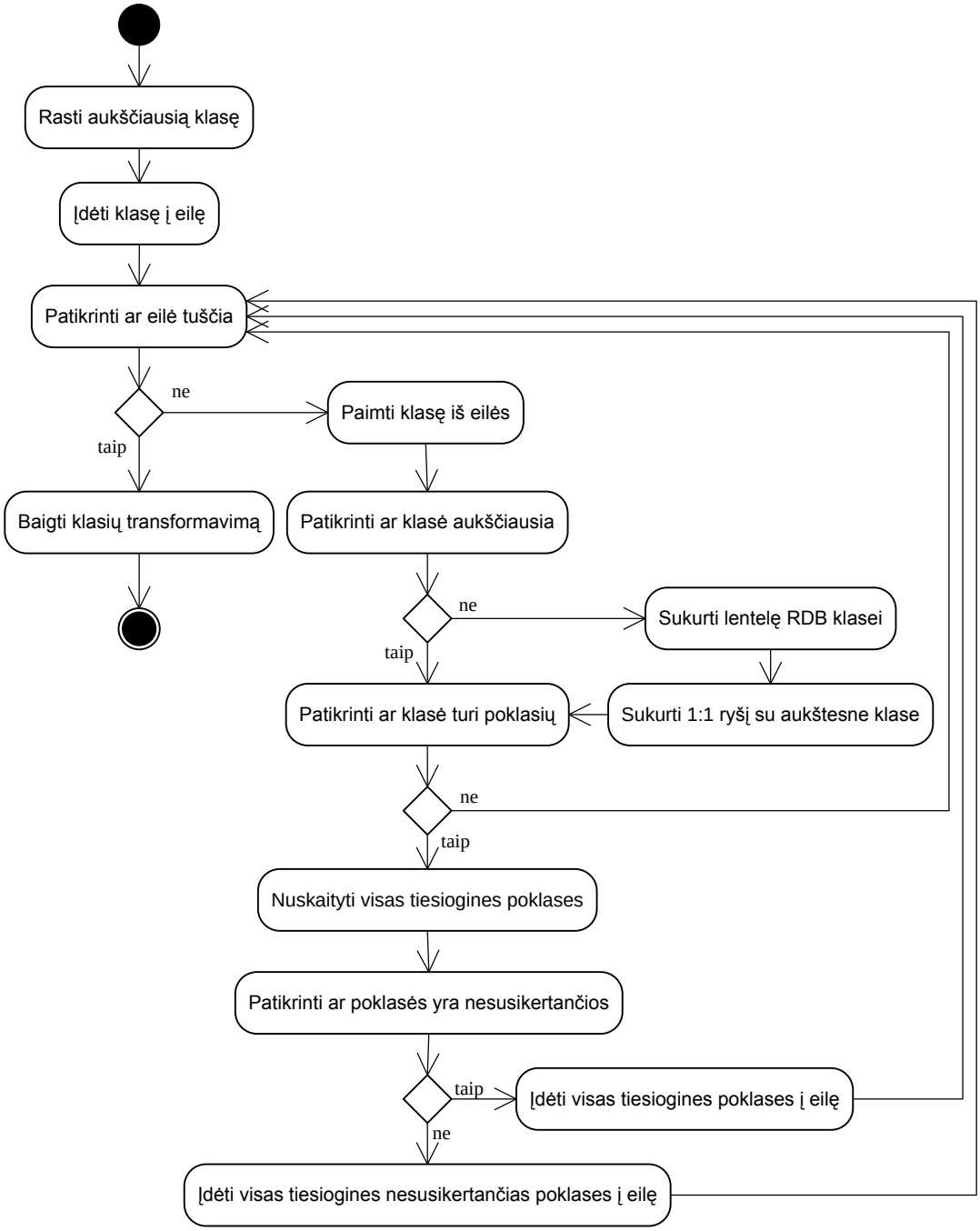
Iš pradžių transformuojamos klasės, kad būtų sudaryta klasių lentelių struktūrą, kurią galima plėsti ir užpildyti duomenimis tolesniuose žingsniuose. Tuomet sudaromi ryšiai transformuojant objektų savybes bei sukuriami duomenų stulpeliai transformuojant duomenų tipų savybes. Tuomet sukuriamos ir užpildomos metaduomenų lentelės, kuriose išsaugojama apribojimų informacija. Galiausiai visos lentelės užpildomos egzempliorių informacija.



Paveikslėlis Nr. 13. Transformavimo veiklos schema

3.2.2. Klasių transformavimo dinaminis vaizdas

Klasės yra transformuojamos panaudojant paieškos platyn algoritmą pagal hierarchiją (Paveikslėlis Nr. 14). Pradėjimas nuo aukščiausios klasės ir žemėjant po vieną lygį leidžia išlaikyti RDB kūrimo nuoseklumą ir išvengti lentelių ryšių analizės, nes kuriant ryšius iš poklasių lentelių yra žinoma, kad aukštesnių klasių lentelės jau sukurtos.



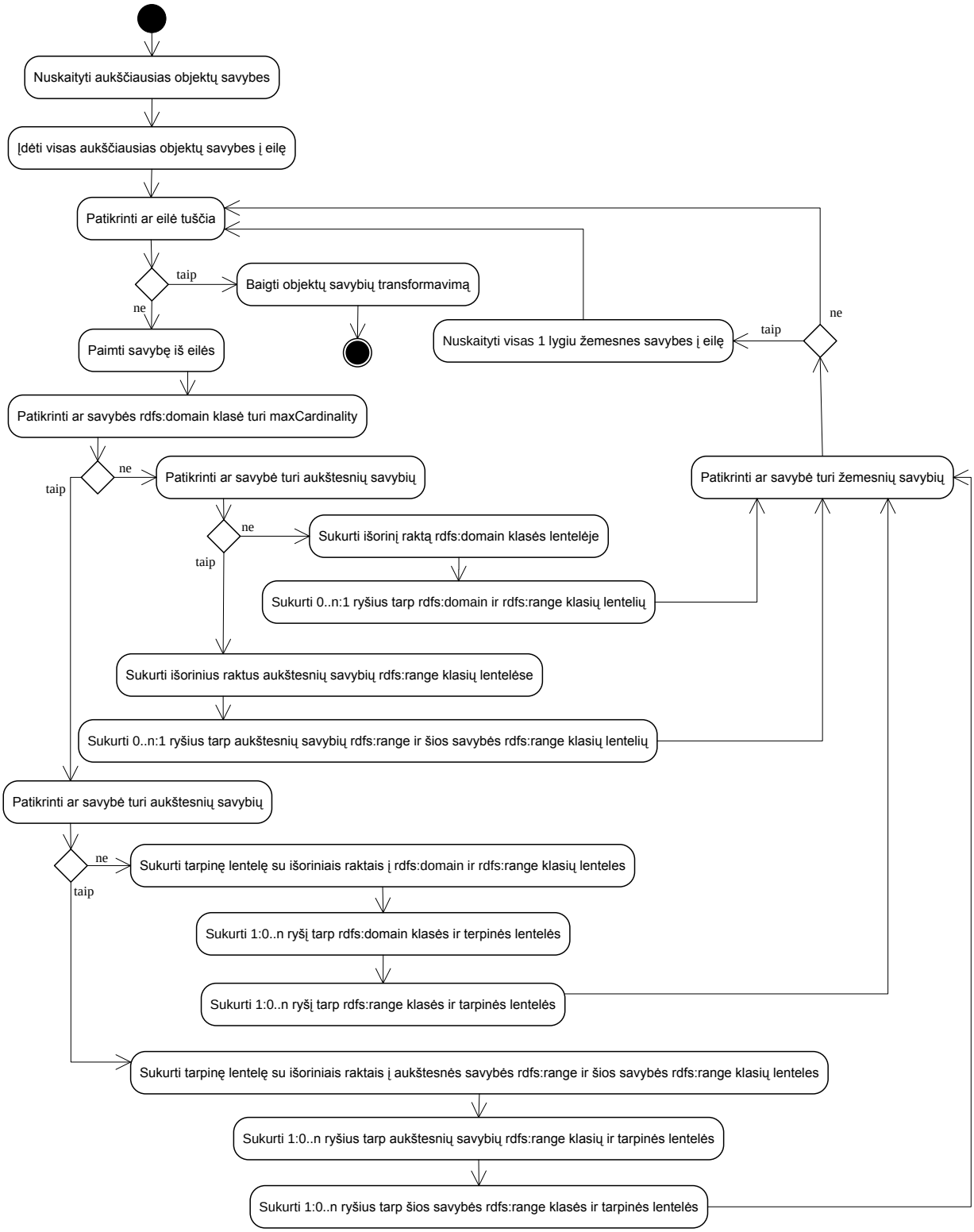
Paveikslėlis Nr. 14. Klasių transformavimo veiklos diagrama

3.2.3. Objektų savybių transformavimo dinaminis vaizdas

Objektų savybės yra transformuojamos panaudojant paieškos platyn algoritmą pagal hierarchiją (Paveikslėlis Nr. 15).

Priklausomai nuo kardinalumo apribojimų, kuriamos tarpinės lentelės realizuojant daug su daug kardinalumą arba papildomi laukai klasių lentelėse realizuojant vienas su daug kardinalumą.

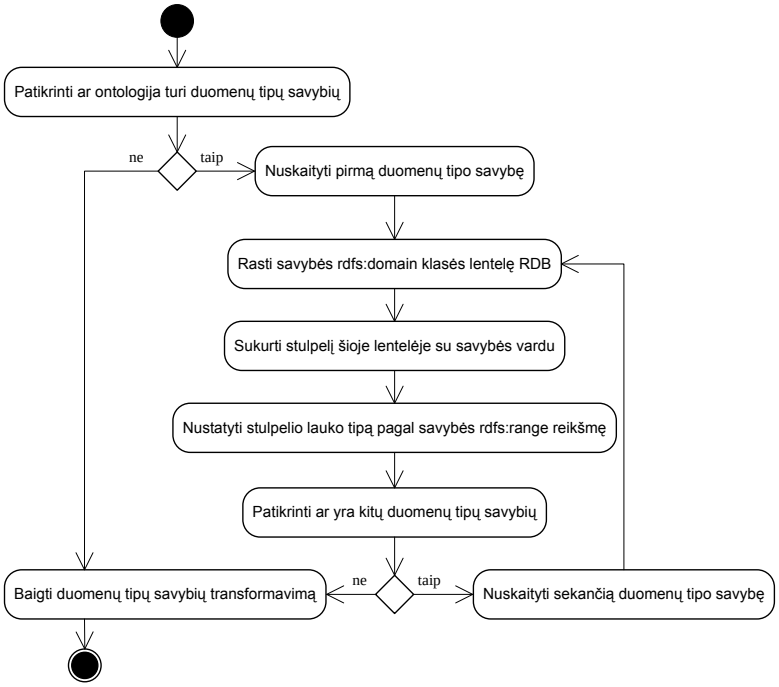
Jeigu savybė neturi aukštesnio lygio savybės, ryšys yra kuriamas tarp jos rdfs:domain ir rdfs:range klasių lentelių. Tačiau jeigu savybė yra žemesnio lygio – tuomet ryšiai yra kuriami tarp aukštesnės savybės rdfs:range ir žemesnės savybės rdfs:range klasių lentelių.



Paveikslėlis Nr. 15. Objektų savybių transformavimo veiklos diagrama

3.2.4. Duomenų tipų savybių transformavimo dinaminis vaizdas

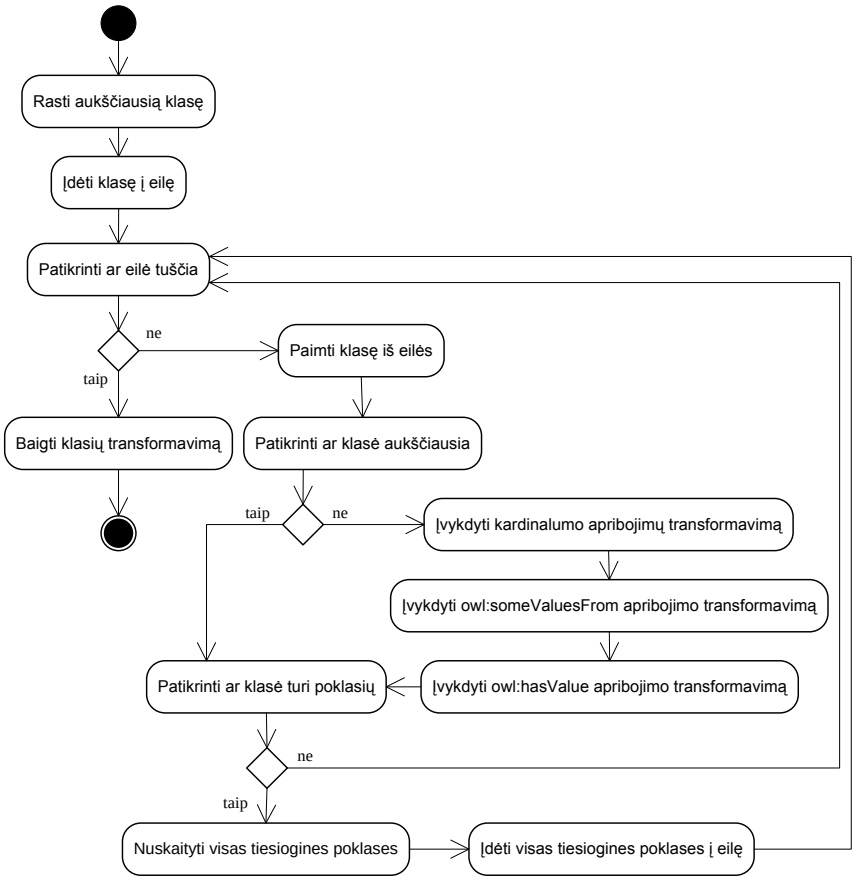
Duomenų tipų savybės transformuojamos be specialios tvarkos – pagal kiekvieną savybę kuriami stulpeliai rdfs:domain klasės lentelėje, jo tipą nustatant iš rdfs:range esančios reikšmės (Paveikslėlis Nr. 16).



Paveikslėlis Nr. 16. Duomenų tipų savybių transformavimo veiklos diagrama

3.2.5. Apribojimų transformavimo dinaminis vaizdas

Apribojimai yra transformuojami panaudojant klasių paieškos platyn algoritmą pagal hierarchiją (Paveikslėlis Nr. 17), nes jie apribojimus įveda pačioms klasėms.



Paveikslėlis Nr. 17. Apribojimų transformavimo veiklos diagrama

4. ONTOLOGIJŲ TRANSFORMAVIMO Į RDB PROJEKTAS

Šiame skyriuje aprašyti reikalavimai ir architektūra OWL ontologijų transformavimo į reliacinių duomenų bazių schemas algoritmui realizuoti kuriamam įrankiui.

4.1. Reikalavimų specifikacija

Protege sistemos paskirtis yra suteikti projektuotojui galimybę kurti ontologijas. Šio darbo metu kuriamo įrankio šiai sistemai paskirtis yra realizuoti tobulinamą transformavimo algoritmą, taip suteikiant galimybę jį tirti realioje aplinkoje, o taip pat suteikti kitiems projektuotojams galimybę transformuoti sukurtas ontologijas į reliacinių duomenų bazių schemas, taip išplečiant jų saugojimo ir pasiekimo galimybes ištobulintomis technologijomis ir metodais.

Protege yra išplėtimui pritaikyta sistema, dėl to ji yra puikiai tinka realizuoti tik sukurtą transformavimo algoritmą paliekant visą ontologijų valdymą pačiai sistemai.

Darbo tikslas yra tirti bei tobulinti transformavimo algoritmą. Tyrimui atlikti būtina sukurti transformavimo įrankį, kuris pagal aprašytus algoritmus atliktų OWL kalba aprašytos ontologijos transformavimą į reliacinės duomenų bazės schemą.

Įrankis yra kuriamas kaip Protege sistemos plėtinys, todėl šia sistema kuriama arba redaguojama OWL ontologija galės būti transformuojama į RDB schemą vartotojo požiūriu nesudėtingu būdu - įrankį bus galima pateikti kaip atviro kodo produktą, prieinamą visiems šia sistema besinaudojantiems projektuotojams.

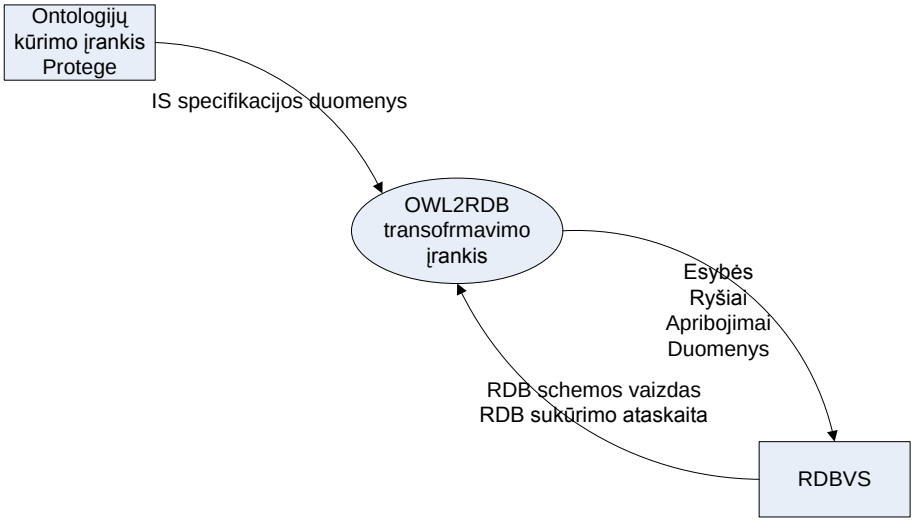
Pradinį transformavimo algoritmą pateikia užsakovas, todėl galutinis algoritmo variantas turi išlaikyti pateikto ideologiją. Įrankis turi būti Protege sistemos plėtinys.Įrankis bus Protege sistemos dalis, todėl su ja bendradarbiaus daugeliu aspektų – jo paties iškvietimo, konfigūravimo bei OWL ontologijos kaip pradinių duomenų gavimo. Taip pat įrankis bendradarbiaus su DBVS, kurioje kurs ar atnaujins reliacinės duomenų bazės schemą.

Protege sistemai plėtiniai yra kuriami remiantis Protege Programming Development Kit dokumentacija. Įrankio kūrimui bus naudojama Protege Core API bendradarbiavimui su pačia sistema ir Protege-OWL API sukurtos OWL ontologijos naudojimui.

Protege sistema yra nuolat kuriama ir tobulinama. Dėl to gali kisti Protege Core API bei Protege-OWL API specifikacijos.

4.1.1. Funkciniai reikalavimai

Pateikiamas transformavimo įrankio (ir algoritmo) veiklos kontekstas (Paveikslėlis Nr. 18).



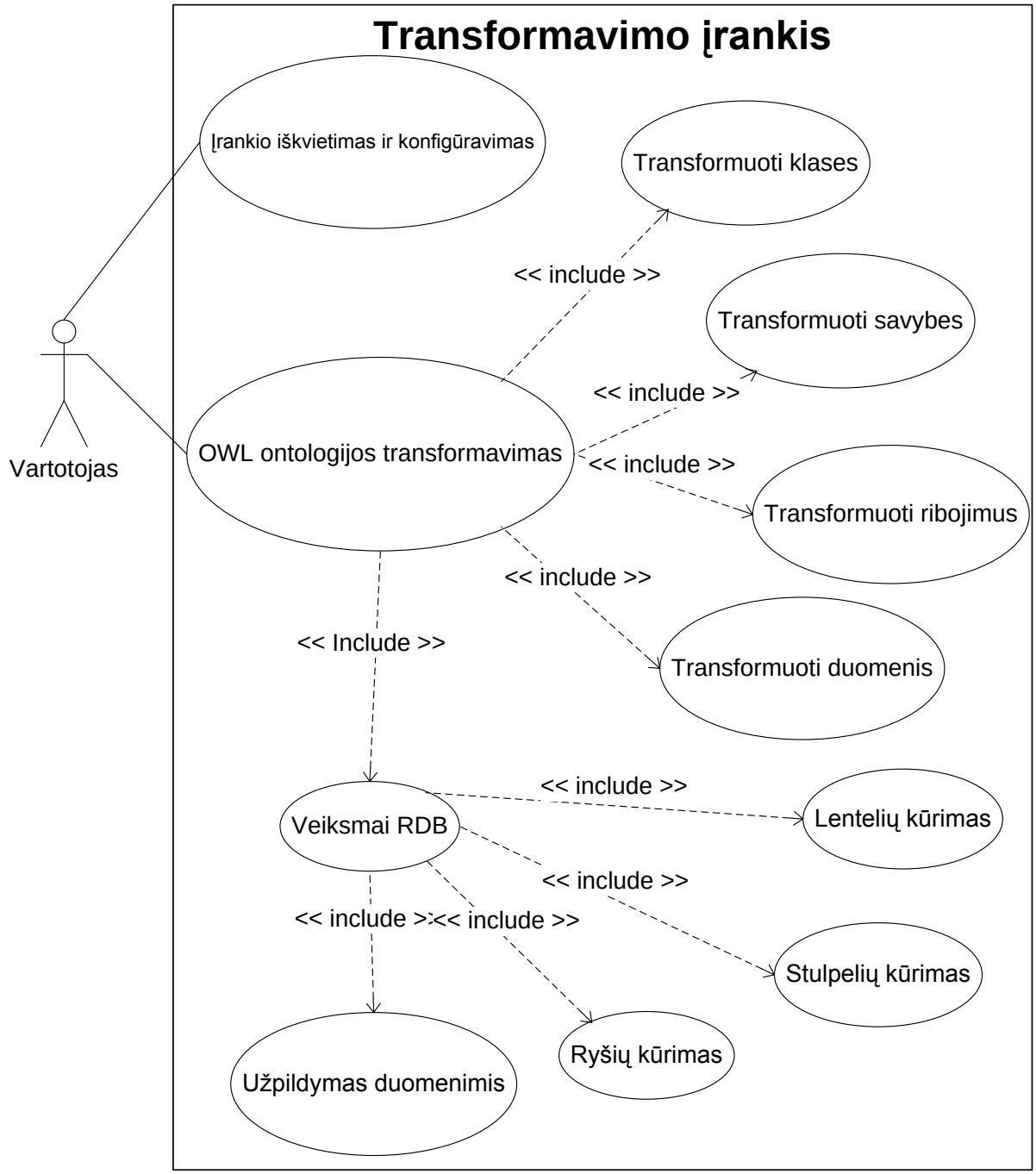
Paveikslėlis Nr. 18. Konteksto diagrama

Veiklos įvykių sąrašas bei juose naudojami informacijos srautai pateikiami lentelėje (Lentelė Nr. 2).

Lentelė Nr. 2. Veiklos įvykių sąrašas

Eil.nr.	**Įvykio pavadinimas**	**Įeinantys/išeinantys informacijos srautai**
1	**OWL2RDB įrankis skaito ontologiją iš Protege**	**OWL ontologija (in)**
2	**OWL2RDB įrankis sukuria RDB schemą**	**Esybės (out)** **Ryšiai (out)** **Apribojimai (out)** **Duomenys (out)**
3	**OWL2RDB įrankis gauna RDB kūrimo ataskaitą**	**RDB sukūrimo ataskaita (in)**
4	**OWL2RDB įrankis gauna RDB schemos vaizdą**	**RDB schemos vaizdas (in)**

Ribas tarp vartotojo ir transformavimo įrankio bei algoritmo nusako panaudojimo atvejų diagrama (Paveikslėlis Nr. 19).



Paveikslėlis Nr. 19. Panaudojimo atvejų diagrama

4.1.1.1. Panaudojimo atvejų sąrašas

Pateikiami visų panaudojimo atvejų aprašymai.

1. PANAUDOJIMO ATVEJIS: Įrankio iškvietimas ir konfigūravimas	
Vartotojas/Aktorius:	Vartotojas
Aprašas:	Apima procesą, kurio metu įrankis yra iškviečiamas projektuotojo arba Protege sistemos ir yra paruošiamas darbui nustatant konfigūraciją Šis panaudojimo atvejis yra išplėstas RDB prisijungimo duomenų saugojimu
Prieš sąlyga:	Transformavimo įrankis nėra paleistas. Sukurta OWL ontologija.
Sužadinimo sąlyga:	OWL ontologija yra sukurta ir norima ją transformuoti į reliacinės duomenų bazės schemą.
Po-sąlyga:	Įrankis paruoštas darbui.

2. PANAUDOJIMO ATVEJIS: OWL ontologijos transformavimas	
Vartotojas/Aktorius:	Vartotojas
Aprašas:	Apima procesą, kurio metu sukurta OWL ontologiją yra transformuojama į RDB schemą Šis panaudojimo atvejis įtraukia klasių, savybių, ribojimų ir duomenų transformavimą
Prieš sąlyga:	Sukurta OWL ontologija. Įrankis paruoštas darbui.
Sužadinimo sąlyga:	Norima transformuoti sukurtą ontologiją.
Po-sąlyga:	Sukurti reliacinės duomenų bazės kūrimo duomenys.

3. PANAUDOJIMO ATVEJIS: Reliacinės duomenų bazės kūrimas/atnaujinimas	
Vartotojas/Aktorius:	Vartotojas
Aprašas:	Apima procesą, kurio metu įrankis sukuria arba atnaujina reliacinės duomenų bazės schemą
Prieš sąlyga:	Paruošti reliacinės duomenų bazės kūrimo duomenys Nurodyti reliacinės duomenų bazės prisijungimo duomenys
Sužadinimo sąlyga:	Norima sukurti reliacinės duomenų bazės schemą.

Po-sąlyga:	Sukurta reliacinės duomenų bazės schema pagal sukurtą OWL ontologiją.

4.1.1.2. Funkciniai reikalavimai ir reikalavimai duomenims

4.1.1.2.1. Funkciniai reikalavimai

Pateikiami visi užregistruoti funkciniai reikalavimai įrankiui bei algoritmui.

Reikalavimas #:	*1*	Reikalavimo tipas:	*1*	Įvykis/panaudojimo atvejis #:	*1/1*
Aprašymas:	*Įrankis integruotas į Protege sistemą*				
Pagrindimas:	*Vartotojas sukūręs ontologiją nori, kad transformavimo įrankį būtų galima patogiai iškviesti naudojantis Protege sistemos meniu*				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	*Įrankis iškviečiamas naudojantis Protege sistemos meniu*				
Užsakovo tenkinimas:	*1*	Užsakovo netenkinimas:	*5*		
Priklausomybės:	Nėra		Konfliktai:	Nėra	
Papildoma medžiaga:					
Istorija:	*Užregistruotas 2009 balandžio 21 d.*				

Reikalavimas #:	*2*	Reikalavimo tipas:	*1*	Įvykis/panaudojimo atvejis #:	*1/1*
Aprašymas:	*Įrankiui nustatoma ir išsaugoma konfigūracija*				
Pagrindimas:	*Vartotojas įveda transformavimo ir RDB prisijungimo nustatymus. RDB prisijungimo duomenys apsaugomi. Sistema išsaugo nustatymus.*				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	*Įrankis išsaugo transformavimo ir RDB prisijungimo nustatymus. RDB prisijungimo duomenys apsaugomi*				
Užsakovo tenkinimas:	*1*	Užsakovo netenkinimas:	*2*		
Priklausomybės:	Nėra		Konfliktai:	Nėra	
Papildoma medžiaga:					
Istorija:	*Užregistruotas 2009 birželio 22 d.*				

Reikalavimas #:	*3*	Reikalavimo tipas:	*2*	Įvykis/panaudojimo atvejis #:	*1/2*
Aprašymas:	*Įrankis turi gebėti nustatyti sukurtos ontologijos tinkamumą transformacijai*				
Pagrindimas:	*Vartotojo sukurta ontologija Protégé sistemoje gali panaudoti OWL kalbos darinius, kurie nėra palaikomi transformuojant ontologiją į reliacinės duomenų bazės schemą*				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	*Įrankis informuoja vartotoją, jei jo sukurta ontologija negali būti transformuojama į reliacinės duomenų bazės schemą*				
Užsakovo tenkinimas:	*2*	Užsakovo netenkinimas:	*4*		
Priklausomybės:	Nėra	Konfliktai:	Nėra		
Papildoma medžiaga:	Transformavimo algoritmo specifikacija				
Istorija:	*Užregistruotas 2009 balandžio 21 d.* *Koreguotas 2009 birželio 22 d.*				

Reikalavimas #:	*4*	Reikalavimo tipas:	*3*	Įvykis/panaudojimo atvejis #:	*1/2*
Aprašymas:	*Įrankis pagal algoritmą turi transformuoti Protégé sistema sukurtą OWL ontologiją į reliacinės duomenų bazės schemos kūrimo SQL skriptą*				
Pagrindimas:	*Reliacinei duomenų bazei aprašyti naudojamas SQL skriptas – dėl to tai yra transformavimo rezultatas*				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	*Atlikus transformaciją, galima išsaugoti/peržiūrėti rezultatus – SQL kalba sukurtą skriptą, kurio pagalba būtų sukurta reliacinės duomenų bazės schema*				
Užsakovo tenkinimas:	*3-4*	Užsakovo netenkinimas:	*5*		
Priklausomybės:	Nėra	Konfliktai:	Nėra		
Papildoma medžiaga:					
Istorija:	*Užregistruotas 2009 balandžio 21 d.* *Koreguotas 2009 birželio 22 d.*				

Reikalavimas #:	*5*	Reikalavimo tipas:	*3*	Įvykis/panaudojimo atvejis #:	*1/2*
Aprašymas:	*Reliacinės duomenų bazės schemos kūrimo SQL skriptas pritaikomas vartotojo pasirinktai reliacinei duomenų bazei*				
Pagrindimas:	*Sistema turi palaikyti Microsoft SQL Server bei Oracle reliacines duomenų bazes*				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	*Atlikus transformaciją, SQL kalba sukurtas skriptas atitinka konfigūracijoje nustatytos reliacinės duomenų bazės sintaksę*				
Užsakovo tenkinimas:	*4*	Užsakovo netenkinimas:	*4*		
Priklausomybės:	4	Konfliktai:	Nėra		
Papildoma medžiaga:					
Istorija:	*Užregistruotas 2009 balandžio 21 d.*				

Reikalavimas #:	*6*	Reikalavimo tipas:	*4*	Įvykis/panaudojimo atvejis #:	*1/2*
Aprašymas:	*Sukurta reliacinės duomenų bazės schema sąlygoja minimalius ontologijos informacijos nuostolius*				
Pagrindimas:	*Algoritmo tikslas yra OWL ontologiją transformuoti į reliacinės duomenų bazės schemą, todėl šio proceso metu atsirandantys informacijos nuostoliai sąlygos sukurtos schemos neatitikimą pradinei ontologijai*				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	*Sulyginama pradinė ontologija ir gautas reliacinė duomenų bazės schema ir nustatomas jų atitikimas*				
Užsakovo tenkinimas:	*5*	Užsakovo netenkinimas:	*4*		
Priklausomybės:	4	Konfliktai:	Nėra		
Papildoma medžiaga:					
Istorija:	*Užregistruotas 2009 balandžio 21 d.*				

Reikalavimas #:	*7*	Reikalavimo tipas:	*5*	Įvykis/panaudojimo atvejis #:	*2/3*
Aprašymas:	*Įrankis prisijungia ir sukuria schemą veikiančioje reliacinėje duomenų bazėje*				

Pagrindimas:	*Projektuotojas nebūtinai moka/turi leidimą naudotis reliacinės duomenų bazės administravimo įrankiais, todėl, gavęs prisijungimo prie duomenų bazės duomenis, įrankis automatizuotai sukuria schemą*		
Šaltinis:	Užsakovas		
Tikimo kriterijus:	*Įrankiui nurodžius prisijungimo duomenis, sukuriama reliacinės duomenų bazės schema*		
Užsakovo tenkinimas:	*3*	Užsakovo netenkinimas:	*3-4*
Priklausomybės:	Nėra	Konfliktai:	Nėra
Papildoma medžiaga:			
Istorija:	*Užregistruotas 2009 balandžio 21 d.*		

Reikalavimas #:	*8*	Reikalavimo tipas:	*5*	Įvykis/panaudojimo atvejis #:	*2/3*
Aprašymas:	*Įrankis prisijungia ir atnaujina schemą veikiančioje reliacinėje duomenų bazėje*				
Pagrindimas:	*Projektuojant ontologija gali būti tobulinama todėl gali būti reikalingos pakartotinės transformacijos, o tuo pačiu ir anksčiau sukurtos reliacinės duomenų bazės atnaujinimas*				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	*Įrankiui nurodžius prisijungimo duomenis, atnaujina reliacinės duomenų bazės schemą*				
Užsakovo tenkinimas:	*3*	Užsakovo netenkinimas:		*2-3*	
Priklausomybės:	Nėra	Konfliktai:		Nėra	
Papildoma medžiaga:					
Istorija:	*Užregistruotas 2009 balandžio 21 d.*				

Reikalavimas #:	*9*	Reikalavimo tipas:	*3*	Įvykis/panaudojimo atvejis #:	*2/2*
Aprašymas:	*Įrankis pagal algoritmą turi sukurti visoms klasėms po lentelę*				
Pagrindimas:	*Vykdant transformaciją, įrankis kiekvienai klasei sukuria po lentelę ir 1:1 ryšius su aukštesnėmis klasėmis*				

Šaltinis:	Užsakovas		
Tikimo kriterijus:	*Kiekviena klasė turi atitinkamą lentelę RDB*		
Užsakovo tenkinimas:	*1*	Užsakovo netenkinimas:	*5*
Priklausomybės:	Nėra	Konfliktai:	Nėra
Papildoma medžiaga:			
Istorija:	*Užregistruotas 2009 birželio 22 d.*		

Reikalavimas #:	*10*	Reikalavimo tipas:	*3*	Įvykis/panaudojimo atvejis #:	*2/2*
Aprašymas:	*Įrankis pagal algoritmą turi sukurti visoms objektų savybėms ryšius*				
Pagrindimas:	*Vykdant transformaciją, įrankis kiekvienai objekto savybei sukuria ryšį su klase, kuriai ji taikoma*				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	*Kiekviena objekto savybė turi atitinkamą ryšį RDB*				
Užsakovo tenkinimas:	*1*	Užsakovo netenkinimas:		*5*	
Priklausomybės:	Nėra	Konfliktai:		Nėra	
Papildoma medžiaga:					
Istorija:	*Užregistruotas 2009 birželio 22 d.*				

Reikalavimas #:	*11*	Reikalavimo tipas:	*3*	Įvykis/panaudojimo atvejis #:	*2/2*
Aprašymas:	*Įrankis pagal algoritmą turi sukurti visoms duomenų tipų savybėms po stulpelį*				
Pagrindimas:	*Vykdant transformaciją, įrankis kiekvienam duomenų tipui sukuria po stulpelį atitinkamoje lentelėje*				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	*Visos duomenų tipų savybės turi po stulpelį*				
Užsakovo tenkinimas:	*1*	Užsakovo netenkinimas:		*5*	
Priklausomybės:	Nėra	Konfliktai:		Nėra	
Papildoma medžiaga:					
Istorija:	*Užregistruotas 2009 birželio 22 d.*				

Reikalavimas #:	*12*	Reikalavimo tipas:	*3*	Įvykis/panaudojimo atvejis #:	*2/2*
Aprašymas:	*Įrankis pagal algoritmą turi sukurti visiems apribojimams metaduomenis tam skirtose lentelėse*				
Pagrindimas:	*Vykdant transformaciją, įrankis visiems apribojimams turi kurti metaduomenis tam skirtose lentelėse – kiekvienam apribojimo tipui yra skirtos atskiros lentelės*				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	*Kiekvienas apribojimas turi metaduomenų eilutę atitinkamoje lentelėje pagal jo tipą*				
Užsakovo tenkinimas:	*1*	Užsakovo netenkinimas:	*5*		
Priklausomybės:	Nėra	Konfliktai:	Nėra		
Papildoma medžiaga:					
Istorija:	*Užregistruotas 2009 birželio 22 d.*				

Reikalavimas #:	*13*	Reikalavimo tipas:	*3*	Įvykis/panaudojimo atvejis #:	*/2*
Aprašymas:	*Transformavimo algoritmas turi būti išplėstas*				
Pagrindimas:	*Projekto metu transformavimo algoritmas turi būti išplėstas naujomis savybėmis, kurios leistų panaudoti daugiau OWL DL kalbos konstruktų*				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	*Transformavimo algoritmas geba transformuoti didesnį OWL DL kalbos konstruktų kiekį nei užsakovo pateiktas*				
Užsakovo tenkinimas:	*5*	Užsakovo netenkinimas:	*3*		
Priklausomybės:	Nėra	Konfliktai:	Nėra		
Papildoma medžiaga:					
Istorija:	*Užregistruotas 2009 birželio 22 d.*				

4.1.1.3. Nefunkciniai reikalavimai

Įrankio sąsaja turėtų būti paprastai naudojama ir nereikalaujanti specialių apmokymų – turėtų pakakti turimo projektuotojo išsilavinimo ir patirties naudojant pačią Protege sistemą.

Transformavimo algoritmo sukurta reliacinės duomenų bazės schema turi užtikrinti geresnes užklausų vykdymo charakteristikas nei naudojant esamas priemones.

Užduočių vykdymo greitis tiesiogiai priklauso nuo sukurtos ontologijos apimties OWL sistemoje. Kadangi jos gali būti labai didelės ir reikalaujančios laiko, įrankis privalo išlikti bendraujantis su vartotoju, suteikti galimybę atšaukti transformavimą, turėtų gebėti bent apytiksliai nurodyti kokia dalis darbų jau atlikta.

Protege yra nemokama atviro kodo sistema. Sukurtas produktas turi būti atviro kodo, nemokamas, prieinamas Protege portale.

Kuriamas atviro kodo produktas, dėl to jį pateikus į Protege portalą visi jo vartotojai galės pataisyti ar patobulinti šį įrankį. Dažniausiai pasitaikanti situacija – kai pradinis plėtinio autorius yra vienas asmuo, o pasiteisinus jo teikiamai naudai – priežiūra ir palaikymu rūpinasi portalo savininkai (save įvardindami „Protege team“).

Įrankio kalba turi būti anglų kalba. Kadangi tai yra atviro kodo įrankis, anglų kalba turi būti taikoma taip pat ir vartotojui nematomose srityse – programiniame kode, jo komentaruose, failų pavadinimuose ir pan.

Įrankis dirbs su duomenų bazės prisijungimo vardu ir slaptažodžiu – svarbu užtikrinti saugų šių duomenų įvedimą kaip įprasta naudojamoje operacinėje sistemoje, t.y. slaptažodį įvedant ar įvedus jis negali būti išvedamas perskaitymui (pvz. pateikiamas ekrane, saugomas failuose ir pan.), o jeigu yra būtinybė išsaugoti – užkoduojamas.

Užsakovo pateiktas algoritmas gali būti tobulinamas, bet to būtinybė paaiškės po detalios jo analizės arba eksperimentų rezultatų.

Galimos naujos sistemos sukeltos problemos:

- Ar įrankis optimaliai išnaudos kompiuterio resursus?
- Ar įrankis pakankamai greitai transformuos ontologijas?
- Ar darbo laiko didėjimas dėl didėjančios ontologijos bus priimtinas?

Perspektyviniai reikalavimai:

- Transformacijos algoritmo veikimas be jokių informacijos nuostolių – sukuriama reliacinės duomenų bazės schema, kuri išlaiko visą ontologijoje esančią informaciją.
- Palaikoma visa OWL kalba.
- Palaikymas vėlesnės OWL kalbos versijoms.

4.2. Projekto modelis

4.2.1. Architektūros pateikimas

Esminės funkcijos, kurias privalo atlikti transformavimo algoritmas ir įrankis, vaizduojamos panaudojimų atvejų diagrama (Paveikslėlis Nr. 19).

Statinė sistemos architektūra pateikiama šiais vaizdais:

- Bendra paketų diagrama, kuri vaizduoja, kokie esminiai programinio kodo paketai sudaro įrankį;
- Detalizuota paketų diagrama, kuri vaizduoja kiekvieno iš anksčiau vaizduotųjų paketų sudėtį, pateikia svarbiausius detalesnius paketus bei klases.

Dinaminė sistemos architektūra pateikiama šiais vaizdais:

- Sekų diagramomis, kurios vaizduoja, kokiomis veiksmų sekomis realizuojamas kiekvienas panaudojimo atvejis;
- Būsenų diagramomis, kurios kiekvienam panaudojimo atvejui vaizduoja sistemos būseną įvairiais laiko momentais bei parodo, kokiomis sąlygomis ji kinta;
- Veiklos diagramomis, kurios vaizduoja kiekvieno panaudojimo atvejo dalyvius bei jų veiklos pasiskirstymą įvairiais laiko momentais;
- Bendradarbiavimo diagrama, kuri vaizduoja antrojo panaudos atvejo esminius elementus bei jų tarpusavio sąveiką.

4.2.2. Architektūros tikslai ir apribojimai

Projektavimo ir įgyvendinimo strategija: sistema projektuojama pagal RUP metodiką. Įrankio kūrimo pagrindas yra transformavimo algoritmas – jį pateikė užsakovas, atliekant analizę ir eksperimentus jis gali būti patobulintas išlaikant pradinę ideologiją.

Įrankis veiks kaip Protégé sistemos plėtinys. Šis plėtinys bus pasiekiamas per Protégé sistemos meniu, jį naudos projektuotojas po to, kai sukurs ontologiją. Ontologijos kūrimas, redagavimas, peržiūrėjimas, tikrinimas ir kt. bus atliekama Protégé sistemos.

Įrankio naudojimas neturėtų reikalauti papildomos techninės įrangos iš jos vartotojo, nei pati Protégé sistema. Tačiau jeigu vartotojas nori atlikti veiksmus su duomenų baze, jam reikalingi atitinkami sistemos komponentai, kurie leistų ją pasiekti (pavyzdžiui, atitinkamos versijos Oracle klientas bei JDBC tvarkyklė).

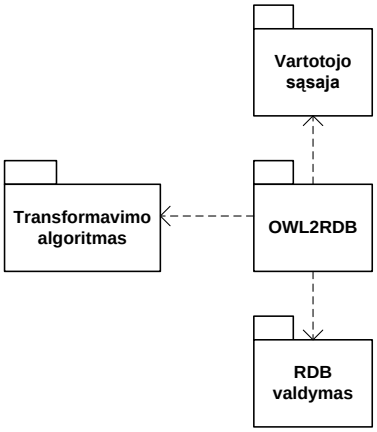
Įrankio architektūra turi būti pritaikyta išplėtimui ir pritaikymui kitose sistemose. Kadangi programavimo kalba yra Java, įrankis neturi operacinės sistemos apribojimų, nes Java kompiliatoriai sukurti visoms šiuo metu populiariausioms operacinėms sistemoms.

Šio darbu metu kuriamas įrankis nėra finansiškai remiamas ir finansuojamas, todėl norint minimizuoti kūrimo išlaidas, bus naudojama nemokama arba tiesiogiai projekto reikmėms neįgyjama programinė įranga (Protégé, Java SE, Eclipse, Microsoft SQL 2005 Express Edition, Oracle Database 10g Express Edition).

4.2.3. Sistemos statinis vaizdas

4.2.3.1. Apžvalga

Pateikiamas įrankio išskaidymas į paketus (Paveikslėlis Nr. 20). Kiekvienas paketas atlieka skirtingas funkcijas. Taip pat tokiu būdu išskaidžius įrankį į paketus bus izoliuotas transformavimo algoritmo funkcionalumas.

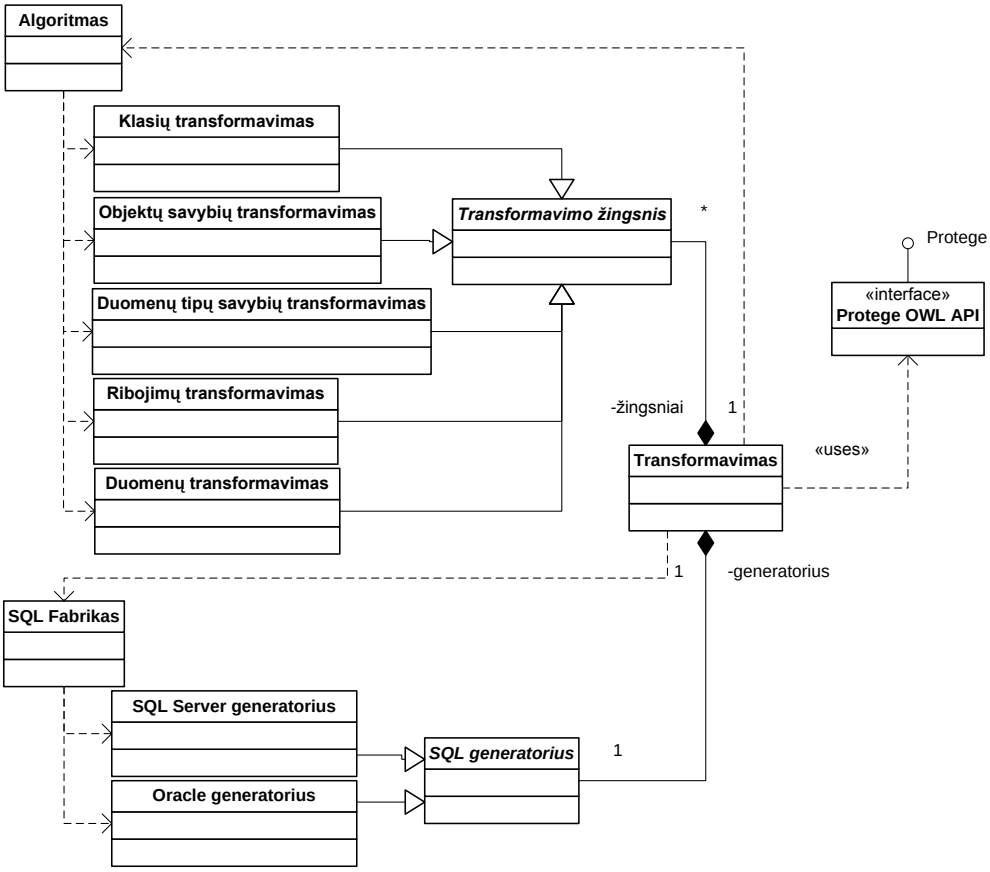


Paveikslėlis Nr. 20. Įrankio išskaidymas į paketus

4.2.3.2. Paketų detalizavimas

4.2.3.2.1. Transformavimo algoritmas

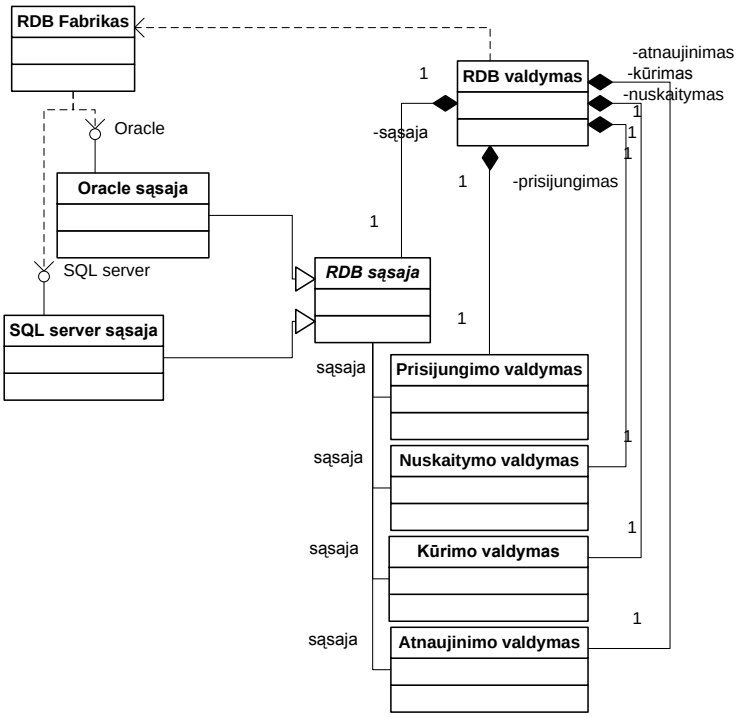
Šis paketas realizuoja transformavimo algoritmą. Jis gauna OWL ontologijos duomenis iš Protégé sistemos ir sukuria RDB kūrimo skriptą.



Paveikslėlis Nr. 21. Transformavimo algoritmo paketo klasių diagrama

4.2.3.2.2. RDB valdymas

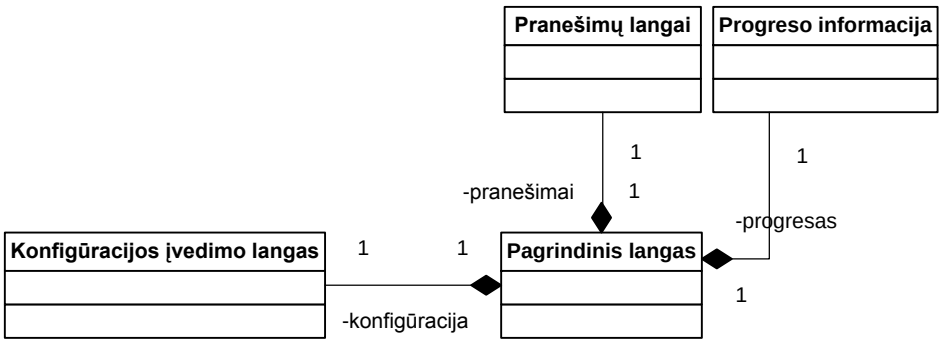
Šis paketas realizuoja sukurto RDB kūrimo skripto įvykdymą. Jis gauna RDB kūrimo skriptą iš transformavimo algoritmo ir sukuria/atnaujina schemą prisijungęs prie vartotojo nurodytos reliacinės duomenų bazės.



Paveikslėlis Nr. 22. RDB valdymo paketo klasių diagrama

4.2.3.2.3. Vartotojo sąsaja

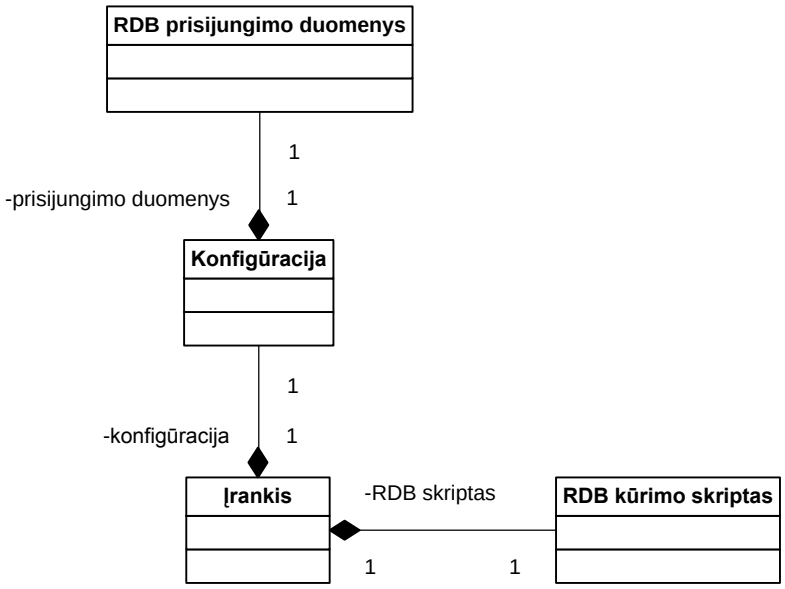
Šis paketas realizuoja grafinę vartotojo sąsają – darbo parametrų įvedimą, informavimą apie progresą, pranešimus apie klaidas arba sėkmingą darbo pabaigą.



Paveikslėlis Nr. 23. Vartotojo sąsajos paketo klasių diagrama

4.2.3.2.4. OWL2RDB

Šis paketas realizuoja centrinį įrankio valdymą.

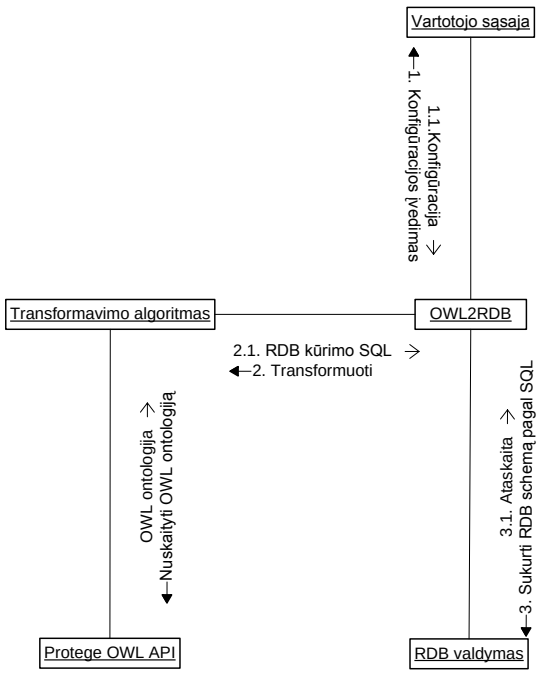


Paveikslėlis Nr. 24. OWL2RDB paketo klasių diagrama

4.2.4. Sistemos dinaminis vaizdas

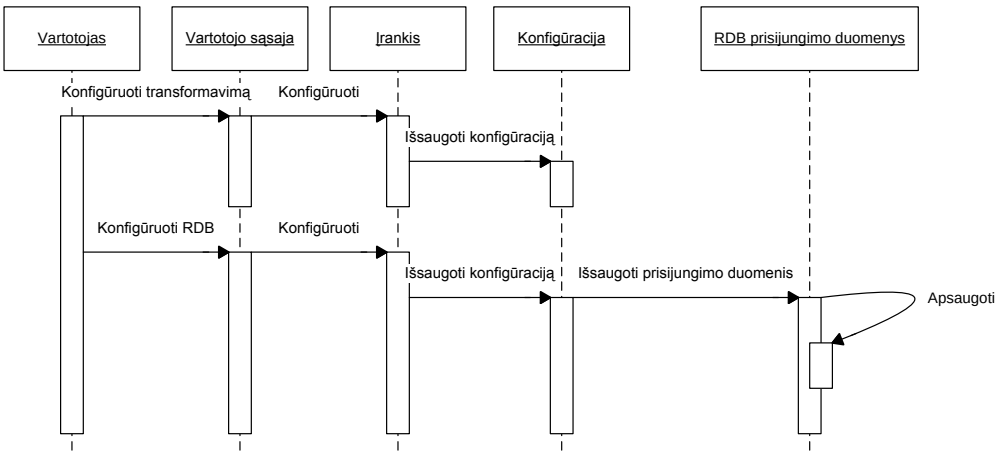
4.2.4.1. Apibendrintas dinaminis vaizdas

Pateikiamas apibendrintas dinaminis vaizdas panaudojant bendradarbiavimo diagramą (Paveikslėlis Nr. 25). Juo apibendrinami informacijos srautai tarp paketų bei jų nuoseklumas.



Paveikslėlis Nr. 25. Apibendrinta įrankio bendradarbiavimo diagrama

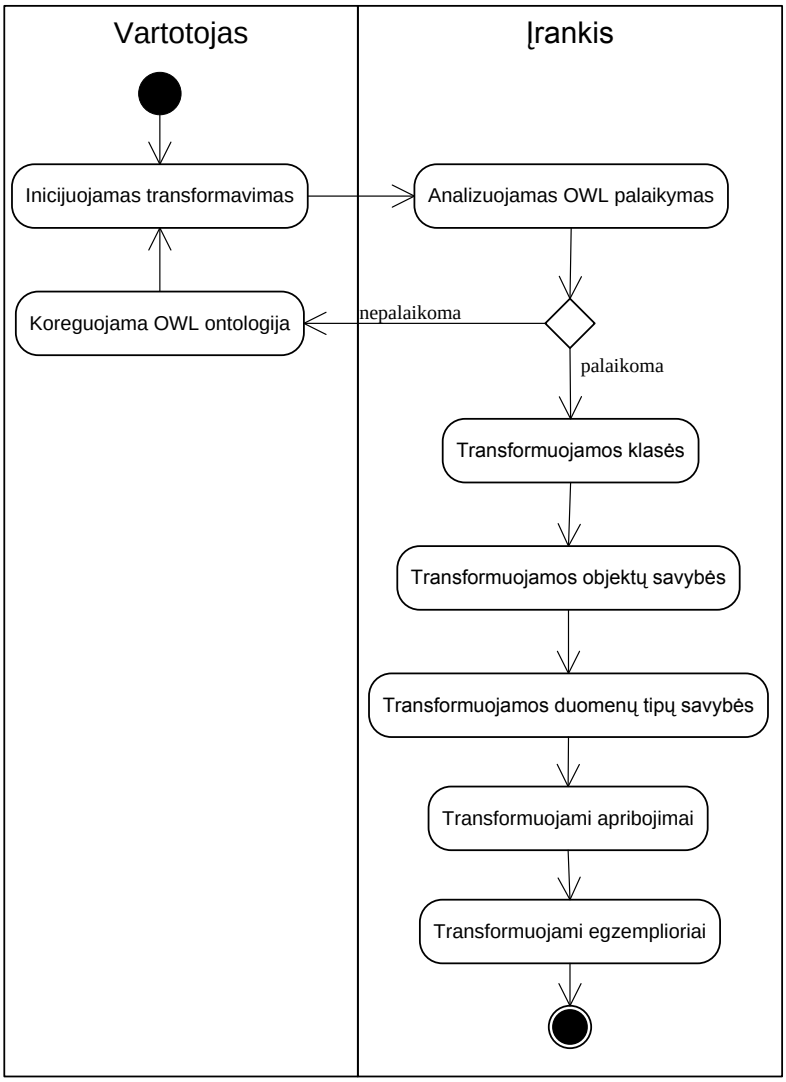
4.2.4.2. Konfigūravimo dinaminis vaizdas



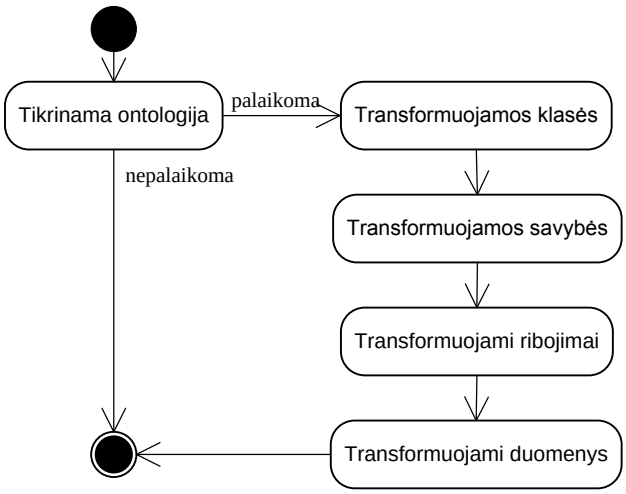
Paveikslėlis Nr. 26. Konfigūravimo sekos diagrama

4.2.4.3. Transformavimo dinaminis vaizdas

OWL ontologijos transformavimas atliekamas nuosekliai vykdant algoritmo veiksmus, skirtus skirtingiems elementams (Paveikslėlis Nr. 29).

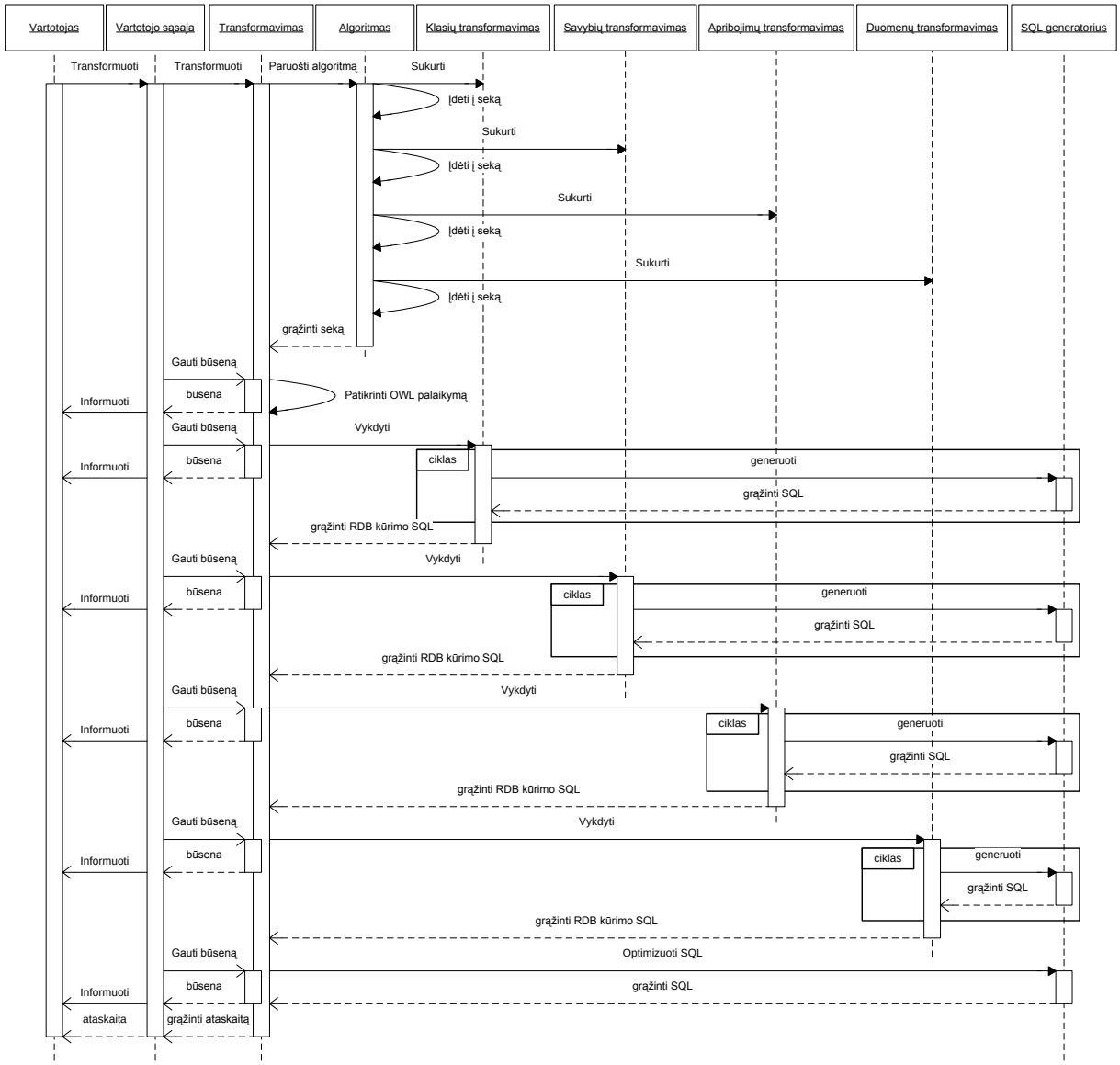


Paveikslėlis Nr. 27. Įrankio transformavimo veiklos schema



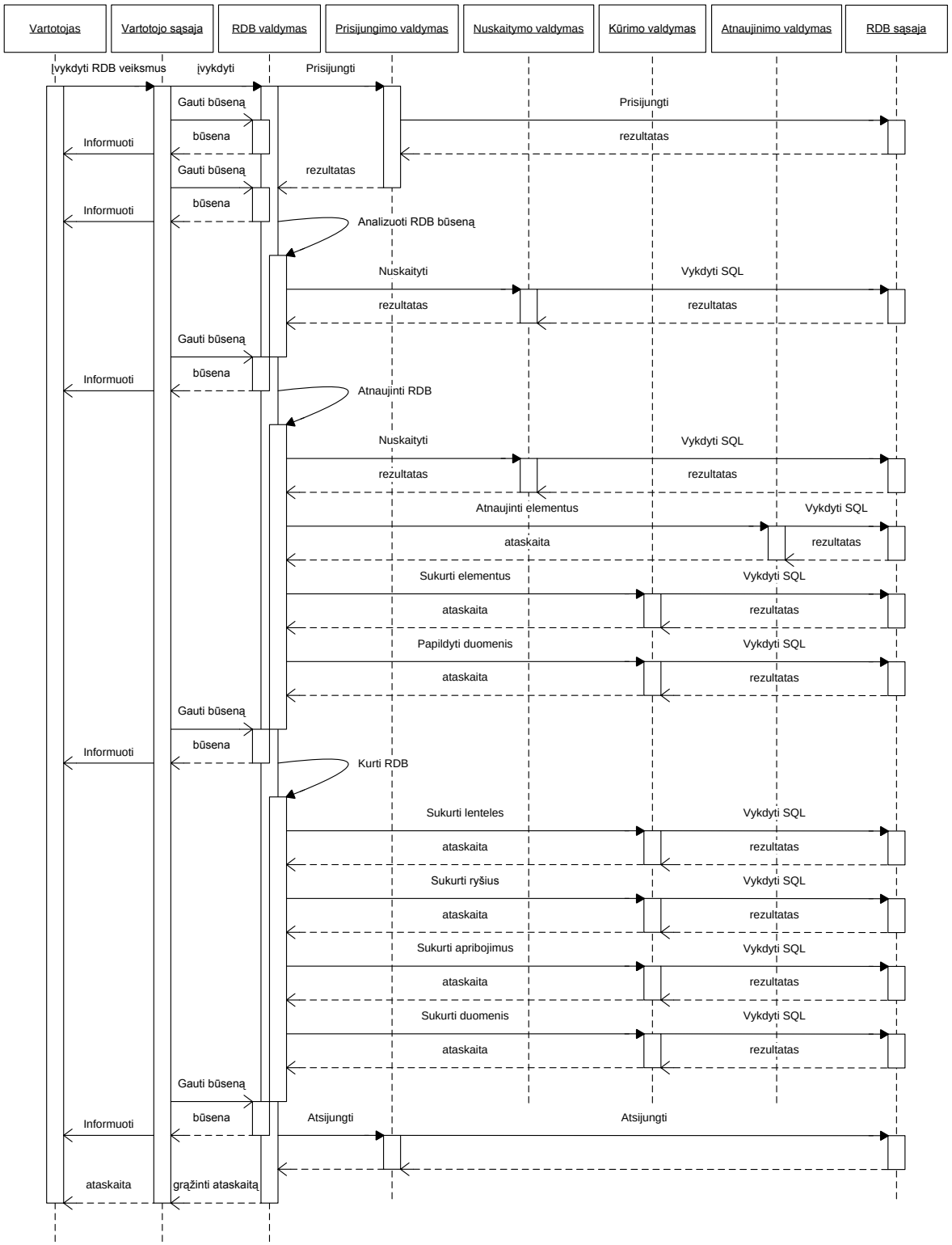
Paveikslėlis Nr. 28. Įrankio transformavimo proceso būsenų diagrama

Viso transformavimo paketo veikimas pateikiamas sekos schema (Paveikslėlis Nr. 31).

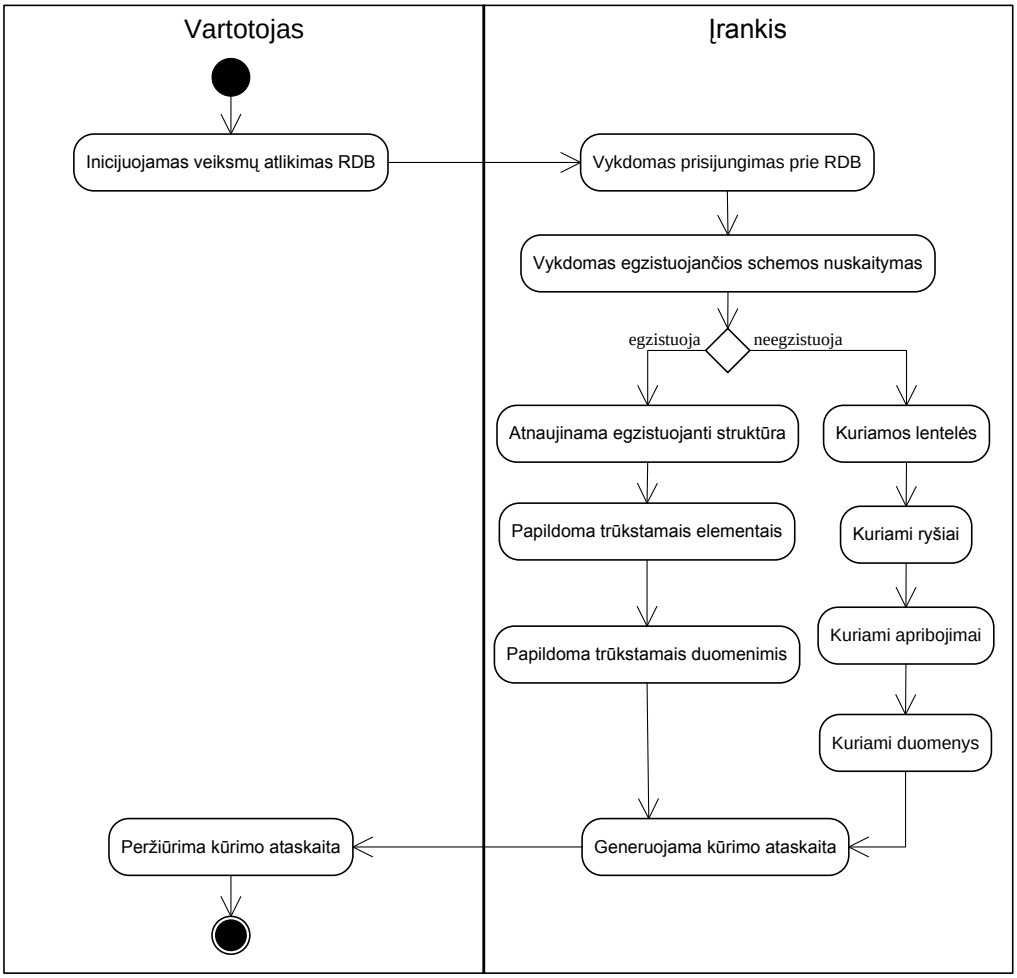


Paveikslėlis Nr. 29. Transformavimo sekos schema

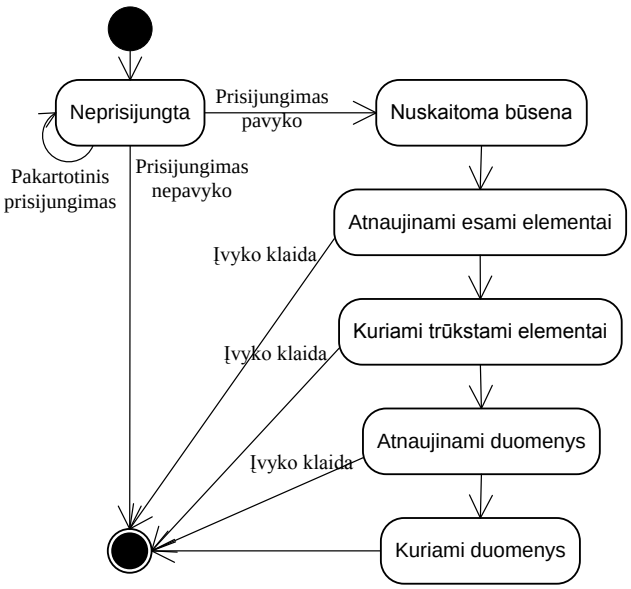
4.2.4.4. Veiksmų RDB dinaminis vaizdas



Paveikslėlis Nr. 30. Veiksmų RDB sekos diagrama



Paveikslėlis Nr. 31. Veiksmų RDB veiklos diagrama



Paveikslėlis Nr. 32. Veiksmų RDB būsenų diagrama

4.2.5. Įrankio kokybė

Saugumas. Dirbant su įrankiu, vartotojas įveda prisijungimo prie reliacinės duomenų bazės duomenis. Šių duomenų apsauga yra labai svarbi, dėl to bus imtasi papildomų priemonių, kad ši informacija būtų koduojama ją saugant.

Pasiekiamumas. Įrankis yra pasiekiamas tiesiogiai iš vartotojo naudojamos Protégé sistemos, todėl projektuotojai galės be vargo bet kuriuo metu savo sukurtą ontologiją transformuoti į reliacinės duomenų bazės schemą.

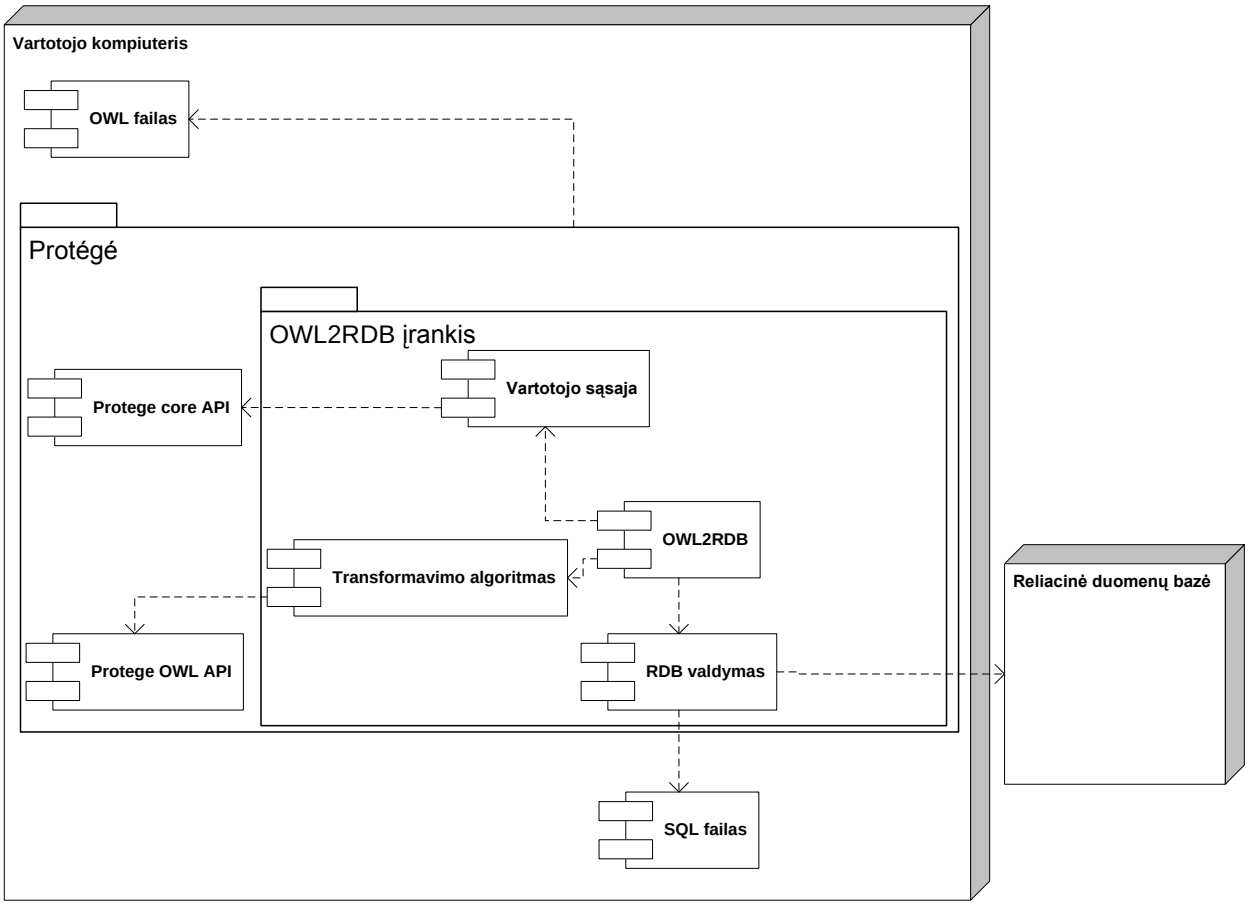
Išplečiamumas. Įrankio architektūra sukurta taip, kad būtų kuo lengviau atlikti naujų sistemos funkcijų integravimą į jau sukurtą programinė įrangą. Specialių sąsajos paketų su Protégé naudojimas leidžia juos pakeisti kitais ir pritaikyti įrankį kitai sistemai. Transformavimo algoritmo paketo struktūra leidžia jį papildyti naujais žingsniai bei nesudėtingai keisti jų seką. Taip pat SQL kodo generavimo bei RDB naudojimo principai leidžia įrankį išplėsti taip, kad palaikytų daugiau reliacinių duomenų bazių arba išnaudotų naujose jų versijose realizuojamą funkcionalumą.

Panaudojamumas. Vartotojai naudojantys sukurtą įrankį neturi būti kažkaip specialiai apmokyti ar būti susipažinę su specialia programine įranga. Po to, kai vartotojas įdiegs įrankį savo Protégé sistemoje, užteks poros trivialių žingsnių, po kurių būtų atliekamas sudėtingas OWL ontologijos į RDB schemą transformacijos procesas.

5. REALIZACIJA IR EKSPERIMENTAS

Šiame skyriuje aprašytas transformavimo įrankio išdėstymas Protege sistemoje, demonstruojama jo realizacija. Taip pat aprašytas atliktas algoritmo rezultatus tiriantis eksperimentas.

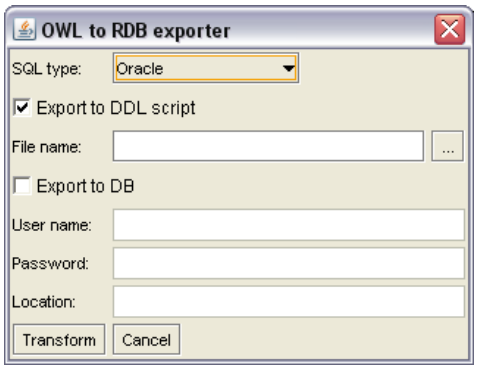
5.1. Įrankio išdėstymas Protege sistemoje



Paveikslėlis Nr. 33. Įrankio išdėstymo diagrama

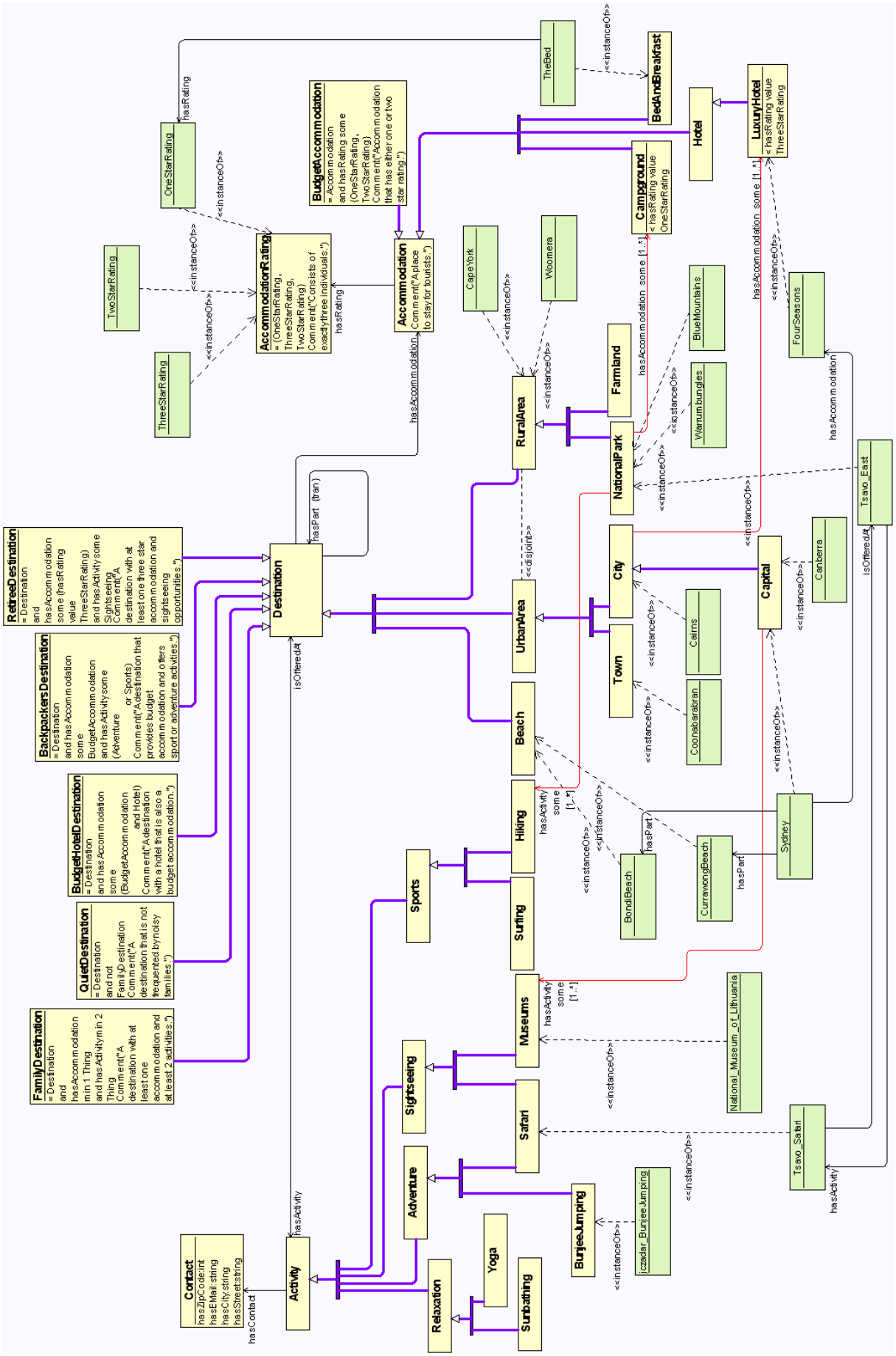
5.2. Realizacija

Sukurtas įrankis realizuoja tiriamą algoritmą ir tokiu būdu geba transformuoti OWL ontologijas automatiškai į RDB schemas. Įrankio grafinė vartotojo sąsaja yra minimali ir neapkrauna Protege sistemos vartotojo (Paveikslėlis Nr. 34).



Paveikslėlis Nr. 34. Įrankio pagrindinis langas

Pradinė ontologija, kuria bus demonstruojama algoritmo realizacija, pateikiama dviem OWL schema (Paveikslėlis Nr. 35). Taip pat papildomai pateikiami visi apribojimai (Lentelė Nr. 3). Tai yra keliones aprašanti ontologija, kuri yra prieinama viešai Protege svetainėje, ontologijų bibliotekoje [17] . Keletas egzempliorių buvo pridėta pademonstruoti lentelių pildymą.



Paveikslėlis Nr. 35. OWL schema

Klasės šioje ontologijoje aprašo galimas kelionių vietas, veiklas ir apgyvendinimo vietas. Šios sritys yra detalizuojamos hierarchiškai, apibrėžiami ryšiai tarp sričių, nurodomos taisyklės pagal kurias egzemplioriai gali būti joms priskiriami.

Objektų savybės šioje ontologijoje nurodo visų sričių ryšius: apgyvendinimo vieta (Accomodation) turi įvertinimą (AccomodationRating), kurį įprasta įvertinti žvaigždutėmis, pagal hasRating savybę; kelionės vieta (Destination) turi apgyvendinimo vietas pagal hasAccommodation savybę, gali turėti kitą vietą kaip savo pačios dalį pagal hasPart savybę, taip pat joje galima užsiimti veiklomis (Activity) pagal hasActivity savybę. Veikloms priskirta atvirkštinė savybė hasActivity savybei – isOfferedAt; taip pat veikloms gali būti nurodoma kontaktinė informacija pagal hasContact savybę.

Taip pat pateikiamoje ontologijoje aprašyti AllValuesFrom, SomeValuesFrom, HasValue ir kardinalumo apribojimai (Lentelė Nr. 3).

Lentelė Nr. 3. OWL apribojimai

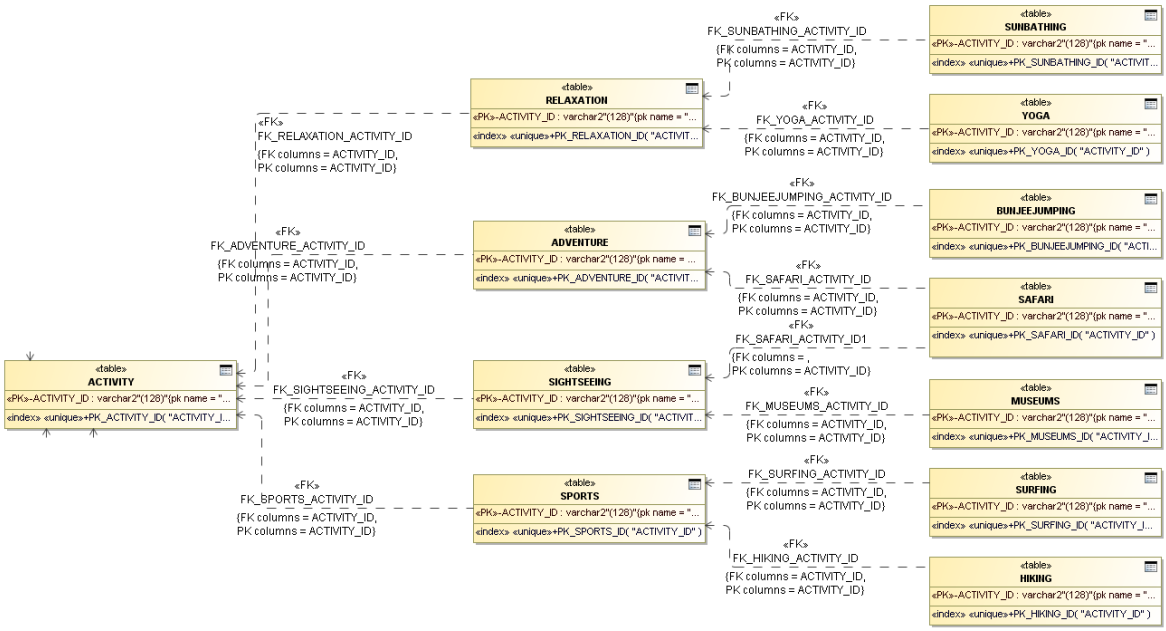
Klasė	Objektų savybė	Apribojimas
BudgetAccommodation	hasRating	someValuesFrom {OneStarRating TwoStarRating}
Campground	hasRating	hasValue OneStarRating
LuxuryHotel	hasRating	hasValue ThreeStarRating
BackPackersDestination	hasAccommodation	someValuesFrom BudgetAccomodation
	hasActivity	someValuesFrom Sports or Adventure
BudgetHotelDestination	hasAccommodation	someValuesFrom BudgetAccomodation and Hotel
FamilyDestination	hasAccommodation	minCardinality 1
	hasActivity	minCardinality 2
RetireeDestination	hasAccommodation	someValuesFrom hasRating has ThreeStarRating
	hasActivity	someValuesFrom SightSeeing
NationalPark	hasAccommodation	someValuesFrom Campground
	hasActivity	someValuesFrom Hiking
City	hasAccommodation	someValuesFrom LuxuryHotel
Capital	hasActivity	someValuesFrom Museums

Apribojimai šioje ontologijoje apibrėžia kokias sąlygas privalo tenkinti egzemplioriai, kad galėtų būti priskirti vienai ar kitai klasei. Pavyzdžiui, norint viešbutį laikyti LuxuryHotel, jis

privalo turėti trijų žvaigždučių reitingą (hasValue ThreeStarRating) – panaudojamas hasValue apribojimas. Lygiai taip pat norint kelionės vietą laikyti FamilyDestination, ji privalo turėti bent vieną apgyvendinimo vietą (minCardinality 1) ir bent dvi veiklas (minCardinality 2) – panaudojamas kardinalumo apribojimas. Apribojimas someValuesFrom apibrėžia, kad pagal objektų savybę priskirtuose egzemplioriuose privalo būti įrašų iš tam tikros klasės – pavyzdžiui, norint kelionės vietos egzempliorių laikyti City, jame privalo egzistuoti viešbutis, kuris yra laikomas LuxuryHotel.

Atlikus pateiktos OWL ontologijos transformaciją, buvo gautos 43 lentelės: 34 lentelės atvaizduoja klases, 5 lentelės išreiškia objektų savybes, 4 lentelėse saugojami metaduomenys apie ontologiją.

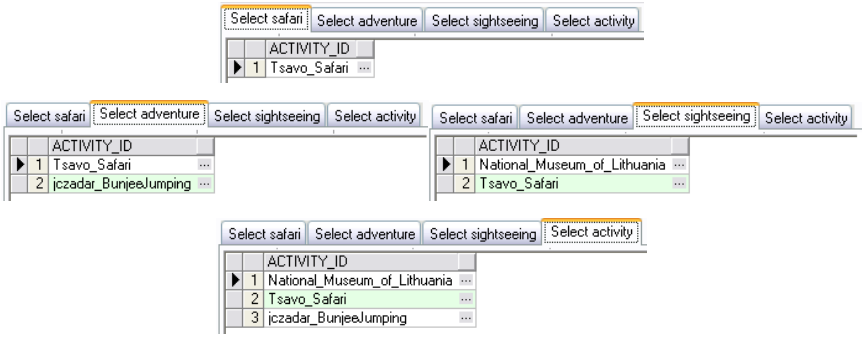
Klasių hierarchinė struktūra išreiškiama žemesnių klasių lentelių pirminiame rakte panaudojant aukštesnės klasės lentelės pirminio rakto pavadinimą iš kurio sukuriamas išorinis raktas į aukštesnę klasę (Paveikslėlis Nr. 36). Gali būti tokių atvejų, kai klasė turi daugiau negu vieną aukštesnę klasę, kurios turi bendrą dar aukštesnę klasė – tokiu atveju iš vieno pirminio rakto lauko žemesnės klasės lentelėje kuriami išoriniai raktai į aukštesnių klasių lenteles (žr. Safari klasės lentelę – iš lauko ACTIVITY_ID sukurti du išoriniai raktai – į ADVENTURE ir SIGHTSEEING lenteles).



Paveikslėlis Nr. 36. RDB lentelių struktūra, kuria išreiškiama hierarchija

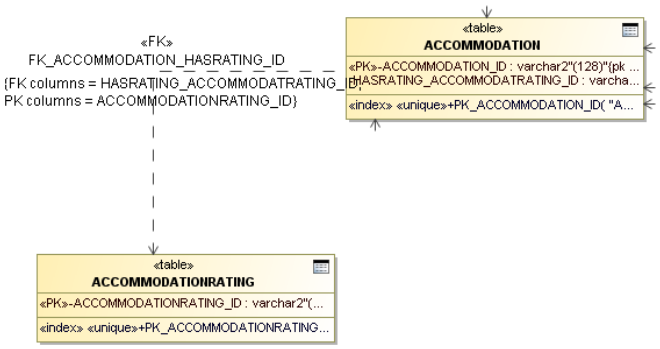
Išsaugojant klasių egzempliorius užpildomos aukštesnių klasių lentelės ir tik tuomet įterpiamas įrašas šios klasės lentelėje – tokiu būdu užtikrinamas duomenų vientisumas, kurio reikalauja visi išoriniai raktai. Taip pat vėliau naudojant įrašus iš aukščiausių klasių lentelių matomi visi žemesnių klasių egzemplioriai. Pavyzdžiui, jeigu klasė SAFARI turi

egzempliorių „Tsavo_Safari“ (Paveikslėlis Nr. 37), tuomet atitinkami įrašai yra sukuriami ACTIVITY, ADVENTURE, SIGHTSEEING lentelėse, ir tik tuomet SAFARI lentelėje. Taip pat tokioje struktūroje norint sužinoti visas veiklas, kurios yra ADVENTURE tipo, užtenka peržiūrėti atitinkamą lentelę, ir nereikia peržiūrinėti visų žemesnių klasių lentelių, kurių struktūra gali tapti sudėtinga augant ontologijos detalizavimo lygiui ir sudėtingumui.

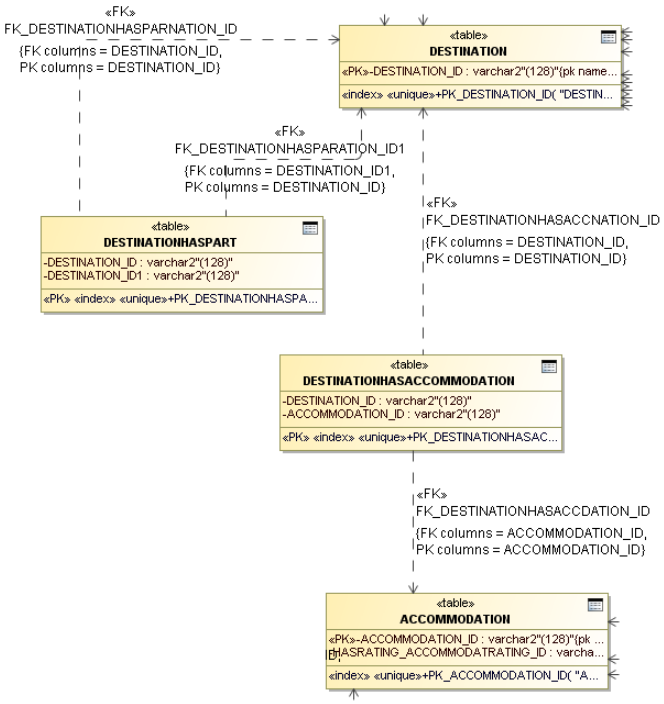


Paveikslėlis Nr. 37. RDB lentelių, kurių klasės susijusios hierarchiškai, turinys

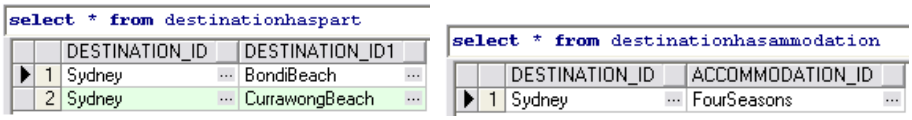
Objektų savybių išsaugojime esminis struktūrinis skirtumas priklauso nuo to ar savybė yra taikoma funkciniu principu. Jeigu savybė yra funkcinė arba jos kardinalumas nurodo tik vieną galima reikšmę – kuriamas 1:N ryšys, kuriam realizuoti užtenka sukurti papildomą lauką(-us), kuriais išreiškiamas išorinis raktas į šios savybės range klasės lentelę (Paveikslėlis Nr. 38). Priešingu atveju realizuojamas M:N ryšys, kuriam reikalinga tarpinė lentelė su pirminiu raktu, kuris sukuria išorinius raktus į domain ir range klasių lenteles (Paveikslėlis Nr. 39). Galima situacija, kai domain ir range klasė sutampa – tuomet laukai pervadinami siekiant išlaikyti jų unikalumą (pavyzdžiui, DestinationHasPart objektų savybės lentelė).



Paveikslėlis Nr. 38. RDB lentelių struktūra, kuria išreiškiamos funkcinės objektų savybės



Paveikslėlis Nr. 39. RDB lentelių struktūra, kuria išreiškiamos nefunkcinės objektų savybės



```
select * from destinationhaspart
```

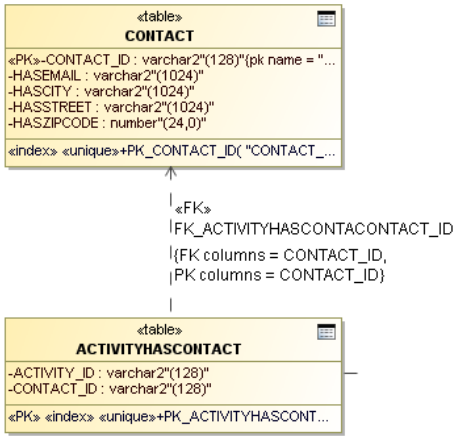
	DESTINATION_ID	DESTINATION_ID1
1	Sydney	BondiBeach
2	Sydney	CurrawongBeach

```
select * from destinationhasammodation
```

	DESTINATION_ID	ACCOMMODATION_ID
1	Sydney	FourSeasons

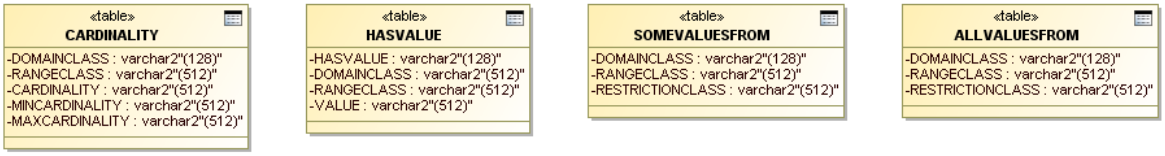
Paveikslėlis Nr. 40. RDB lentelių, kurios išreiškia nefunkcines objektų savybes, turinys

Duomenų tipų savybės išreiškiamos paprastai – kuriami stulpeliai klasių lentelėse, kurioms jos taikomos, panaudojant informaciją apie OWL duomenų tipą (Paveikslėlis Nr. 41). Pavyzdyje pateikta CONTACT klasės lentelė, kurioje matomi laukai sukurti pagal duomenų tipų savybes (HASEMAIL, HASCITY, HASSTREET ir HASZIPCODE).



Paveikslėlis Nr. 41. RDB lentelių struktūra, kuria išreiškiamos duomenų tipų savybės

Metaduomenų lentelių struktūra yra visuomet vienoda.



Paveikslėlis Nr. 42. RDB lentelių struktūra, kurioje saugojami metaduomenys

Peržvelgdami šiose lentelėse esančius duomenis (Paveikslėlis Nr. 43), matome, kad jie pilnai atitinka visus nagrinėjamos ontologijos apribojimus (Lentelė Nr. 3).

Select somevaluesfrom | Select hasvalue | Select cardinality

	DOMAINCLASS	RANGECLASS	RESTRICTIONCLASS
1	BudgetAccommodation	AccommodationRating	{OneStarRating TwoStarRating}
2	BackpackersDestination	Accommodation	BudgetAccommodation
3	BackpackersDestination	Activity	Sports or Adventure
4	BudgetHotelDestination	Accommodation	BudgetAccommodation and Hotel
5	RetireeDestination	Accommodation	hasRating has ThreeStarRating
6	RetireeDestination	Activity	Sightseeing
7	NationalPark	Activity	Hiking
8	NationalPark	Accommodation	Campground
9	City	Accommodation	LuxuryHotel
10	Capital	Activity	Museums

Select somevaluesfrom | Select hasvalue | Select cardinality

	HASVALUE	DOMAINCLASS	RANGECLASS	VALUE
1	hasRating has OneStarRating	Campground	AccommodationRating	OneStarRating
2	hasRating has ThreeStarRating	LuxuryHotel	AccommodationRating	ThreeStarRating

Select somevaluesfrom | Select hasvalue | Select cardinality

	DOMAINCLASS	RANGECLASS	CARDINALITY	MINCARDINALITY	MAXCARDINALITY
1	FamilyDestination	Accommodation		1	
2	FamilyDestination	Activity		2	

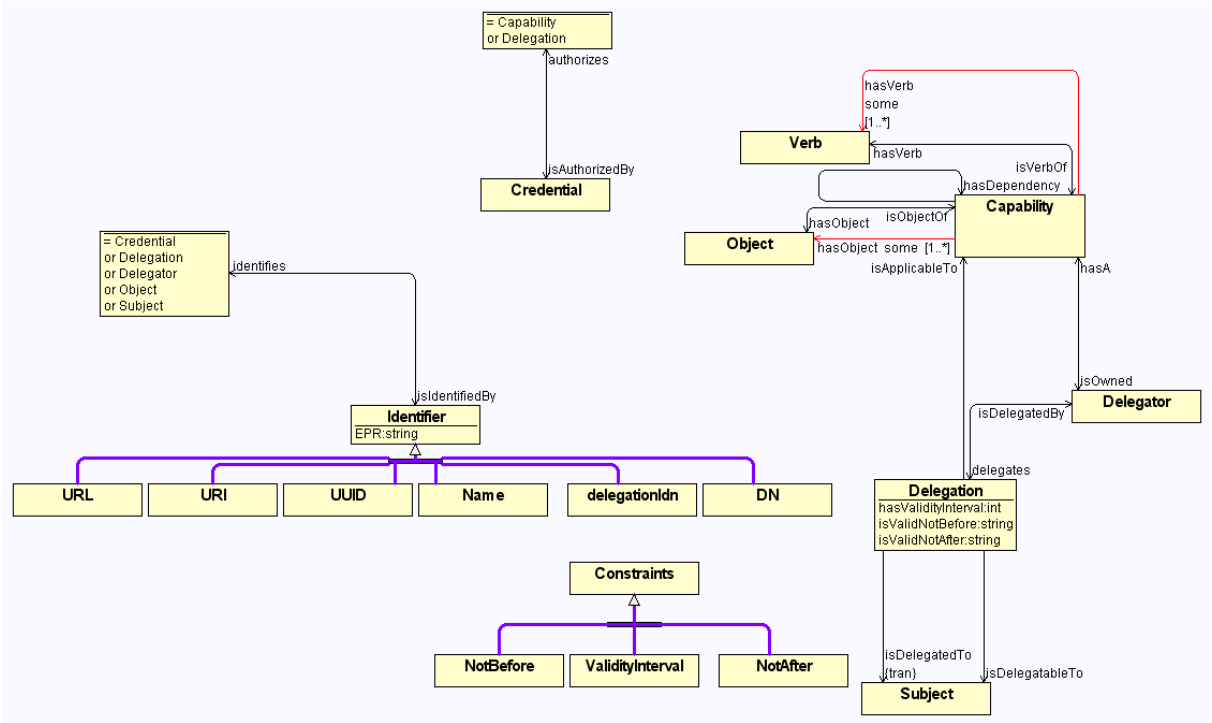
Paveikslėlis Nr. 43. RDB metaduomenų lentelių turinys

5.3. Eksperimentas

5.3.1. Internete patalpintų ontologijų transformavimo tyrimas

Eksperimento tikslas yra nustatyti ar OWL ontologijų transformavimo į RDB schemas algoritmą realizuojantis įrankis geba teisingai atlikti transformavimą su įvairiomis internete patalpintomis ontologijomis. Visos tiriamos ontologijos yra Protege ontologijų bibliotekoje [17] .

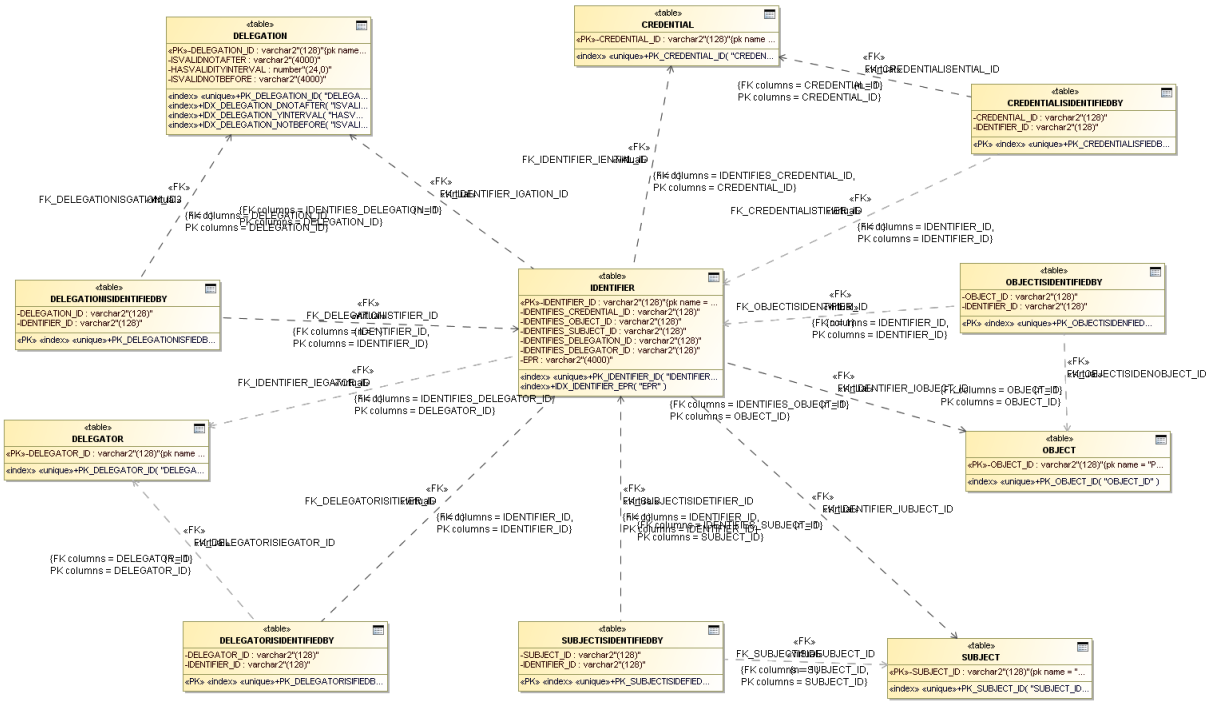
Pasirinkta Delegation.owl ontologija (Paveikslėlis Nr. 44). Tai yra ontologija, kuri apibūdina delegavimo sąvokas tinklo skaičiavimų (angl. grid computing) kontekste.



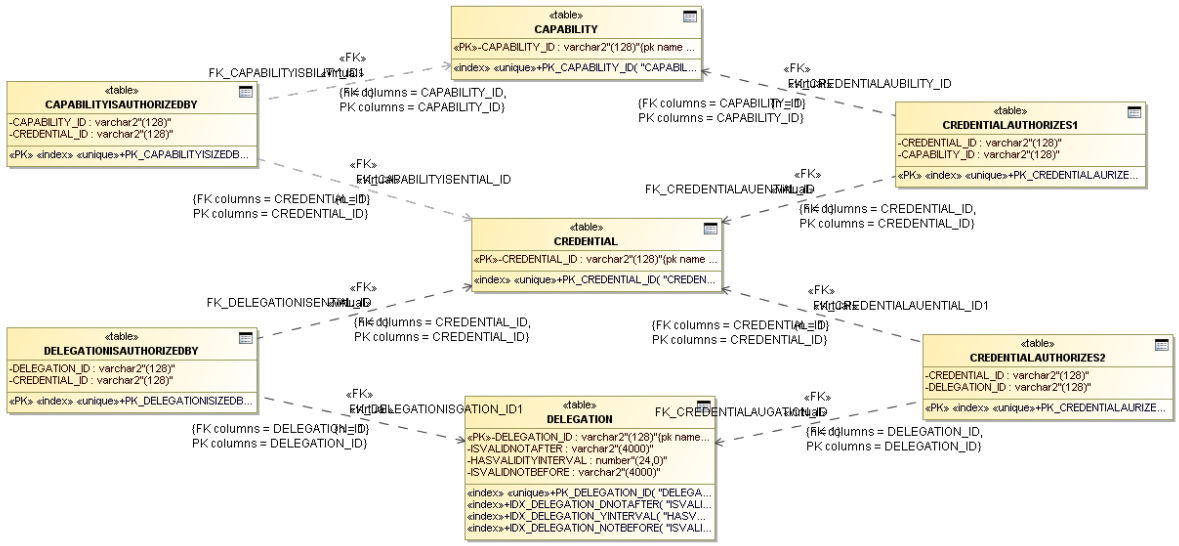
Paveikslėlis Nr. 44. Delegation OWL ontologijos schema

Įdomesni šios ontologijos struktūroje esantys aspektai atliekant transformaciją yra objektų savybių rdfs:domain ir rdfs:range klasių sąjungos. Tokios savybės yra identifies (rdfs:domain klasė yra Identifier, o rdfs:range klasės gali būti Credential, Delegation, Delegator, Object arba Subject) ir jos priešinga isIdentifiedBy (rdfs:domain klasės gali būti Credential, Delegation, Delegator, Object arba Subject, o rdfs:range yra klasė Identifier); taip pat authorizes ir isAuthorizedBy objektų savybės. Nors objektų savybė identifies yra funkcinė ir jai atvaizduoti užtenka sukurti laukus su išoriniais raktais, tačiau atvirkštinė isIdentifiedBy nėra funkcinė ir buvo sukurtos tarpinės lentelės (Paveikslėlis Nr. 45) – nors taip RDB dažniausiai nedaroma ir kuriamas ryšys tik vienoje pusėje, tačiau siekiant išsaugoti kuo daugiau ontologijos informacijos – kuriami dvipusiai ryšiai. Savybės authorizes bei

isAuthorizedBy nėra funkcinės dėl to joms atvaizduoti buvo sukurtos 4 lentelės (Paveikslėlis Nr. 46).



Paveikslėlis Nr. 45. Identifies ir isIdentifiedBy ryšiai po transformacijos



Paveikslėlis Nr. 46. Authorizes ir isAuthorizedBy ryšiai po transformacijos

Iš viso po transformacijos RDB schemą sudaro 43 lentelės: 18 lentelių klasėms (CAPABILITY, OBJECT, CONSTRAINTS, VERB, DELEGATOR, DELEGATION, SUBJECT, IDENTIFIER, CREDENTIAL, NOTBEFORE, NOTAFTER, VALIDITYINTERVAL, DELEGATIONIDN, URI, NAME, UUID, DN, URL), 21 lentelė objektų savybėms (CAPABILITYHASVERB, CAPABILITYHASOBJECT, CAPABILITYISOWNED, DELEGATORHASA, DELEGATORDELEGATES, DELEGATIONISDELEGATEDBY, CAPABILITYISAUTHORIZEDBY, DELEGATIONISAUTHORIZEDBY, CREDENTIALAUTHORIZES1, CREDENTIALAUTHORIZES2,

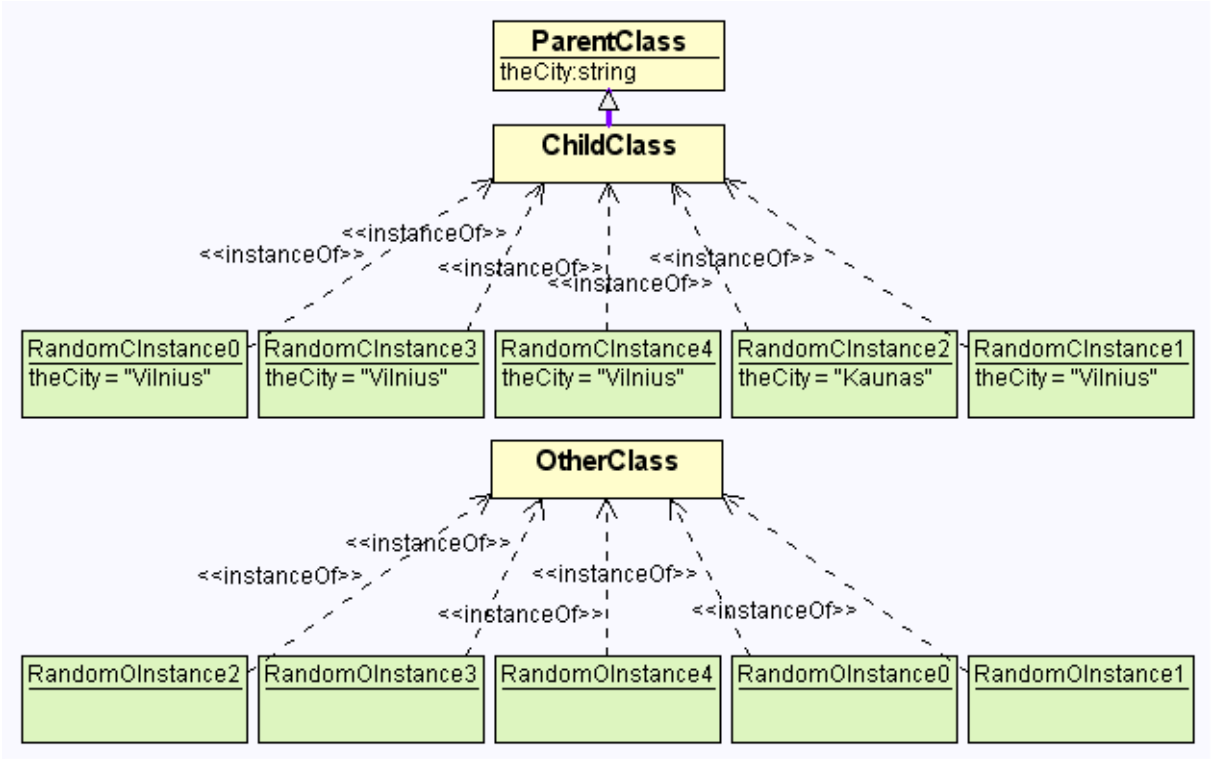
DELEGATIONISAPPLICABLETO, OBJECTISOBJECTOF, CAPABILITYHASDEPENDENCY, CREDENTIALISIDENTIFIEDBY, OBJECTISIDENTIFIEDBY, SUBJECTISIDENTIFIEDBY, DELEGATIONISIDENTIFIEDBY, DELEGATORISIDENTIFIEDBY, DELEGATIONISDELEGATABLETO, VERBISVERBOF, DELEGATIONISDELEGATEDTO) ir 4 lentelės apribojimams (ALLVALUESFROM, SOMEVALUESFROM, HASVALUE, CARDINALITY).
Taigi, tyrimas parodė, kad visa ontologijos informacija buvo transformuota.

5.3.2. Paieškos greitaveikos tyrimas

Eksperimento tikslas yra nustatyti informacijos paieškos greitaveikos skirtumus naudojant senas OWL ontologijų saugojimo priemones ir naudojant RDB po transformacijos. Šio empirinio tyrimo tipas yra eksperimentas.
Eksperimentas atliekamas vieno žmogaus, laboratorinėmis sąlygomis naudojant Protege sistemą ir į ją integruojamomis papildomomis priemonėmis duomenų generavimui.
Eksperimento metu naudojama statinė OWL ontologijos klasių struktūra (Paveikslėlis Nr. 47) ir vienoda naudojama informacijos paieškos užklausa (Lentelė Nr. 4).
Hipotezės:
H_0: $t_{senas} \leq t_{naujas}$
$H_1(H_a)$: $t_{senas} > t_{naujas}$
Nepriklausomi kintamieji yra:

- Bandomi paieškos metodai
    - Senas metodas – paieška naudojant Protege sistemos SPARQL, kai OWL ontologija saugojama faile
    - Senas metodas – paieška naudojant SPARQL, kai OWL ontologija saugojama vienos lentelės duomenų bazėje
    - Naujas metodas – paieška naudojant Protege sistemos Oracle SQL užklausas transformavimo algoritmo sukurtoje duomenų bazėje
- Klasių egzempliorių kiekis

Priklausomas kintamasis yra informacijos paieškos laikas



Paveikslėlis Nr. 47. Naudojama Statinė ontologijos klasių struktūra

Lentelė Nr. 4. Naudojamos paieškos užklausos, kurių rezultatai sutampa

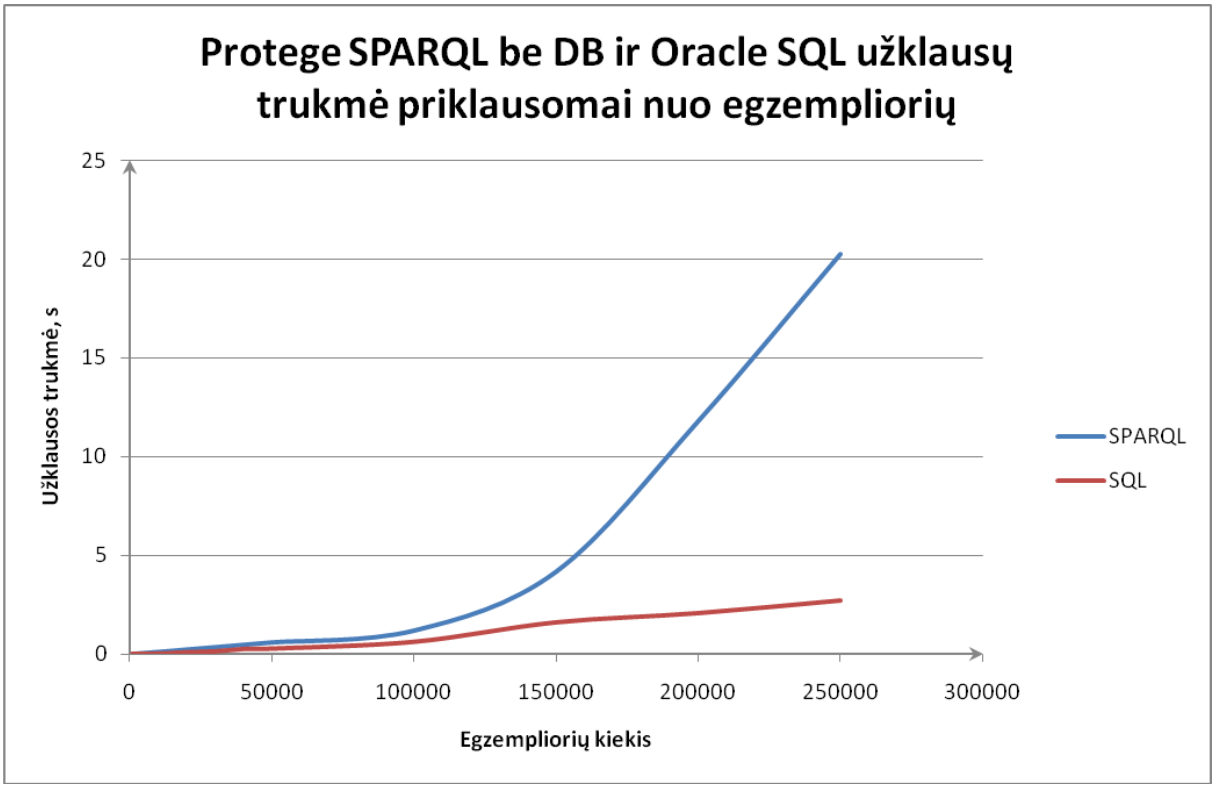
Protege SPARQL užklausa	Oracle SQL užklausa

`SELECT ?c`
`WHERE {`
`?c :theCity ?k .`
`FILTER ((?k) != "Vilnius")`
`}`
	`select cc.parentclass_id`
	`from parentclass pc, childclass cc`
	`where cc.parentclass_id = pc.parentclass_id`
	`and pc.theCity <> 'Vilnius'`

Eksperimentas atliekamas pateiktoje ontologijoje generuojant „ChildClass“ ir „OtherClass“ klasių egzempliorius (Paveikslėlis Nr. 47). Klasei ParentClass priskiriama „theCity“ duomenų tipo savybė – tokiu būdu ją įgyja ir „ChildClass“. Visiems „ChildClass“ klasės egzemplioriams priskiriama „theCity“ vienokia reikšmė (pavyzdžiui, „Vilnius“), o vienam – bet koks kitoks (pavyzdžiui, „Kaunas“). Tokiu būdu pateiktos užklausos (Lentelė Nr. 4) randa vieną įrašą visoje sugeneruotoje ontologijoje.

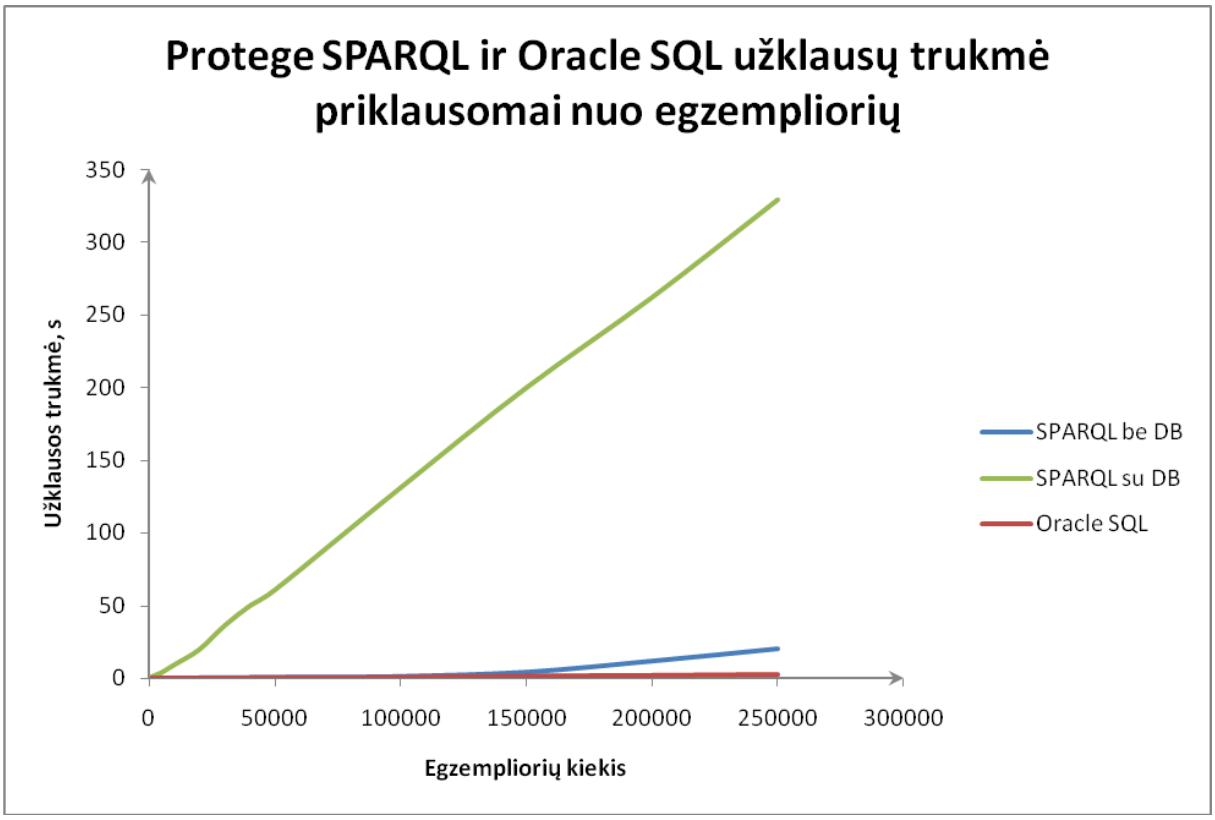
Sugeneravus ir išsaugojus faile ontologiją eksperimentui, atliekama SPARQL užklausa naudojant Protege sistemą ir nustatomas jos vykdymo laikas. Tuomet Protege sistemos priemonėmis ontologija išsaugojama duomenų bazėje ir įvykdoma ta pati SPARQL užklausa siekiant nustatyti darbo laiką naudojant šį Protege sistemos saugojimo būdą. Galiausiai ontologija yra transformuojama į Oracle duomenų bazę panaudojant šiame darbe aprašytą algoritmą ir nustatomas laikas, kurio reikia norint surasti tą pačią informaciją duomenų bazėje. Eksperimentas kartojamas su vis didesniu egzempliorių skaičiumi ir fiksuojami skirtingi paieškos užklausų laikai (žr. priedus - Lentelė Nr. 6 ir Lentelė Nr. 7).

Atlikus šį eksperimentą nustatyta, kad Oracle SQL veikia greičiau nei SPARQL naudodama OWL failą (Paveikslėlis Nr. 48), o kai visi egzemplioriai nebetelpa atmintyje – matomas stiprus SPARQL savybių suprastėjimas, nes operacinė sistema pradeda naudoti virtualią atmintį kietajame diske. Tačiau, kadangi klasių struktūra yra nekintama, suprastėjus paieškos laikui jo priklausomybė nuo egzempliorių kiekio išlieka tiesiška.



Paveikslėlis Nr. 48. Protege SPARQL ir Oracle SQL palyginimas

Taip pat atliekant eksperimentą nustatyta, kad Protege sistemai saugant ontologiją vienos lentelės duomenų bazės schemoje, paieška yra nesulyginamai lėtesnė (Paveikslėlis Nr. 49). Nors šis Protege ontologijų saugojimo būdas leidžia turėti ontologiją, kuri gerokai viršyja kompiuterio atmintį, tačiau paieškos greitaveika yra problema.



Paveikslėlis Nr. 49. Protege SPARQL su duomenų baze palyginimas su kitais būdais

Gautų rezultatų analizė parodė, kad naudojant užklausas transformuotoje RDB schemoje rezultatai gaunami greičiau, nei esamais Protege ontologijų saugojimo metodais – tiek faile, tiek vienos lentelės duomenų bazėje.

Taigi, hipotezė $H_1(H_a)$: $t_{senas} > t_{naujas}$ patvirtinta.

6. IŠVADOS

1. Literatūros šaltinių analizė rodo, kad didelių ontologijų saugojimas yra problema, o DBVS panaudojimas leidžia saugoti, ieškoti ir manipuliuoti OWL ontologijų duomenimis efektyviau
2. Esamų ontologijų saugojimo RDB metodų analizė parodė, kad jie turi trūkumų ir neišnaudoja visų RDB galimybių.
3. Ištirtas algoritmas transformuoja svarbiausią OWL ontologijų informaciją į RDB schemas.
4. Atliktas eksperimentas parodė, kad sukurtu įrankiu projektuotojas galės transformuoti OWL ontologiją Protégé sistemoje į RDB schemą ir vykdyti ontologijų užklausas efektyviau, nei nenaudojant RDB arba naudojant esamą Protege algoritmą.
5. Taip pat atliktas eksperimentas parodė, kad taikant RDB galima efektyviai vykdyti užklausas didelėse ontologijose, kurios netelpa į užklausų procesoriaus atmintį.

7. LITERATŪRA

[1] OWL Web Ontology Language Overview. http://www.w3.org/TR/owl-features/, 2009 03 08

[2] Maria del Mar Roldan-Garcia, Joaquin J. Molina-Castro, Jose F. Aldana-Montes. ECQ: A Simple Query Language for the Semantic Web. 19th International Conference on Database and Expert Systems Application, 190-194

[3] Boris Motik, Ian Horrocks, Ulrike Sattler. Bridging the Gap Between OWL and Relational Databases.

[4] Nadine Cullot, Caristine Parent, Stefano Spaccapietra, and Christelle Vangenot. Ontologines: A contribution to the DL/DB debate.

[5] Sandra Geisler, Andreas Brauers, Christoph Quix, Anke Schmeink. Ontology-based system for clinical trial data management. IEE Benelux EMBS Symposium, 2007 December 6-7.

[6] Sean Bechhofer, Ian Horrocks, Daniele Turi. The OWL Instance Store: System Description. Proceedings CADE-20; Lecture Notes in Computer Scrience, 2005.

[7] Martin Šeleng, Michal Laclavík, Zoltán Balogh, Ladislav Hluchý. RDB2Onto: Approach for creating semantic metadata from relational database data.

[8] Justas Trinkūnas, Olegas Vasilecas. A graph oriented model for ontology transformation into conceptual data model. Information technology and control, 2007, Vol. 36, No. 1A, 126-132.

[9] Zhengxiang Pan, Jeff Heflin. DLDB: Extending Relational Databases to Support Semantic Web Queries.

[10] Juhnyoung Lee, Richard Goodwin. Ontology management for large-scale enterprise systems. Electronic Commerce and Applications 5, 2006, 2-15.

[11] R. Fikes, P. Hayes, I. Horrocks, OWL-QL – A Language for Deductive Query Answering on the Semantic Web, Knowledge Systems Laboratory, Stanford University, Stanford, CA, 2003.

[12] Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. http://www.omg.org/spec/QVT/1.0/PDF, April 2008

[13] Richard Goodwin, Juhnyoung Lee, George A. Mikaila, Ioana Stanoi. Leveraging Relational Database Systems for Large-Scale Ontology Management. CIDR Conference, 2005.

[14] Irina Astrova. Towards the Semantic Web – An Approach to Reverse Engineering of Relational Databases to Ontologines. Advances in Databases and Information Systems: proceedings of the 9th East-European Conference, ADBIS 2005, Talin, September 12-15, 2005. 111-122.

[15] Ernestas Vyšniauskas, Lina Nemuraitė. Transforming Ontology Representation from OWL to Relational Database. Information technology and control, 2006, Vol.35, No. 3A, 333-343.

[16] Ernestas Vyšniauskas, Lina Nemuraitė. Mapping of OWL ontology concepts to RDB schemas.

[17] Protege Ontology Library – Protege Wiki. http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library#OWL_ontologies, 2010 05 09

8. TERMINŲ IR SANTRUMPŲ ŽODYNAS

Lentelė Nr. 5. Terminų ir santrumpų žodynas

OWL	Žinių atvaizdavimo kalba ontologijoms kurti, patvirtinta World Wide Web konsorciumo
IT	Informacinės technologijos
DBVS	Duomenų bazių valdymo sistema
SQL	Struktūrizuota užklausų kalba (angl. Structured Query Language) - populiariausia iš šiuo metu naudojamų kalbų, skirtų aprašyti duomenis ir manipuliuoti jais reliacinių duomenų bazių valdymo sistemose
JDK	Java programų kūrimo įrankių komplektas (angl. Java Development Kit)
Java	Bendros paskirties, aukšto lygio, objektiškai orientuota, nuo platformos nepriklausoma programavimo kalba sukurta Sun Microsystems firmoje.
ES	Europos Sąjunga
PĮ	Programinė įranga
UML	Unifikuota modeliavimo kalba (angl. Unified Modeling Language)
RUP	Kartotinio programinės įrangos kūrimo metodika (angl. Rational Unified Process), sukurta įmonės Rational Software
CASE	Automatizuotas kompiuterinis programinės įrangos projektavimas (angl. Computer-Aided Software Engineering)
RDB	Reliacinė duomenų bazė
Protégé	Atviro kodo ontologijų kūrimo sistema
API	Application Programming Interface. Tai yra sąsaja, kuria viena programa sąveikauja su kita programa
Protege Core API	Protege sukurta aplinka, kuria naudojantis galima kurti paprastas programos funkcijas ir vartotojo sąsają
Protege OWL API	Protege sukurta aplinka, kuria naudojantis galima pasiekti šia sistema sukurtas ontologijas kuriant sistemos vidinius įrankius ar išorines programas
OWL2RDB	OWL ontologijų transformavimo į reliacinės duomenų bazės schemą įrankis

9. PRIEDAI

9.1. Užklausų trukmės eksperimento rezultatai

Lentelė Nr. 6. SPARQL užklausų vykdymo rezultatai

egz. nr.	1000	2000	3000	4000	5000	10000	20000	30000	40000	50000	100000	150000	200000	250000
1	15	25	36	48	59	117	237	349	489	590	1206	3631	12502	14296
2	16	24	35	48	57	119	238	360	503	593	1207	6586	16032	15038
3	17	26	36	48	57	118	243	361	495	586	1182	4219	7279	13810
4	18	24	35	48	62	118	232	363	478	585	1210	2279	12203	16449
5	14	25	35	48	60	115	231	345	472	596	1210	3570	10174	17645
6	13	24	35	48	58	114	237	340	467	582	1183	3541	12665	17540
7	13	25	36	49	59	114	234	358	483	582	1188	4169	9879	13524
8	14	24	40	56	58	118	237	362	484	590	1182	4415	9414	17347
9	14	25	35	49	62	123	235	355	484	606	1203	4303	23546	34084
10	13	25	36	48	59	114	238	355	475	579	1190	3630	9622	18430
11	13	25	35	49	59	115	235	350	467	600	1201	3381	8964	19150
12	13	24	36	48	58	120	236	351	466	588	1181	6886	9471	15698
13	14	25	36	49	58	116	244	353	470	590	1188	2661		28862
14	14	26	35	48	59	118	234	361	474	596	1189	4499		17139
15	13	25	35	47	58	115	231	365	480	585	1199	8214		50218
16	13	25	36	47	58	114	243	345	467	602	1195	6916		22979
17	14	25	35	49	60	113	240	359	482	604	1230	3869		21700
18	14	24	43	48	58	115	231	370	479	586	1197	3614		11559
19	13	25	36	49	59	117	236	348	473	599	1192	3901		19560
20	13	25	35	48	58	114	231	352	460	597	1186	3457		19441
21	13	24	35	48	59	114	236	360	464	676	1196	7595		18891
22	13	24	35	50	58	116	239	340	483	601	1185	2991		22751
23	14	25	36	50	58	118	230	356	478	579	1223	2970		
24	14	26	42	49	58	119	237	368	489	610	1193	3741		
25	15	26	34	48	57	116	232	347	478	601	1183	2995		
26	13	24	36	48	59	116	227	348	461	611	1167	2858		
27	14	25	36	48	57	114	241	350	471	580	1212	3811		
28	14	24	40	47	58	116	235	346	474	603	1195	3216		
29	13	25	36	49	59	117	236	351	483	594	1204	3335		
30	13	25	35	49	60	117	232	365	475	607	1175			
t, ms	13,9	24,7	36,3	48,6	58,7	116,3	235,5	354,4	476,8	596,6	1195,1	4181,1	11812,5	20277,8

Lentelė Nr. 7. SQL užklausų vykdymo rezultatai

egz. nr.	1000	2000	3000	4000	5000	10000	20000	30000	40000	50000	100000	150000	200000	250000
1	12	8	10	19	15	26	79	149	134	244	800	1602	1887	2669
2	8	8	9	29	15	23	63	138	257	278	610	1685	2096	3302
3	7	8	9	18	16	21	102	109	346	373	611	1671	2071	2894
4	6	8	11	25	21	22	63	92	366	189	549	1564	2080	3097
5	6	8	10	11	15	24	93	145	247	177	731	1602	2148	2881
6	7	8	10	17	17	23	69	152	200	214	622	1616	2188	2673
7	7	8	9	11	18	24	75	262	151	249	535	1558	2013	2096
8	6	8	10	21	19	23	97	159	196	229	709	1657	2071	2369
9	6	8	33	14	23	25	118	135	299	159	594	1688	2092	2254
10	7	8	9	21	15	22	108	127	297	277	667	1639	2159	3078
11	7	8	10	12	15	25	88	97	188	292	648	1665	2025	2237
12	7	8	10	18	15	24	104	133	295	209	548	1613	2175	2342
13	6	8	10	11	13	25	123	123	275	316	546	1488		2644
14	6	8	10	18	15	26	66	119	194	329	444	1517		2463
15	6	8	10	11	25	25	77	125	381	194	641	1687		2598
16	6	8	10	14	18	25	44	112	258	223	854	1613		2522
17	6	8	10	12	16	23	299	98	219	313	591	1700		3268
18	7	8	9	15	22	24	115	148	273	177	553	1602		4562
19	7	9	9	11	18	24	99	126	242	343	541	1608		2514
20	6	10	10	18	15	25	62	158	256	366	581	1559		2444
21	7	8	9	11	19	21	110	184	201	400	596	1569		2353
22	6	8	10	26	19	45	79	120	283	357	598	1660		2700
23	6	8	10	11	18	24	72	140	152	373	641	1629		
24	6	8	14	21	13	25	92	124	309	375	665	1643		
25	6	9	10	12	19	24	117	162	358	309	482	1585		
26	6	8	10	22	24	46	75	160	429	180	664	1634		
27	6	9	9	12	33	25	110	180	170	345	505	1547		
28	7	8	12	21	20	26	113	111	201	366	768	1604		
29	6	8	9	11	18	26	73	126	329	362	813	1621		
30	6	8	10	20	27	23	74	144	292	178	665			
t, ms	6,7	8,0	10,7	16,4	18,6	25,6	95,3	138,6	259,9	279,8	625,7	1614,7	2083,8	2725,4