KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Andrius Kondratas

Įrankis *OWL 2* ontologijoms transformuoti į reliacines duomenų bazes

Magistro darbas

Darbo vadovas
prof. Lina Nemuraitė

Kaunas, 2012

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Andrius Kondratas

Įrankis *OWL 2* ontologijoms transformuoti į reliacines duomenų bazes

Magistro darbas

Recenzentas
doc. dr. Tomas Blažauskas
2012-05-28

Vadovas
prof. Lina Nemuraitė
2012-05-28

Atliko
IFME-0/4 gr. stud.
Andrius Kondratas
2012-05-28

Kaunas, 2012

Turinys

Summary

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.

This work covers analysis of the process and algorithms, which were proposed in *OWL2ToRDB* method for transformation of domain ontology, described in *OWL*, to relational database. According this algorithm, ontology classes are mapped to relational tables, properties to relations and attributes, and constraints – to metadata. The proposed algorithm is capable to transform all *OWL*2 Lite and part of *OWL*2 *DL* syntax. The prototypical tool, performing transformations, has been implemented as add-in for the ontology development tool "*Protégé*". The methodology of transformation is illustrated with an example.

Keywords: information system, ontology, transformation, relational database, knowledge base, *OWL* 2.

Terminų ir santrumpų žodynėlis

DBVS	Duomenų bazių valdymo sistema. Programinės įrangos rinkinys skirtas organizuoti informacijai duomenų bazėje
Duomenys	Faktai ar teiginiai, kurie yra ar gali būti surinkti, išsaugoti, apdoroti, tačiau nėra organizuoti ar pateikti kontekste
Informacija	Duomenų prasmė, reikšmė, numatyta interpretacija. Duomenų organizavimas, susiejimas asociacijomis, apribojimas, sudarantis galimybę juos panaudoti
Informacinė sistema	Automatizuota ar rankinė sistema, apimanti priemones ir komponentus informacijos surinkimui, apdorojimui, saugojimui, persiuntimui, pateikimui, platinimui, panaudojimui
Modelis	Supaprastintas sudėtingos esybės ar proceso aprašas (pvz. formulių rinkiniu), atvaizdas ar pan.
Ontologija	Tam tikros srities bendrai naudojamos sąvokų/konceptų, esybių tipų, jų tarpusavio priklausomybių, sąryšių, aksiomų, dėsningumų ir kt. visumos formalus aprašas.
OWL (angl. *Web Ontology Language*)	Ontologijos aprašymui skirta kalba
RDB	Reliacinė duomenų bazė
RDBVS	Reliacinių duomenų bazių valdymo sistema
RDF (angl. *Resource Description Framework*)	Išteklių aprašymo kalba
SQL (angl. *Structured Query Language*)	Struktūrinių užklausų kalba
XML(angl. *Extensible Markup Language*)	Išplečiamoji žymėjimo kalba, naudojama aprašyti duomenims

Lentelių sąrašas

Paveikslėlių sąrašas

1. Įvadas

Ontologijos vis plačiau naudojamos įvairiuose taikymuose: verslo procesų ir informacijos integravime, paieškoje ir tvarkyme. Tokie taikymai reikalauja 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 tinkle ir įmonėse.

Esami ontologijų saugojimo RDB metodai turi trūkumų [9]:

– dalis esamų metodų išsaugo tik ontologijų egzempliorius, o geriausiu atveju tik klases ir savybes;

– kita dalis metodų saugo visą ontologiją kartu su egzemplioriais. Šiuos metodus galima skirstyti į dvi dalis: vieni saugo ontologiją su egzemplioriais vienoje lentelėje, dėl to augant duomenų apimtims veikimas labai sulėtėja. Kiti metodai saugo ontologijos metamodelį, tačiau šiuo atveju neišnaudojami RDB privalumai (greitas informacijos išrinkimas), taip pat tokio vaizdavimo negalima pritaikyti jau esamoms duomenų bazėms;

– Informacijos sistemų katedroje sukurtame *OWLToRDB* prototipe [14, 13] taikomas transformavimo metodas, kai dalis ontologijos tiesiogiai vaizduojama RDB schema, taip išnaudojant RDB privalumus, o ontologijos elementai, kurie tiesiogiai nepavaizduojami RDB, transformuojami į metaduomenų lenteles. Tačiau prototipas neapima visų ontologijos savybių ir yra skirtas pirmai *OWL* versijai, o šiuo metu atsirado *OWL 2* versija [8, 5] ir tikslinga pereiti prie šios versijos.

Šiame darbe tiriamas informacinių sistemų katedroje sukurtas algoritmas *OWLToRDB*, kuris atlieka dalykinės srities ontologijos aprašo transformaciją į reliacinę duomenų bazę. Šiam metodui buvo sukurtas patobulintas realizacijos prototipas įvertinant *OWL 2* standartą, taikant hibridinį vaizdavimo būdą ir siekiant pavaizduoti didesnę ontologijos konceptų aibę. Pagrindinis metodo kokybės kriterijus – informacijos nepraradimas. Tai reiškia, kad bus siekiama transformuoti ontologiją ir jos egzempliorius taip, kad iš reliacinės duomenų bazės būtų galima atstatyti visą ontologiją ir jos egzempliorius.

Algoritmo veikimas eksperimento metu buvo tiriamas naudojant patobulintą realizacijos prototipą. Tyrimui sudaryta pavyzdinė ontologija, kuri atitinka ontologijų projektavimo taisykles, bei atsitiktinai parinktos kitos, žiniatinklyje publikuojamos ontologijos. Atliekamo eksperimento tikslas – metodo kokybės įvertinimas informacijos praradimo aspektu transformavimo metu. Šio kriterijaus įvertinimui buvo užrašomi: ontologijos elementų skaičius, transformuotų į reliacinę duomenų bazę elementų skaičius.

Testuojant sukurtą *OWL2ToRDB* transformavimo įrankį, paaiškėjo, kad jis gerai veikia su taisyklingomis ontologijomis, atitinkančiomis ontologijų kūrimo reikalavimus, papildytus reikalavimais transformavimui į RDB. Tačiau dauguma praktikoje sutinkamų ontologijų neatitinka šių reikalavimų, todėl *OWL2ToRDB* įrankio pagalba transformuojama nuo 40 iki 99 procentų jų elementų ir tik pavyzdinė ontologija transformuojama visa. Dažniausia informacijos praradimo priežastis – nenurodyta objektų savybių apibrėžimo (angl. *domain*) arba kitimo sritis (angl. *range*).

Darbo naujumas. Šiame darbe sudaryti algoritmai *OWL 2* transformuoti į reliacines duomenų bazes, taip išplečiant *OWLToRDB* metodą, bei aprašyti reikalavimai, kuriuos turi tenkinti ontologijos, kurias norima saugoti reliacinėse duomenų bazėse.

Patobulintą *OWL2ToRDB* hibridinio ontologijų saugojimo metodo prototipą tikslinga taikyti tada, kai esamas ar kuriamas ontologijas norima susieti su taikomosiomis programomis, naudojančiomis reliacines duomenų bazes.Tuomet tikslinga ne tik kurti taisyklingas ontologijas, atitinkančias ontologijų kūrimo reikalavimus, bet ir pritaikyti prie reikalavimų, keliamų saugojimui RDB.

Darbo struktūra:

Skyriuje „Ontologijų transformacijų tyrimo sritis ir analizė“ suformuluoti darbo tikslai, apibrėžta tyrimo sritis, objektas ir problema. Pateikiamas ontologijos apibrėžimas ir pagrindinės sąvokos. Tolesniuose skyriuose aprašoma atlikta ontologijos aprašymo kalbų analizė, bei literatūros šaltiniuose pateiktų sprendimų problemai spręsti apžvalga. Nustatytas dalykinės srities žinių bazės kūrimo procesas ir galimas architektūros modelis. Atlikta transformacijos sistemos vartotojų analizė, iškeltas projekto tikslas, nustatyti kokybės kriterijai, pasirinkti projektavimo metodai ir priemonės. Skyriaus pabaigoje suformuluotos analizės išvados.

Skyriuje „Transformavimo sistemos reikalavimai“ nustatyti transformavimo įrankio nefunkciniai reikalavimai, išskirti sistemos panaudojimo atvejai ir sudarytos detalios jų specifikacijos.

Skyriuje „*OWL2ToRDB* metodas ir algoritmai“ pateikiami ontologijos klasių, objektinių ir duomenų tipų savybių transformavimo į RDB algoritmai. Sekančiuose skyriuose pateikiamas sukurtas ontologijos aprašo transformavimo įrankio programinės realizacijos projektas bei aprašyta prototipo realizacija.

Skyriuje „Eksperimentinis tyrimas ir darbo rezultatų įvertinimas“ pateikiami pavyzdinės bei praktikoje sutinkamų ontologijų transformavimo rezultatai bei analizė.

Dalyvauta konferencijoje, kurioje pristatytas su šio darbo tematika susijęs straipsnis (straipsnio medžiaga pateikta priede):

- Informacinės technologijos 2012 (17-toji tarpuniversitetinė doktorantų ir magistrantų konferencija), VU KHF, skaitytas straipsnis „Įrankis *OWL 2* ontologijoms transformuoti į reliacines duomenų bazes“;

2. Ontologijų transformacijų tyrimo sritis ir analizė

2.1. Tyrimo sritis, tikslas ir uždaviniai

Šio darbo tyrimo sritis yra ontologijų inžinerija, bei ontologijų atvaizdavimas į informacinių sistemų inžinerijos modelius. Tyrimo objektas – *OWL* 2 ontologijos atvaizdavimo į reliacinę duomenų bazę metodika

Šis tyrimas turi padėti spręsti didelių ontologijų saugojimo ir prieigos problemas, kadangi augant ontologijos egzempliorių skaičiui, dideles ontologijas sunku saugoti failuose, o jų prieiga labai sulėtėja.

Šiame darbe numatoma plėsti *OWLToRDB* prototipą įvertinant *OWL* 2 standartą, taikant hibridinį vaizdavimo būdą ir siekiant pavaizduoti didesnę ontologijos konceptų aibę. Pagrindinis metodo kokybės kriterijus – informacijos nepraradimas. Tai reiškia, kad bus siekiama transformuoti ontologiją ir jos egzempliorius taip, kad iš reliacinės duomenų bazės būtų galima atstatyti visą ontologiją ir jos egzempliorius.

Tobulinamo ontologijų modeliavimo ir atvaizdavimo įrankio vartotojas bus kliento informacinės sistemos duomenų bazės projektuotojas. Jis naudos šį įrankį tam, kad neformalias produktų specifikacijas galėtų ontologijų pagalba aprašyti formaliai, taip pat gautus formalius aprašus atvaizduoti į reliacinę duomenų bazę.

Šio **tyrimo tikslas**: patobulinti ontologijų saugojimo reliacinėse duomenų bazėse galimybes, išplečiant ir realizuojant algoritmus *OWL* 2 ontologijų aprašams transformuoti į duomenų bazių schemas.

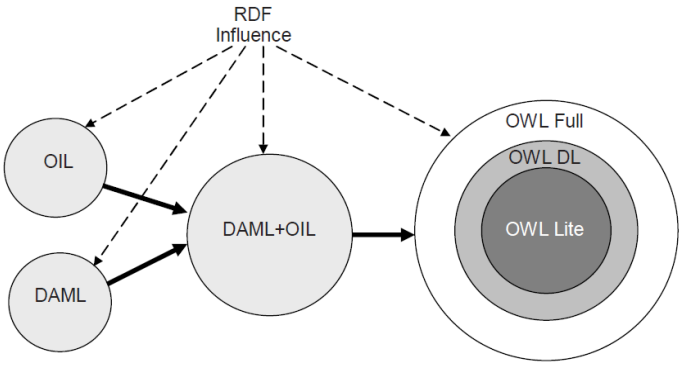
Tyrimo uždaviniai:

1. Atlikti esamos situacijos analizę:
   − išanalizuoti *OWL 2* ir RDB konceptus, savybes, metamodelius, jų skirtumus ir panašumus;
   − išanalizuoti esamus ontologijų transformavimo į RDB metodus;
   − išanalizuoti ontologijų kūrimo ir transformavimo į RDB įrankius;
   − apsibrėžti esamų transformavimo įrankių patobulinimo kriterijus, pasirinkti realizavimo priemones.
2. Sudaryti patobulintą transformavimo algoritmą ir suprojektuoti *OWL2ToRDB* transformavimo įrankį;
3. Realizuoti ir ištestuoti patobulintą *OWL2ToRDB* įrankį;
4. Atlikti eksperimentą ir įvertinti sukurtą metodą bei jį realizuojantį įrankį.

2.2. *OWL* ontologijos kalbos analizė

OWL (angl. *Web Ontology Language*). yra nauja formali kalba, sukurta ontologijų atvaizdavimui semantiniame tinkle [8]. *OWL* skirta naudoti tuomet, kai esamas formalias turinio pateikimo specifikacijas žiniatinklyje siekiama papildyti informacijos interpretavimo galimybėmis. *OWL* naudojama norint aiškiai pateikti žodynuose esančias išraiškas, jų prasmę bei tarpusavio ryšius. Šios išraiškos ir jų semantiniai tarpusavio ryšiai vadinami ontologija.

OWL turi keleto ankstesnių atvaizdavimo kalbų bruožų, tokių kaip *RDF* (angl. *Resource Description Framework*). Tačiau *OWL* turi daugiau galimybių prasmės ir semantikos išreiškimui nei *XML* (angl. *eXtensible Markup Language*), *RDF* ir *RDFS* (angl. *Resource Description Framework Schema*), todėl *OWL* pirmauja prieš šias kalbas savo galimybėmis pateikti tinkle mašinos apdorojamą turinį [5]. *OWL* taip pat daug sukauptos patirties įtraukė iš *DAML+OIL* ontologijos kalbos kūrimo ir taikymo. (1 pav.)



1 pav. *OWL* genezė

2.2.1. Pagrindiniai *OWL* kalbos dariniai ir sąvokos

OWL kalbos pagrindiniai konceptai yra aprašomi *OWL Lite* poaibyje.

- **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.

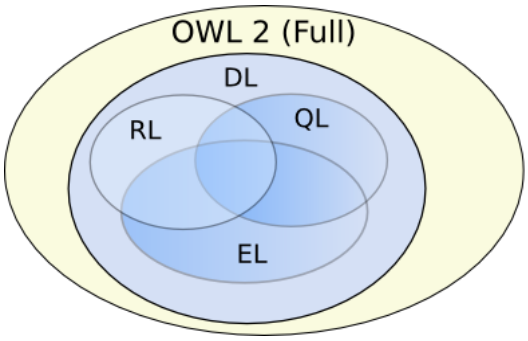
2.2.2. *OWL 2* kalbos skirtumai

Nors *OWL 2* kalba ir papildo savo pirmtakę *OWL* keliomis ne esminėmis, tačiau labai naudingomis savybėmis, bendra struktūra bei pagrindinės kalbos konstrukcijos išlieka nepakitusios. Pagrindinė *OWL* 2 atsiradimo priežastis –modeliavimo galimybių išplėtimas, esamų trūkumų pašalinimas, papildant *OWL* naujais dariniais.

- *OWL* abstrakti sintaksė yra pagrindas *OWL* 2 struktūrai bei funkcinei sintaksei. Naujoji *OWL 2* yra artimesnė *RDF graph* bei formaliai yra lygiavertė *UML*;
- *OWL 2* kaip ir *OWL* apibrėžia tikslų ontologijos struktūrų bei *RDF graph* atitikimą;
- *OWL 2* palaiko naują Manchester sintaksę;
- *OWL 2* turi tris poaibius (EL, QL ir RL) (2 pav.);
- *OWL 2* palaiko visas *OWL* konstrukcijas t.y. ontologijos, aprašytos *OWL*, yra teisingos *OWL 2* kontekste;
- *OWL 2* turi mažiau apribojimų;

  Naujos *OWL 2* apibendrintos savybės:
  - raktai;
  - savybių ryšiai;
  - papildomi duomenų tipai (subsets), duomenų ribojimai;
  - pilni kardinalumo ribojimai;
  - asymmetric, reflexive, disjoint savybės;

- papildomos anotacijų galimybės;



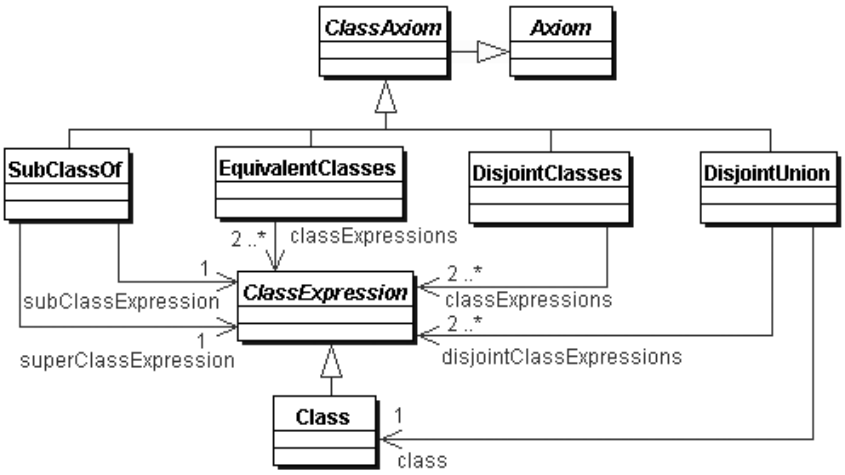
2 pav. *OWL* 2 poaibiai

- *OWL RL* – Turi galimybę modeliuoti validavimo algoritmus naudojant taisyklėmis išplėstas duomenų bazių technologijas, kurios operuoja tiesiogiai su *RDF* „trejetais“. Tinka sistemoms, kuriose naudojamos santykinai nesudėtingos ontologijos dideliems duomenų kiekiams išsaugoti. Duomenys tokiose sistemose dažniausiai yra naudojami kaip *RDF* „trejetukai“ (*subject, predicate, object*)
- *OWL QL* – Apibrėžia jungtines užklausų konstrukcijas panaudojant reliacinių duomenų bazių technologiją. Tinka sistemoms, kuriose naudojamos santykinai nesudėtingos ontologijos dideliems duomenų kiekiams išsaugoti. Duomenys tokiose sistemose dažniausiai yra pasiekiami tiesiogiai naudojant reliacines užklausas (pvz. *SQL*)
- *OWL EL* - Poaibis skirtas didelėms sistemoms, kuriose turi būti užtikrintas veikimas, modeliuoti. Ontologija gali turėti daug klasių ir savybių. Apibrėžti algoritmai visiems standartiniams validavimo - analizės uždaviniams.

2.2.3.*OWL 2* Klasės ir RDB Lentelės

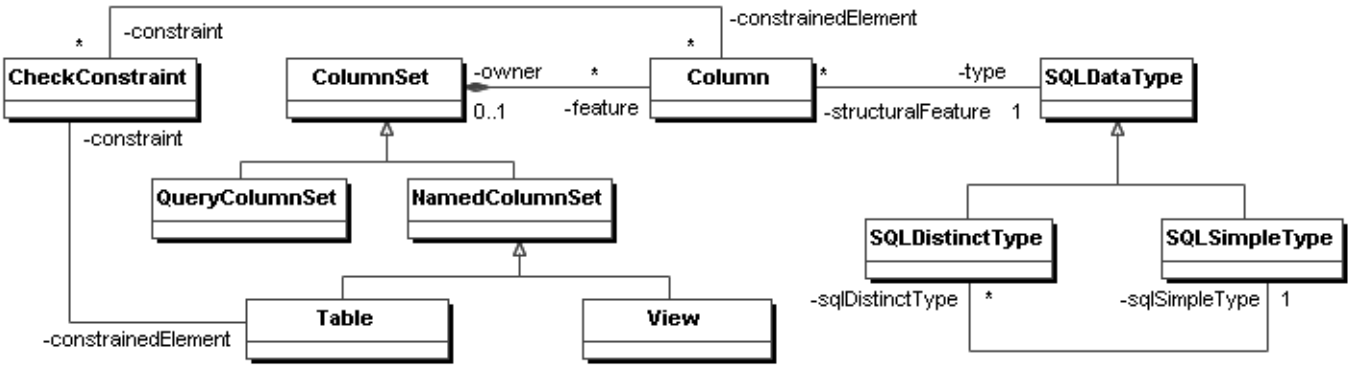
Šioje dalyje apžvelgsime *OWL 2* kalbos bei reliacinių duomenų bazių konceptų pagrindines sąvokas, kurios yra esminės modeliuojant ontologijų išsaugojimą reliacinėse duomenų bazėse.

OWL 2 klasių (angl. *Class*) pagalba, kurios suteikia galimybę taikyti atitinkamą abstrakcijos lygį, galima grupuoti panašias savybes turinčius objektus. Klasė gali būti traktuojama kaip rinkinys individų (objektų). Formaliai nurodant sąlygas individų savybėms, *OWL 2* klasės bei savybių išraiškos (kombinacijos) yra naudojamos klasių išraiškoms (angl. *ClassExpression*), konstruoti. Individai tenkinantys nurodytas sąlygas gali būti laikomi atitinkamos klasės išraiškos atstovais. *OWL 2* apibrėžia aksiomas (angl. *ClassAxiom*), kurių pagalba galima sudaryti ryšius tarp klasių išraiškų. (3 pav.)



3 pav. *OWL* klasių sudarymo schema

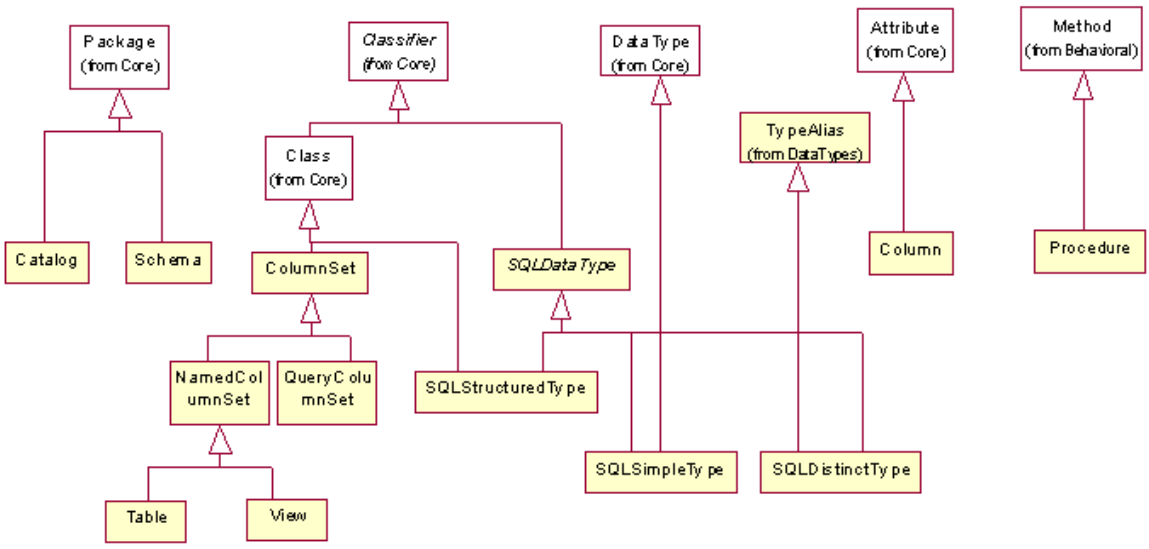
Pagrindiniams reliacinės duomenų bazės konceptams modeliuoti panaudosime dalį CWM metamodelio. (4 pav.) *CWM* [2] – *Common Warehouse Metamodel* (Apibendrintas saugyklos metamodelis). Planuojama išplėsta *CWM 2.x* versija, kuri pavadinta *IMM - Information Management Metamodel* (Informacijos valdymo metamodelis).



4 pav. RDB lentelės, stulpeliai bei duomenų tipai

Stulpelių (angl. *Column*) rinkinys (angl. *ColumnSet*) reliaciniame metamodelyje atitinka bet kokią susijusių duomenų išraišką (angl. *QueryColumnSet*, *NamedColumnSet*). Duomenų išraiška (angl. *NamedColumnSet*) kitaip klasifikuotas stulpelių rinkinys yra schemos objektas, kurį atitinka loginis vaizdas (angl. *View*) arba fizinė duomenų bazės lentelė (angl. *Table*). Lentelių rinkinys laikomas schema. *SQL* (angl. *Structual query language*) užklausos rezultatas - kitas duomenų išraiškos poaibis (angl. *QueryColumnSet*). Stulpeliams yra priskiriamas susietas duomenų tipas (angl. *SQLDataType*), naudojant tipo susiejimą tarp struktūrinės konstrukcijos (angl. *Structural Feature*) ir klasifikatoriaus (angl. *Classifier*) paveldėto iš *OM* (angl. *Object Model*). Paveikslėlyje 4 pavaizduoti du pagrindiniai duomenų tipai: paprastas ir individualus (angl. *simple, distinct*). Paprastieji tipai yra apibrėžiami *SQL* standarto. Kai kurios RDBVS naudoja papildomus tipus. Individualūs tipai yra sukuriami konstruojant paprastus tipus.

Reliacinės duomenų bazės metamodelis yra paveldėtas iš tokių modelių : *Object Model* , *Foundation Model*. Bendra paveldėjimo schema pavaizduota.(5 pav.)



5 pav. RDB paveldėjimo schema

2.3. Esamų sprendimų analizė

2.3.1.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* (angl. *Extended ConjunctiveQueries*) [7]. 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.

2.3.2. 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 [11] 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:

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

Ontologijos pavertimą *ER* modeliu galima vaizduoti kaip jų grafų transformavimą: *GO* (angl. *OWLElement*) → *GER* (angl. *ERElement*) Elementariosios transformacijos yra:

- *OWL* ontologijos elementas yra keičiamas į ER modelio elementą.
- *OWL* klasės yra paverčiamos ER esybėmis.
- *OWL* duomenų tipo savybės yra paverčiamos atributais.
- *OWL* objektų savybės yra paverčiamos sąryšiais.
- *OWL* duomenų ribos yra paverčiamos ribojimais.

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.

2.3.3. Pagrindiniai skirtumai tarp *OWL* ir reliacinių duomenų bazių

Pagrindinis skirtumas tarp *OWL* ontologijų ir duomenų bazių yra tas [9], 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 [2]. 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.

2.3.4.Reliacinių duomenų bazių išplėtimas ontologijų palaikymui

Duomenų bazėse galima saugoti ontologijas keletu būdų [10]:

- 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 sukuriant horizontalias klases ir lenteles kiekvienai savybei. 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* išvedimą, 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.

2.3.5. Universalios duomenų bazės schemos naudojimas

Vienas iš galimų *OWL* ontologijų vaizdavimo į duomenų bazes variantų yra naudojant universalią duomenų bazės schemą [1]. 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.

2.3.6. Ontologijos pavertimas duomenų bazės schema

Šaltinyje [4] parodoma, kaip buvo konkreti ontologija pusiau automatiškai paversta į duomenų bazės schemą:

- Paverčiama koncepcijos hierarchija: kiekviena koncepcija yra paverčiama lentele. Jei super-koncepcija yra parenkama – kuriamos dvi lenteles su tuo pačiu pirminiu raktu.
- Paverčiamos duomenų tipų savybės: kiekviena savybė yra paverčiama lentelės atributu atitinkamoje lentelėje.
- 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.
- Egzemplioriai yra paverčiami duomenų eilutėmis, kurios įdedamos į naują duomenų bazę.

2.3.7. 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 [6] buvo patikrintos ir nuolat optimizuojamos, o tokios savybės kaip plėtojamos automatizuoto atkūrimo ar optimizavimo galimybės suteikia stipresnį pagrindą panaudoti šias technologijas.

Straipsnyje [14] 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.

2.3.8. Transformavimo metodų apžvalga

Apžvelgtų transformavimo algoritmų santrauka pateikta lentelėje:

Lentelė 1 Transformavimo algoritmų santrauka

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

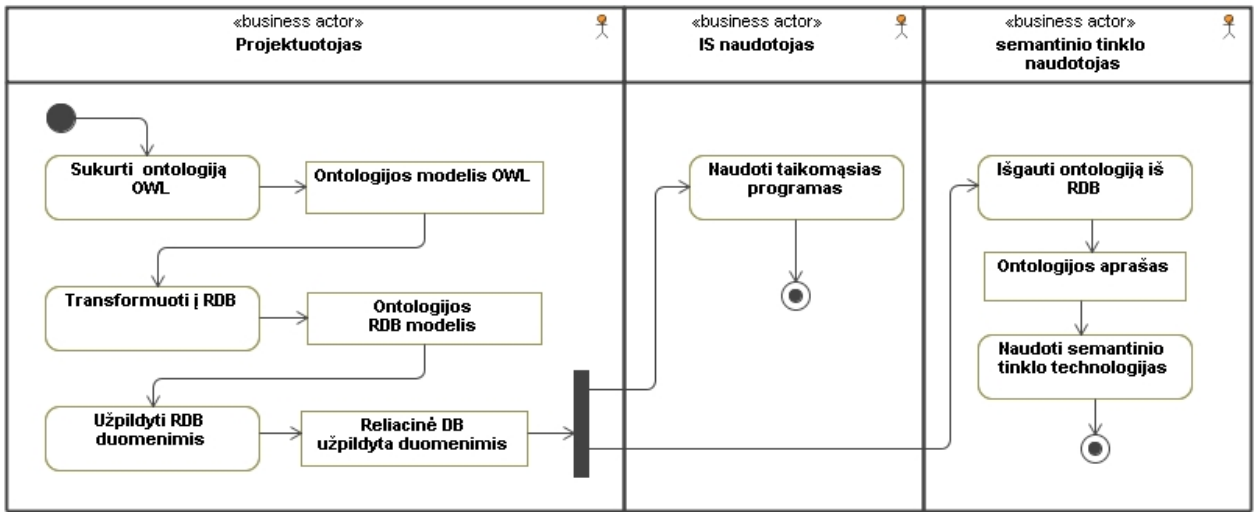
Horizontali klasė	Išlaikomas struktūrinis panašumas - lentelės atitinka klases Panašus į duomenų bazės kūrimo pagal esybes metodą	Prarandama objektų savybių ir apribojimų informacija
Lentelė kiekvienai savybei	Išlaikomas struktūrinis panašumas - lentelės atitinka savybes	Užklausos, kurios naudoja numanomus duomenis, yra labai lėtos
Hibridinis klasių ir savybių transformavimo 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
Hibridinis klasių, savybių ir ribojimų transformavimo metodas (*OWLToRDB*)	Išlaikomas struktūrinis panašumas, neprarandama informacija	Priklauso nuo standarto, kuriuo yra aprašomas modelis iš kurio yra atliekama transformacija (*OWL, OWL2*)

Informacijos sistemų katedroje sukurtame *OWLToRDB* transformavimo sistemos prototipe taikomas transformavimo metodas, kai dalis ontologijos tiesiogiai vaizduojama RDB schema, taip išnaudojant RDB privalumus, o ontologijos elementai, kurie tiesiogiai nepavaizduojami RDB, transformuojami į metaduomenų lenteles. Tačiau prototipas neapima visų ontologijos savybių ir yra skirtas pirmai *OWL* versijai, o šiuo metu atsirado *OWL 2* versija ir tikslinga pereiti prie šios versijos.

Šiame darbe numatoma plėsti *OWLToRDB* prototipą įvertinant *OWL 2* standartą, taikant hibridinį vaizdavimo būdą ir siekiant pavaizduoti didesnę ontologijos konceptų aibę. Pagrindinis metodo kokybės kriterijus – informacijos nepraradimas. Tai reiškia, kad bus siekiama transformuoti ontologiją ir jos egzempliorius taip, kad iš reliacinės duomenų bazės būtų galima atstatyti visą ontologiją ir jos egzempliorius.

2.4. Siekiamos sistemos apibrėžimas

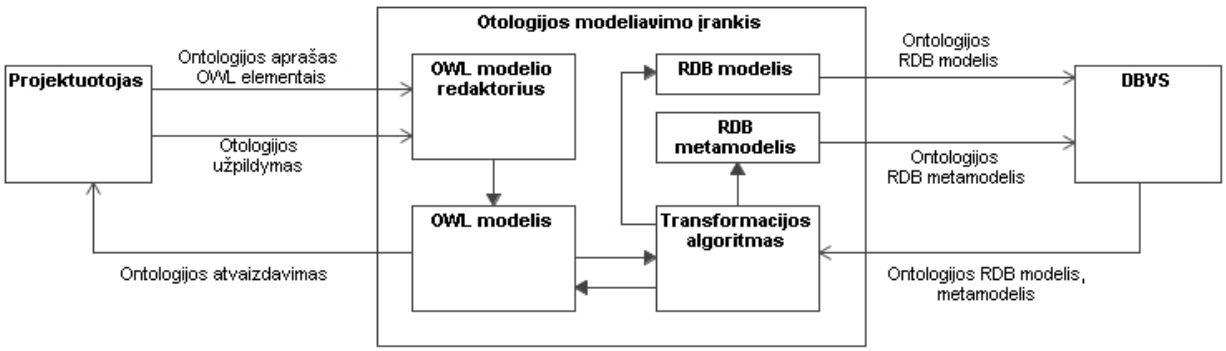
Ontologijų transformavimo proceso modelis gali būti nusakomas veiklos schema (6 pav.)



6 pav. Veiklos procesas

Projektuotojas, naudodamas modeliavimo įrankį, sukuria formalų dalykinės srities modelį (ontologiją). Naudojant transformavimo įrankį, ontologijos aprašas su visais apribojimais atvaizduojamas į *DDL* skriptą, kuris naudojamas ontologijos išsaugojimui reliacinėje duomenų bazėje. Toliau yra išskiriami alternatyvūs ontologijos RDB panaudojimo scenarijai. Informacinė sistema kreipiasi į duomenų bazę ir naudodamasi ten esančiais duomenimis bei metaduomenimis, pateikia rezultatus vartotojui. Kitas scenarijus yra ontologijos panaudojimas semantiniame tinkle, kur technologijos pritaikytos dirbti su ontologiniais aprašais, todėl turi būti atlikta atvirkštinė transformacija t.y sukurtas *OWL* modelis.

Siekiamos sistemos kontekstą sudaro žinių bazės *projektuotojas*, ontologijos *modeliavimo įrankis* bei *duomenų bazių valdymo sistema* (7 pav.)



7 pav. Transformavimo sistemos kontekstas

Įėjimo duomenys

- (*projektuotojas →OWL modelio redaktorius*):
  ontologijos aprašas *OWL* elementais - struktūros sudarymas;
  ontologijos užpildymas - egzempliorių įvedimas.
- (*dbvs →transformavimo algoritmas*):

Ontologijos RDB modelis - ontologijos šaltinis pasirenkamas DBVS (kiti galimi šaltiniai: failas, žiniatinklis, „TONES“ saugykla);

Ontologijos metamodelis - ontologijos šaltinis pasirenkamas DBVS.

Išėjimo duomenys

- (*OWL modelis →projektuotojas*):
  Ontologijos atvaizdavimas – struktūros *OWL* aprašo perteikimas grafine forma.
- (*rdb modelis→ dbvs)*
  Ontologijos RDB modelis - ontologijos, skirtos išsaugojimui reliacinėje duomenų bazėje, transformuotas modelis.
- (*rdb metamodelis→ dbvs*)
  Ontologijos RDB metamodelis - ontologijos, skirtos išsaugojimui reliacinėje duomenų bazėje, pritaikytas metamodelis.

Sudarant sistemos kontekstą apibrėžti abstraktūs vidiniai elementai. Modeliavimo įrankio komponentai: *OWL modelio redaktorius*, *OWL modelis* – integracinė terpė, *transformavimo algoritmas, rdb modelis, rdb metamodelis* – kuriami komponentai.

Siekiamas galutinis rezultatas yra dalykinės srities duomenų modelis reliacinėje duomenų bazėje, kuriam taikomi atitinkami kokybės kriterijai:

- Kategorijų vientisumas;
- Nuorodų vientisumas;
- Funkcinės priklausomybės;
- Loginė schema yra bent 3-ios normalinės formos.

Sprendimo/algoritmo kriterijai:

- Transformavimo proceso metu negali būti prarandami duomenys, duomenų tipai bei pati ontologijos struktūra;
- Iš ontologijos sugeneruota reliacinės duomenų bazės struktūra turi būti lanksti informacijos išsaugojimui bei užtikrinti efektyvų jos panaudojimą informacinėse sistemose;
- Transformavimo procesas turi būti pilnai automatizuotas, realizuojamas bei patikrintas įrodymais;

Transformavimo sistemos realizacijai bei kūrimo procesui keliamos grėsmės:

- Priklausomybė nuo pasirinktos integravimo terpės;
- Suderinamumas su skirtingomis DBVS;

- Operavimo laikas;
- Sudėtinga *OWL* struktūrinių elementų analizė;
- Didelė realizacijos apimtis.

Priklausomybė nuo pasirinktos integravimo terpės

Kuriama algoritmo realizacija – komponentas priklauso nuo integravimui pasirikto įrankio. Nepakankamai dokumentuoti įrankio realizacijos komponentai kelia grėsmę ne tik transformavimo sistemos integravimui bet ir projekto įgyvendinimui. Renkantis ontologijų modeliavimo įrankį, į kurį bus integruojamas algoritmas ši grėsmė privalo būti įvertinta.

Suderinamumas su skirtingomis DBVS

Sprendimo įgyvendinimui būtinas komponentas yra duomenų bazių valdymo sistema. Siekiamas rezultatas – nepriklausomas nuo pasirinktos DBVS transformavimo sistemos veikimas. Projektuojant realizacijos komponentus turės būti suprojektuota nepriklausoma DBVS sąsaja arba pasirinktas sąsajai keliamus reikalavimus atitinkantis trečios šalies komponentas.

Operavimo laikas

Ontologijos *OWL* modelį sudaro aibė elementų bei ryšių. Pati ontologija gali būti užpildyta taip pat aibe egzempliorių. Pertekliniai arba per daug sudėtingi elementų analizės bei modelio sudarymo algoritmai kelia grėsmę įrankio operavimo laikui. Ontologijos išsaugojimo duomenų bazėje operacijos vykdymo laikas gali būti santykinai ilgas lyginant su alternatyviais ontologijos išsaugojimo būdais. Todėl labai svarbu sudarant algoritmus vengti ciklinių, pasikartojančių žingsnių bei daugkartinės *OWL* elementų analizės.

Sudėtinga *OWL* struktūrinių elementų analizė

Transformavimo sistema *OWL* elementų analizei naudos integracinės terpės komponentą, kuris skirtas *OWL* ontologijos modelio sudarymui bei saugojimui kompiuterio atmintyje. Todėl algoritmo sudėtingumas bei sprendimo veikimo laikas tiesiogiai priklauso nuo šio komponento realizacijos. Ryšių klasių struktūroje trūkumas bei nepakankamas interfeisų apibrėžimas lemia sudėtingesnę bei mažiau lankstesnę transformavimo sistemos realizaciją. Įvertinus šią grėsmę, nuspręsta algoritmą bei kuriamus modelius suderinti su esamu *OWL* modelio operaciniu komponentu.

Didelė realizacijos apimtis

Ontologijos *OWL* modelio išsaugojimui RDB turės būti sukurtos net tik būtiniausių elementų transformacijos. Siekiant pritaikyti atvirkštinį metodą – *OWL* modelio sukūrimas iš reliacinėje duomenų bazėje išsaugoto modelio, turės būti sukurtas metaduomenų modelis. Visi *OWL* elementai bei jų ryšiai išsaugomi šiame modelyje. Siekiant įgyvendinti visą numatytą transformavimo sistemos funkcionalumą turės būti taikomi realizacijos klasių generavimo iš modelių sprendimai.

2.5. Architektūros ir galimų įgyvendinimo priemonių analizė

Ontologijų projektavimo įrankių analizė buvo atlikta,remiantis siekiamo sprendimo integracinės terpės poreikiu. Nors rinkoje šiuo metu yra nemažai ontologijų aprašymo įrankių, išsami analizė atlikta tarp dviejų lyderių: „*Altova Semantic Works 2011*“ ir „*Protégé 4.1*“.

„Altova Semantic Works 2011“ yra vizualus *RDF*/*OWL* redaktorius. Suteikiamos galimybės:

- Vaizdinis kūrimas ir redagavimas *RDF, RDF Schema (RDFS), OWL Lite, OWL DL ir OWL Full* dokumentų naudojant intuityvią, vaizdinę vartotojo sąsają ir „*drag-and-drop*“ funkcionalumą.
- Sintaksės tikrinimas užtikrinant atitikimą su *RDF/XML* specifikacijomis.
- *RDF/XML* arba *N-triples* formatų automatinis generavimas ir redagavimas naudojantis grafiniu *RDF/OWL* projektu.
- Grafinių *RDF* ir *OWL* atvaizdų spausdinimas tam, kad sukurti semantinio tinklo elemento dokumentaciją.

Lentelė 2 „Altova Semantic Works 2011” charakteristika

Semantinio tinklo kūrimo palaikymas	▫ grafinis *RDF* dokumentų redaktorius ▫ grafinis *RDF* schemų žodynų redaktorius ▫ grafinis *OWL* ontologijų redaktorius ▫ semantinis ontologijų tikrinimas
RDF/RDFS redagavimas (***RDF/RDFS Editing***)	▫ Pasirinkimas reikiamo *RDF* lygmens (*RDF*, *RDFS*); ▫ Sintaksės tikrinimas; ▫ Grupavimo pagal klases, savybes, egzempliorius peržiūra; ▫ Galimybė paslėpti tuščius mazgus ▫ Dokumentų importavimas ▫ Importuotų dokumentų atidarymas iš naujo ▫ Galimybė keisti peržiūros nustatymus: braižymo kryptį, atstumus ir t.t.; ▫ *RDF*, *RDF* schemų, *OWL* failų atidarymas ir saugojimas *N-triple (nt)* formatu; ▫ *RDF*/XML formato eksportavimas į *N-triples* formatą; ▫ *N-triples* eksportavimas į *RDF*/*XML* formatą; ▫ Patogus spausdinimas *RDF*/*XML* failų per *RDF*/*XML* eksportavimą; ▫ Sutrumpinimai skirti *URI* (angl. *Uniform Resource Identifiers*) nuorodoms gali būti patalpinti į specialią lentelę; ▫ Galimybė įvesti *URI* tiek pilna, tiek sutrumpinta forma;

OWL redagavimas (***OWL Editing***)	▫ Pasirinkimas reikiamo *OWL* lygmens (*OWL Lite, OWL DL, OWL Full*); ▫ Sintaksės tikrinimas ▫ Grupavimo peržiūra pagal klases, savybes, egzempliorius, ontologijas; ▫ Konteksto peržiūra, kuri pateikia visas savybes susijusias su pažymėtomis klasėmis ar jų egzemplioriais; ▫ Semantinis *OWL Lite* ir *OWL DL* tikrinimas (logiškas tikrinimas) ▫ Galimybė paslėpti tuščius mazgus ▫ Dokumentų importavimas ▫ Importuotų dokumentų atidarymas iš naujo ▫ Galimybė keisti peržiūros nustatymus: braižymo kryptį, atstumus ir t.t.; ▫ *RDF, R*DF schemų, *OWL* failų atidarymas ir saugojimas *N-triple (nt)* formatu; ▫ *RDF/XML* formato eksportavimas į *N-triples* formatą; ▫ N-triples eksportavimas į *RDF/XML* formatą; ▫ Patogus spausdinimas *RDF/XML* failų per *RDF/XML* eksportavimą; ▫ Sutrumpinimai skirti *URI* (angl. *Uniform Resource Identifiers*) nuorodoms gali būti patalpinti į specialią lentelę; ▫ Galimybė įvesti *URI* tiek pilna, tiek sutrumpinta forma;
Vartotojo sąsaja (***User Interface***)	▫ Nauja ir pagerinta sąsajos išvaizda ▫ Praplėstas šriftų savybių nustatymas, skirtas visiems modeliams ▫ Konfigūruojamas spausdinimas ir spausdinimo peržiūrėjimas ▫ Paieškos ir pakeitimo savybių lankstumas ▫ Įrankių juosta ir nuorodos pritaikytos lanksčiam vartotojų naudojimui ▫ Neribotas atšaukimo/pakartojimo funkcijų naudojimas ▫ Pagerintas kilnojamų/fiksuojamų meniu juostų naudojimas ▫ Daugiadokumentinė sąsaja ▫ Lengvai naudojama, efektyvi kontekstinė vartotojo pagalba;
Palaikomos operacinės sistemos (***Platforms***)	▫ *Microsoft Windows application* (*NT 4.0, 2000, XP, Server 2003*)
Tarptautinis palaikymas (***International support***)	▫ *Unicode (UTF-7, UTF-8, UTF-16, ISO-10646, UCS-2, UCS-4)* ▫ Visi pagrindiniai simbolių rinkinių atkodavimai (*ASCII, ISO-8859, CJKV,* t.t.) ▫ Bendradarbiavimas tarp skirtingų simbolių rinkinių ir *Unicode*

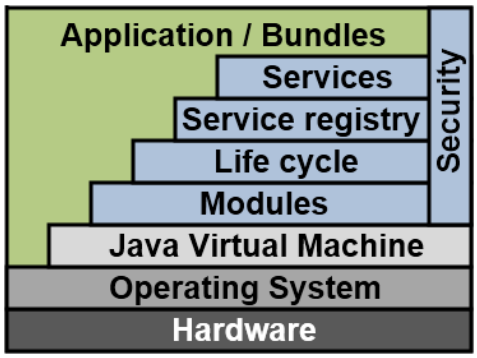
Kitas ontologijų kūrimo įrankis „*Protégé 4.1*“ yra rinkos lyderis dėl savo paplitimo, atviro kodo licencijos ir praplečiamumo.

„Protégé 4.1“ platforma palaiko du pagrindinius ontologijų kūrimo kelius: naudojant *Protege*-karkasus (angl. *Protege-Frames*) ir *Protege-OWL* redaktorius. „*Protégé*“ ontologijos gali būti eksportuojamos į daugybę formatų, įskaitant *RDF(S), OWL* ir *XML Schema*. „*Protégé 4.1*“ sukurta naudojantis Java programavimo kalba, pats produktas yra praplečiamas, turi „*plug and play*“ aplinką.

Lentelė 3 „Altova Semantic Works 2011“ ir *Protégé* palyginimas

	Altova Semantic Works	Protege
Ontologijų kūrimas	+	+
OWL2 palaikymas	-	+
Patogi vartotojo sąsaja	+	+
Platus OS palaikymas	-	+
Nemokamas	-	+
Praplečiamumas	-	+

Abu ontologijų kūrimo įrankiai yra pakankamai išvystyti ir palaiko ontologijų aprašymo kalbą *OWL*, tačiau „*Altova Semantic Works*“ yra mokamas komercinis produktas, be to, jo neįmanoma praplėsti, t.y. labiau pritaikyti specifinei tyrimo sričiai. Tuo tarpu „*Protege*“ yra ne tik atviro kodo įrankis, bet ir suteikia plėtinių mechanizmą (angl. *OSGI Framework*) (8 pav.)



8 pav. *OSGi* karkaso architektūra

OSGi (angl. *Open Services Gateway initiative framework*) – modulinė sistema bei paslaugų platforma skirta Java programavimo kalbai. Sistema realizuoja dinaminį komponentinį modelį. Programiniai komponentai gali būti įdiegiami ar atnaujinami nuotoliniu būdu nepertraukiant sistemos ar kitų komponentų darbo. Sistemą sudarantys sluoksniai:

- Aplikacijų – komponentų rinkiniai;
- Paslaugų – jungiamoji komponentų dalis;
- Paslaugų registras – paslaugų valdymas;
- Moduliai – komponentų priklausomybių deklaravimas, kodo publikavimas;
- Saugumas – komponentų funkcionalumo ribojimas;
- Vykdymo aplinka – komponentų naudojami metodai.

Remiantis kriterijais, pateiktais(Lentelė 3 „Altova Semantic Works 2011“ ir *Protégé* palyginimas), nuspręsta ontologijų kūrimui naudoti „*Protege*“ kūrimo įrankį.

2.6. Analizės išvados

Darbe buvo apžvelgtos problemos, susijusios su dalykinės srities ontologijų naudojimu, detaliau nagrinėjant produktų konfigūravimo žinių bazės modeliavimą. Analizės metu padarytos tokios išvados:

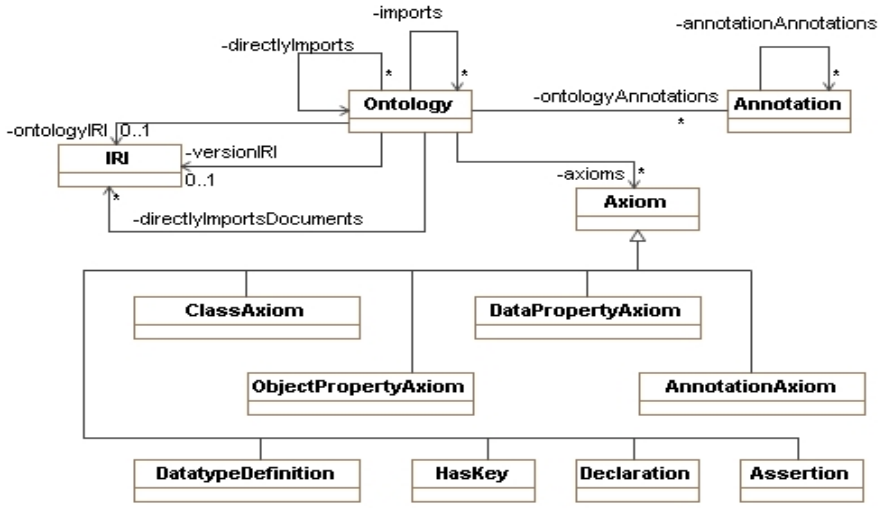
1. Išnagrinėjus literatūros šaltinius, galima daryti išvadą, kad šiuo metu nėra standartų, kaip ontologijos *OWL* 2 aprašus transformuoti į reliacines duomenų bazes. Tačiau yra pavyzdžių bei transformavimo sistemos prototipų, kurių pagrindu galima sukurti savo transformavimo sistemą bei patobulinti naudojamus algoritmus.
2. Išanalizuoti ontologijų atvaizdavimo į reliacinę duomenų bazę metodai ir algoritmai parodė, kad tinkamiausias yra klasių, savybių ir ribojimų transformavimo algoritmas, kurio principai bus tobulinami šiame darbe.
3. Ontologijų transformavimo į reliacinę duomenų bazę proceso analizė parodė, kad jam vykdyti reikalingi dalykinės srities ontologinio modeliavimo ir formalaus aprašo transformavimo į duomenų bazę įrankiai.
4. Atlikus ontologijų modeliavimo įrankių analizę, tolesniam tyrimui pasirinkta „Protege“ ontologijų valdymo sistema, kadangi tai plačiausiai naudojamas, nemokamas, nuolat tobulinamas įrankis.
5. Kadangi klasių, savybių ir ribojimų transformavimo algoritmas buvo realizuotas su *OWL* ankstesne versija ir neapima visų *OWL* 2 savybių, suformuluotas tolesnio tyrimo uždavinys sukurti patobulintus ontologinio aprašo *OWL* 2 transformavimo į reliacinį pavidalą algoritmus ir realizuoti transformavimo įrankį.

3. *OWL2ToRDB* metodas bei algoritmai

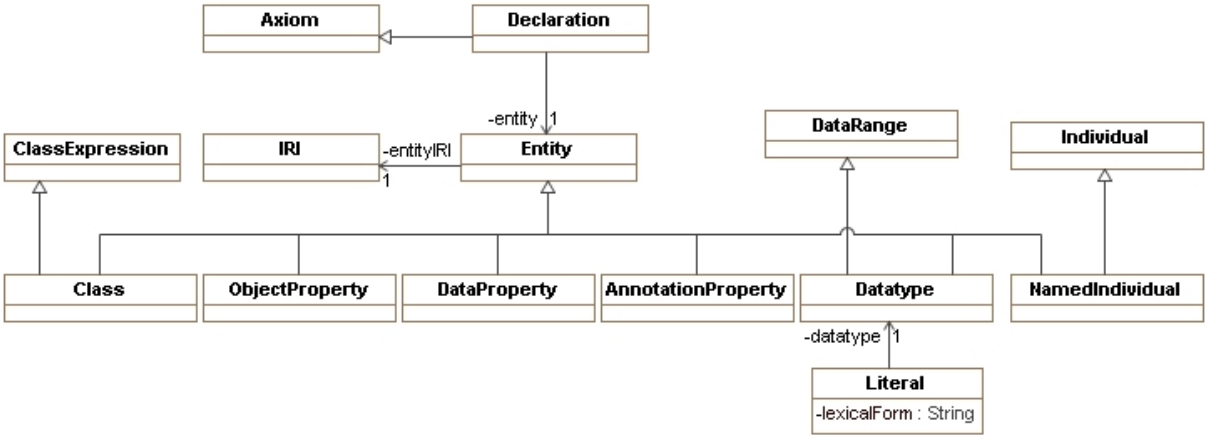
3.1. *OWL2* ir RDB metamodeliai

Transformacijos remiasi vykdomos tarp *OWL* 2 ir RDB modelių, kurių abstrakčią sintaksę aprašo *OWL* 2 ir RDB metamodeliai (jų dalys).

Abstraktūs modeliai bus naudojami įrankio elgsenos, struktūros bei realizacijos projektavime.Transformavimo algoritmo aprašymui naudosime metamodelių fragmentus. Aukščiausio lygio *OWL 2* metamodelį sudaro elementai : *ontologija*, *anotacijos*, *aksiomos.*(9 pav.) *Aksioma* yra pagrindinis elementas apibrėžiant *OWL 2* semantines konstrukcijas. *OWL 2* struktūriniai elementai - *Esybės (klasės, objektinės savybės, anotacijų savybės, duomenų savybės, individai, duomenų tipai),* sudaro ontologijos žodyną (10 pav.)



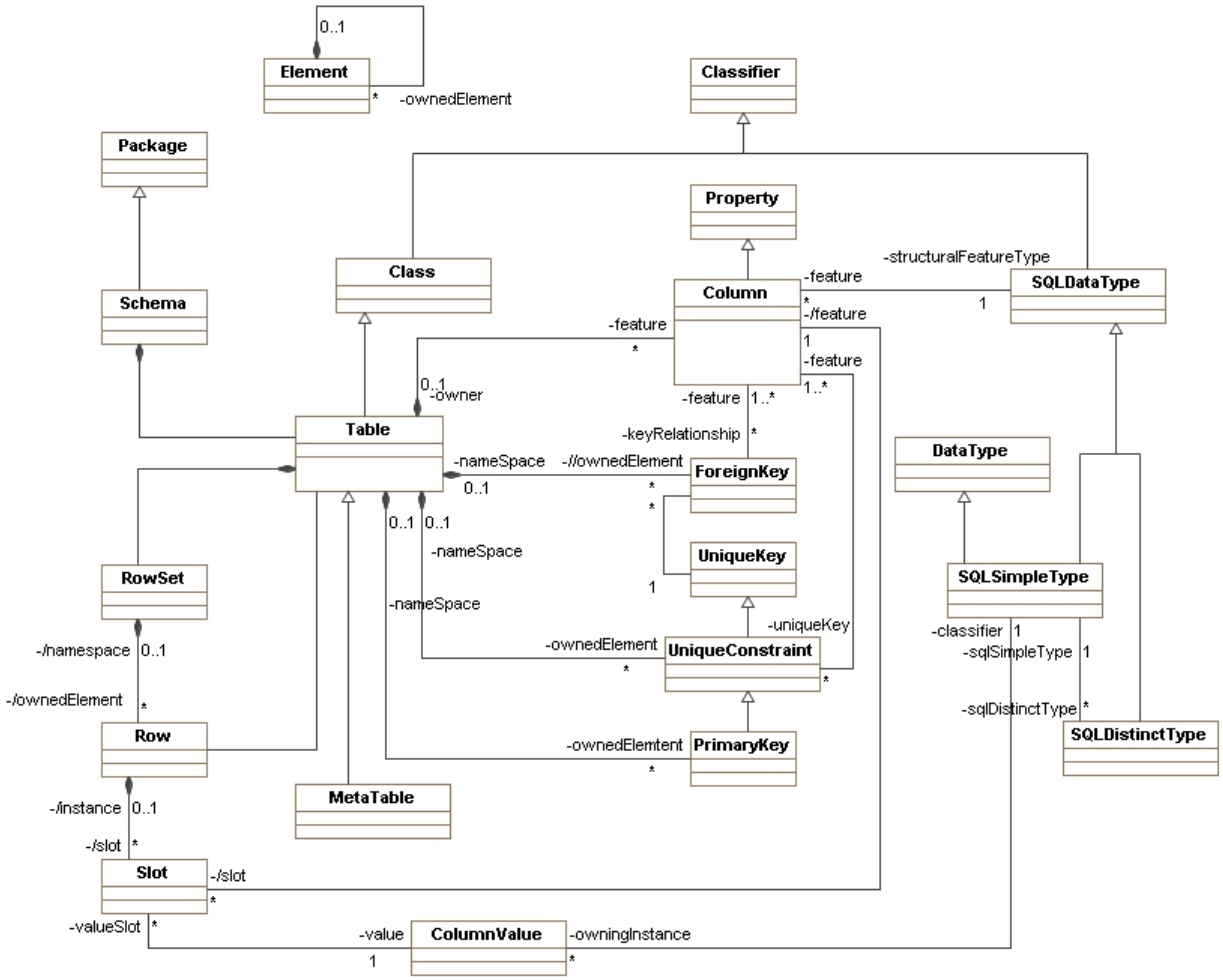
9 pav. *OWL 2* ontologijos metamodelis



10 pav. *OWL 2* esybių metamodelis

Pagrindiniams reliacinės duomenų bazės konceptams modeliuoti panaudosime dalį CWM metamodelio (11 pav.) CWM [2] – *Common Warehouse Metamodel* (Apibendrintas saugyklos metamodelis). Metamodelį papildysime metalentelėmis (angl. *Metatable*), kurios bus panaudojamos ontologijos struktūros (bendro *OWL* 2 aprašo) išsaugojimui. Stulpeliams yra

priskiriamas susietas duomenų tipas (angl. *SQLDataType*), naudojant tipo susiejimą tarp struktūrinės konstrukcijos (angl. *structual Feature*) ir klasifikatoriaus (angl. *Classifier*) paveldėto iš *OM* (angl. *Object Model)*. Modelio schemoje pavaizduoti du pagrindiniai duomenų tipai: paprastas ir individualus (angl. *simple, distinct*). Paprastieji tipai yra apibrėžiami *SQL* standarto. Kai kurios RDBVS naudoja papildomus tipus. Individualieji tipai sudaromi konstruojant paprastus tipus.



11 pav. Reliacinės duomenų bazės metamodelis

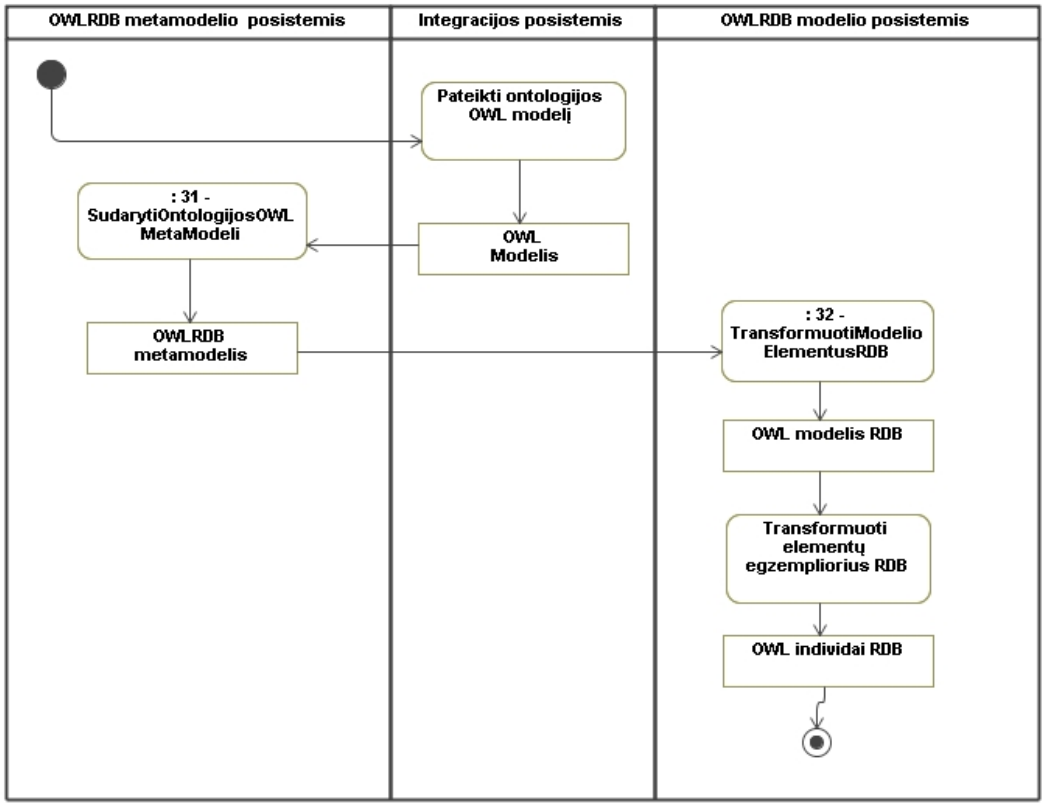
3.2. Transformavimo procesas

Veiklos procesų modeliais aprašomi *OWLRDB* modelio bei *OWLRDB* metamodelio posistemio komponentai. Integracinio posistemio komponentas yra realizuotas ontologijų modeliavimo įrankyje.

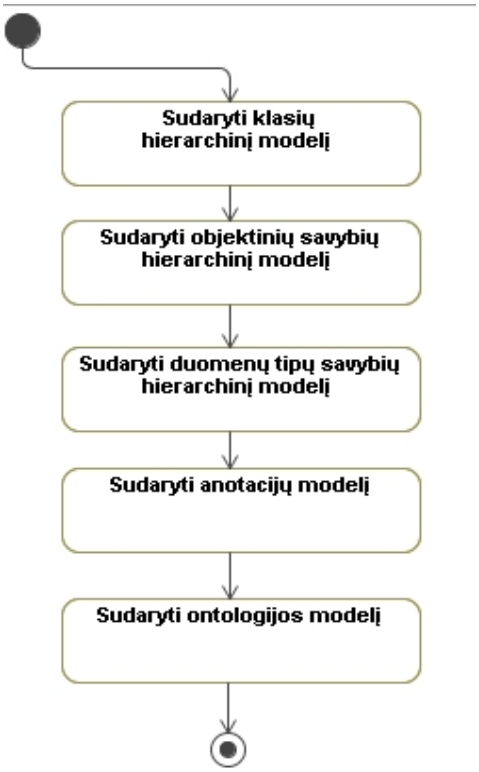
Bendra transformavimo sistemos veiklos proceso schema pateikta (12 pav.). Analizuojant *OWL* modelio elementus bei jų ryšius sukuriamas metaduomenų modelis, kuris toliau naudojamas *OWL* elementų transformavimui. Galutinis proceso rezultatas – reliacinėje duomenų bazėje išsaugotas ontologijos *OWL* metamodelis, transformuoti elementai bei egzemplioriai.

Toliau detalizuojami bei aprašomi procesų modeliai:

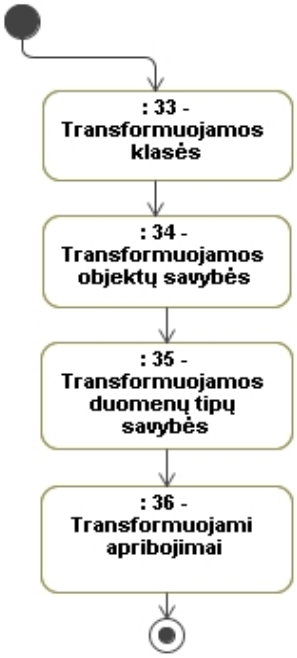
- "Įrašyti metaduomenis" (13 pav.);
- "Transformuoti modelio elementus RDB" (14 pav.);



12 pav. Transformavimo sistemos veiklos proceso schema



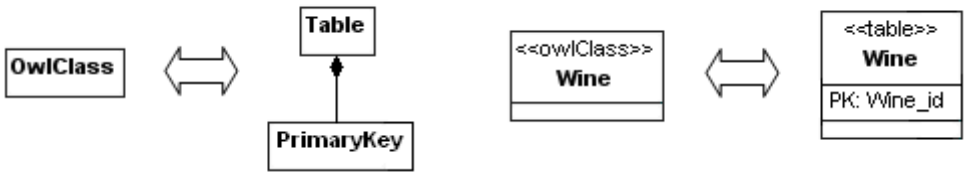
13 pav. Metaduomenų modelio sudarymo proceso schema



14 pav. *OWL 2* modelio elementų transformavimo proceso schema

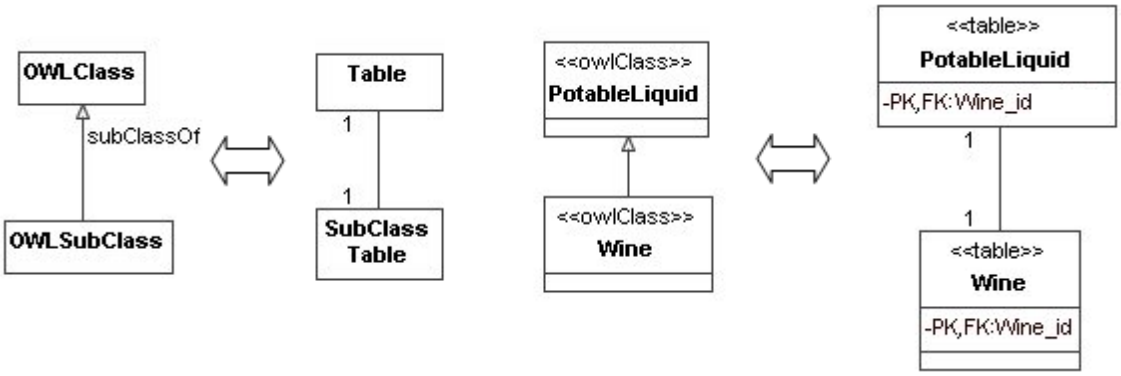
3.3. Ontologijos klasių transformavimas į RDB

Ontologijos klasės yra transformuojamos į lenteles duomenų bazėje (15 pav.). 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.



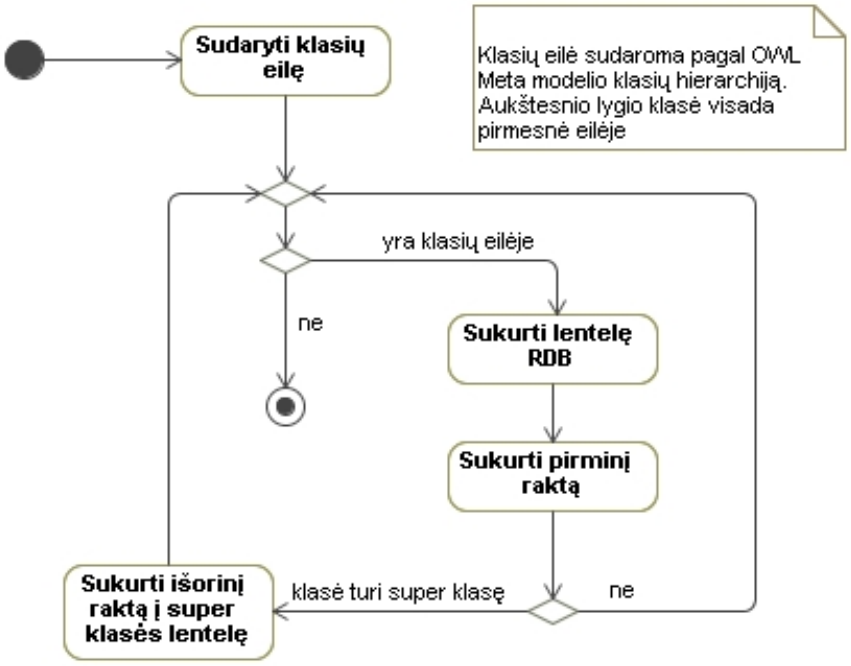
15 pav. Klasių transformavimas

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.



16 pav. Poklasių transformavimas

Algoritmas, transformuojantis ontologijos klases į reliacinę duomenų bazę (17 pav.), naudoja metaduomenų modelio klasių hierarchiją. Klasių eilė sudaryta pagal paieškos į gylį principą t.y einama nuo aukščiausios klasės prie žemiausio lygio klasės kol klasė turi poklasių (rekursijos principas). Tokiu principu sudaryta eilė užtikrina, jog aukštesnio lygio klasė visada bus pirmesnė. Kiekvienai ontologijos klasei duomenų bazėje sukuriama lentelė, klasių hierarchiniai ryšiai užtikrinami 1:1 ryšiais tarp klasės ir jos poklasės.



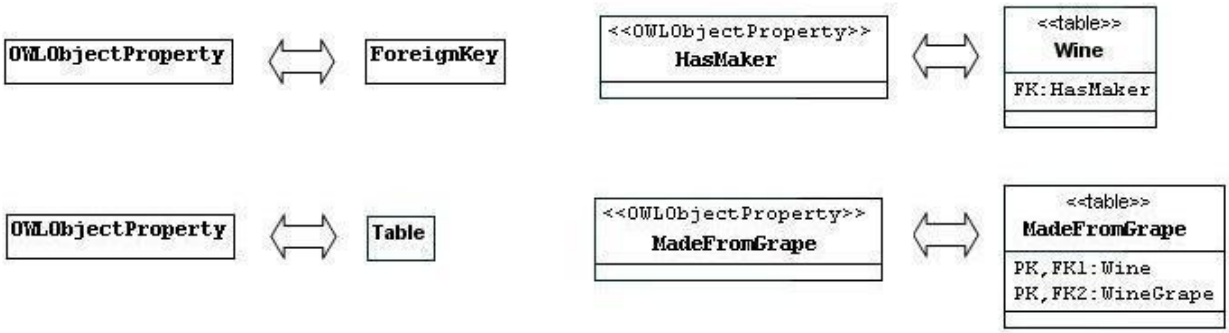
17 pav. Klasių transformavimo proceso schema

3.4. Ontologijos objektinių savybių transformavimas į RDB

Atliekant ontologijos transformavimą į reliacinę duomenų bazę, po ontologijos klasių transformavimo į reliacinės duomenų bazės lenteles vykdomas objektinių klasių savybių transformavimas į ryšius tarp klasių lentelių.

Objektinė savybė yra ryšys tarp dviejų klasių egzempliorių. Sukuriant savybę, jos apribojimą galima apibrėžti keliais būdais: galima nurodyti domeną (angl. *domain*) ir sritį (angl. *range*), galima nurodyti, kad savybė yra tam tikros kitos savybės specializacija.

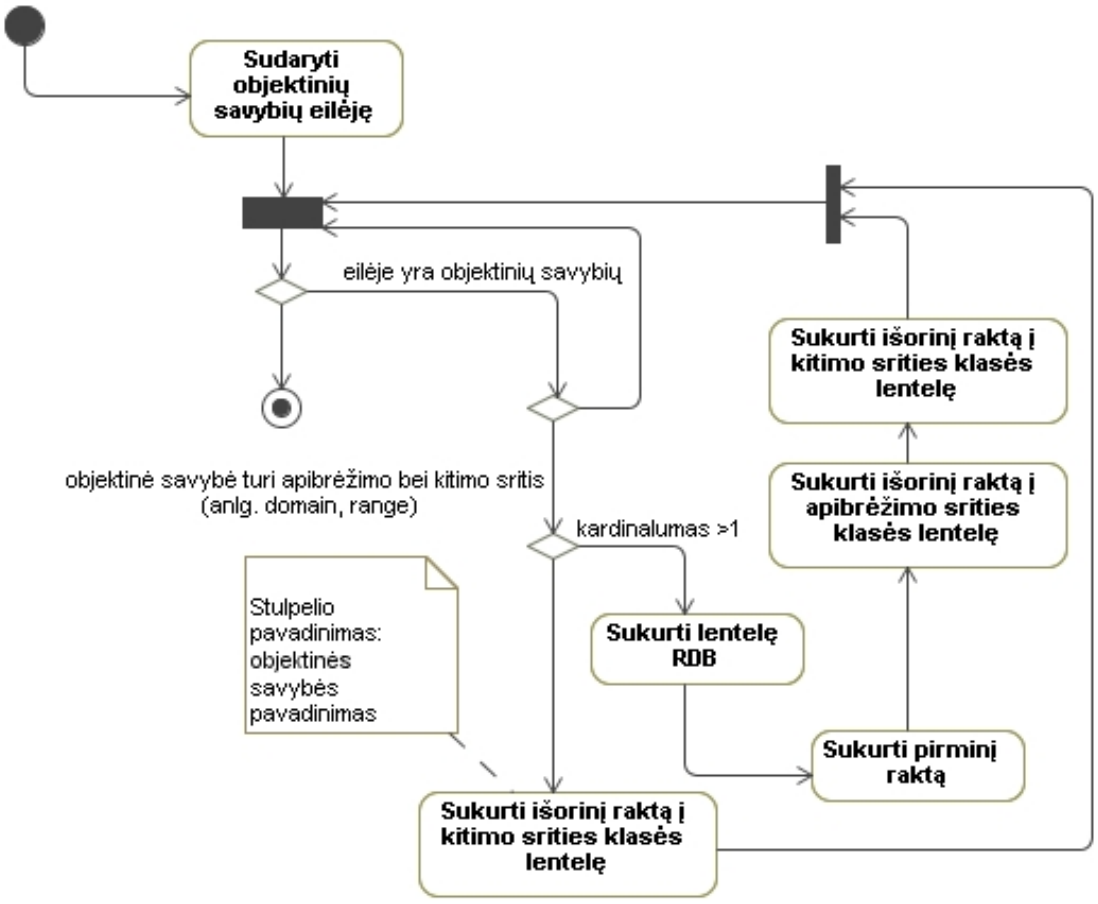
OWL objektų savybės yra paverčiamos į ryšius tarp klasių lentelių. Priklausomai nuo jų kardinalumo, naudojamos tarpinės lentelės su išoriniais raktais, kai kardinalumas daug su daug, arba kuriami stulpeliai, kai kardinalumas yra vienas su daug (18 pav.)



18 pav. Objektinių savybių transformavimas

Algoritmas, transformuojantis ontologijos savybes į reliacinės duomenų bazės ryšius tarp lentelių (19 pav.), taip pat naudoja metamodelio objektinių savybių modelį. Pirmiausiai nagrinėjamos savybės, kurios neturi hierarchiškai aukštesnių savybių, toliau jas paveldinčios

savybės ir t.t. Priklausomai nuo klasės lokalaus kardinalumo apribojimo savybei, sukuriamas vienas-su-daug arba daug-su-daug ryšys tarp klasių lentelių. Daug-su-daug ryšio atveju sukuriama tarpinė lentelė, taip daug-su-daug ryšį pakeičiant dviem vienas-su-daug ryšiais.



19 pav. Objektinių savybių transformavimo proceso schema

3.5. Ontologijos duomenų tipų savybių transformavimas į RDB

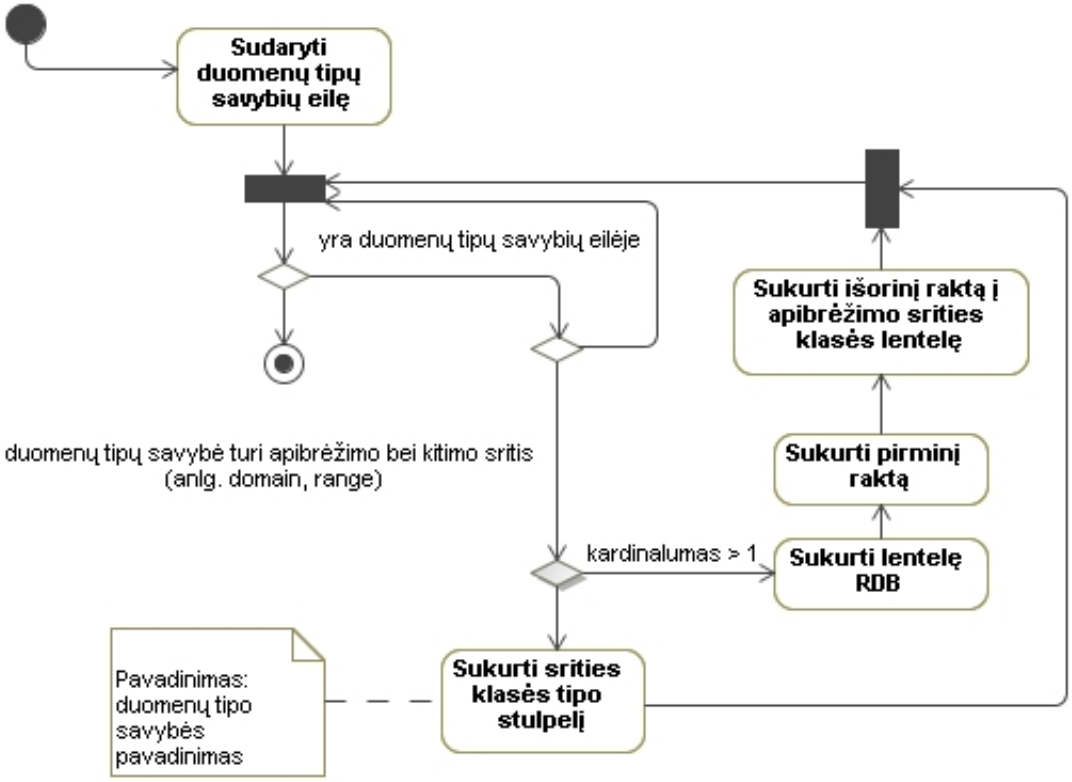
Atliekant ontologijos transformavimą į reliacinę duomenų bazę, po ontologijos objektinių savybių transformavimo į reliacinės duomenų bazės ryšius tarp lentelių, vykdomas duomenų tipų klasių savybių transformavimas į klasių lentelių duomenų stulpelius. Duomenų tipų savybės ontologijoje susieja klasių egzempliorius su duomenų tipais.

OWL duomenų tipų savybės yra paverčiamos naujais stulpeliai lentelėse, kurių klasėms jos taikomos, ir užpildomos atitinkama kiekvieno egzemplioriaus turima reikšme (20 pav.).

20 pav. Duomenų tipų savybių transformavimas

Algoritmas, transformuojantis ontologijos duomenų tipų savybes į reliacinės duomenų bazės lentelių stulpelius (21 pav.), analizuoja visas duomenų tipų savybes. Pagal domain reikšmę

surandama duomenų bazės lentelė, lentelėje sukuriamas stulpelis su savybės pavadinimu. Lentelės stulpelio laukams nustatomas duomenų tipas pagal savybės kitimo srities (angl. *range*) reikšmę. Patikrinus visas duomenų tipų savybes, algoritmo darbas baigiamas.



21 pav. Duomenų tipų savybių transformavimo proceso schema

3.6. *OWL2ToRDB* metodo keliami reikalavimai ontologijoms

Siekiant neprarasti ontologinės informacijos, transformuojama į reliacinę duomenų bazę ontologija turėtų atitikti tam tikrus kokybės reikalavimus. Pirma, ontologija turėtų būti suprojektuota remiantis ontologijų projektavimo taisyklėmis. Pagrindinis reliacinių duomenų bazių kokybės reikalavimas yra jų normalizavimas.Informacinių modelių normalizavimo metodas yra plačiai priimtas ir taikomas praktikoje, kuriat duomenų bazes ir informacines sistemas.Analogiško ontologijų normalizavimo metodo tikslas:

- domeno teisingumas - tai reiškia, kad išvestinės modelio klasifikacijos, gaunamos įvairių ontologijos validatorių pagalba, atitinka ontologijos kūrėjų lūkesčius;
- naudingumas – ontologijos taikymas sukuria pridėtinę vertę informacijos sistemai;
- modalumas– skirtingos ontologijos dalys gali būti kuriamos atskirai ir vėliau sujungiamos. Kuriant tokiu būdu yra galimas pakartotinis panaudojimas, palaikymas bei vystymas.

Transformuojama ontologija taip pat turi atitikti tam tikrus integralumo apribojimus, kurie yra taip pat taikomi reliacinėms duomenų bazėms. Kitu atveju reliacinės duomenų bazės struktūra, kurioje talpinama ontologijoje saugoma informacija, tampa labai sudėtinga arba informacija gali būti prarasta dėl to, kad jos neįmanoma išsaugoti sukurtoje duomenų bazės struktūroje. Ontologijų aksiomos yra naudojamos išvedimui, bet ne integralumo užtikrinimui. Integralumas nepalaikomas ontologijų kūrimo įrankiuose.

Ontologijų transformavimo į reliacines duomenų bazes reikalavimai nėra įvertinami daugumos esamų transformavimo metodų. Dan-Tong et al. [3] pasiūlė integralumo apribojimus ontologijoms aprašyti *OWL* aksiomų pagalba bei pritaikyti ontologijas transformavimui naudojant pačius ontologijų kūrimo įrankius. Nors *OWL2ToRDB* metodas skiriasi nuo Dan-Tong et al., jam taikomi analogiški ontologijų normalizavimo reikalavimai bei integralumo apribojimai remiantis *OWL* 2 specifikacija [4]:

- kiekvienas ontologijos elementas turi turėti tik vieną tėvinį elementą;
- kiekviena objektinė bei duomenų tipų savybė privalo turėti po vieną apibrėžimo bei kitimo sritį;
- elementų pavadinimuose turi būti vengiama naudoti specializuotus terminus (angl. *like, union, table ir kit.*), kurie gali būti laikomi tarnybiniais skirtingose duomenų bazių valdymo sistemose;
- ontologija negali būti sudaryta iš importuotų (svetimų) elementų;
- tiesioginė bei atvirkštinė objektinės savybės turi būti neprieštaringos.

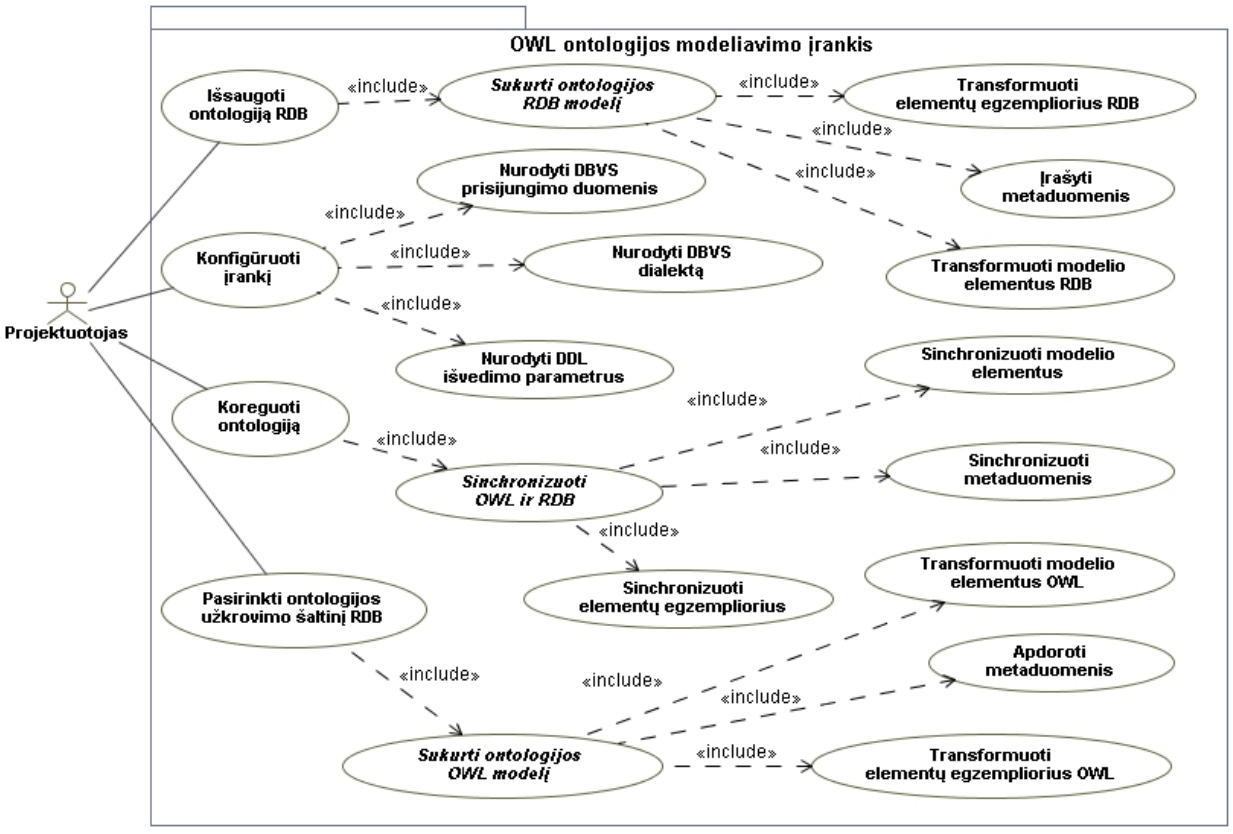
Prieš ontologijos transformavimą ji turi būti patikrinta validatoriumi, kuris padės aptikti neatitikimus.

4. Transformavimo sistemos reikalavimai

4.1. Reikalavimų specifikacija

Transformavimo sistemos funkciniai reikalavimai aprašomi panaudojimo atvejų diagrama ir PA specifikacijomis (22 pav.) Įrankis turėtų nuskaityti ontologijos aprašą *OWL* 2 kalba, atlikti *OWL* 2 kalbos sintaksės validavimą bei atvaizduoti modelio struktūrą. Vartotojas turėtų turėti galimybę sukurti objektus bei redaguoti dalykinės srities ontologiją. Pagrindinė sistemos savybė – ontologijos aprašų eksportavimas į reliacinių duomenų bazių schemas. Įrankio pagalba galima nuskaityti reliacinėje duomenų bazėje išsaugotą ontologiją bei atvaizduoti modelio struktūrą *OWL* 2. Panaudojimo atvejų specifikacijos pateiktos:

- Lentelė 4 PA „Išsaugoti ontologiją RDB“ specifikacija;
- Lentelė 5 PA „Konfigūruoti įrankį“ specifikacija;
- Lentelė 6 PA „Koreguoti ontologiją“ specifikacija;
- Lentelė 7 PA „Pasirinkti ontologijos užkrovimo šaltinį RDB“ specifikacija.



22 pav.Transformavimo sistemos panaudojimo atvejų diagrama

Lentelė 4 PA „Išsaugoti ontologiją RDB“ specifikacija

Panaudojimo atvejis	Išsaugoti ontologiją RDB.
Panaudojimo atvejo aprašymas	Tai panaudojimo atvejis, apibūdinantis ontologijos, atvaizduojamos modeliavimo įrankyje, išsaugojimą į reliacinę duomenų bazę.
Numeris	PA1
Aktorius	Projektuotojas
Sistema	*OWL* ontologijos modeliavimo įrankis
Prieš sąlyga	Ontologija yra atvaizduojama (aktyvi) modeliavimo įrankyje
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas naudojasi papildomu modeliavimo įrankio plėtiniu ir pasirenka išsaugoti *OWL* ontologiją į reliacinę duomenų bazę. 2. Vartotojas nurodo prisijungimo prie duomenų bazės parametrus. 3. Vartotojas pasirenka konfigūravimo parametrus.	1. Vartotojui parodomas dialogo langas, kuriame galima nurodyti prisijungimo prie duomenų bazės, konfigūravimo parametrus . 2. Sistema atlieka ontologijos metaduomenų modelio sudarymą 3. Sistema atlieka elementų transformaciją.Transformuojamos ontologijos klasės, savybės, apribojimai. Sudaromas RDB modelis. 4. Sistema atlieka elementų egzempliorių transformaciją.Transformuojami duomenys. Sudaromas RDB modelis. 5. Sistema prisijungia prie duomenų bazių valdymo sistemos serverio ir duomenų bazės. 6. Sistema atlieka modelių (elementai,duomenys,metaduomenys) eksportavimo į RDB procesą.
Po sąlyga	Ontologija išsaugota reliacinėje duomenų bazėje.
Alternatyvos (nesėkmės atvejai)	1. Duomenų bazių valdymo sistemos serveris neveikia. 2. Sistemai nesuteiktos teisės prisijungti prie duomenų bazių valdymo sistemos serverio. 3. Vartotojas nurodo neteisingus prisijungimo duomenis 4. Nepavyko sudaryti bent vieno iš modelių (elementų, egzempliorių, metaduomenų)
Vykdymo variantai	1. Vartotojas pasirenka išsaugoti aktyvią modeliuojamą ontologiją į reliacinę duomenų bazę.
Veiklos taisyklės	1. Vartotojas privalo nurodyti teisingą duomenų bazių valdymo sistemos serverio adresą. 2. Sistemai turi būti suteiktos pilnos teisės dirbti su duomenų baze.
Specialūs (nefunkciniai) reikalavimai	Išsaugojimas turi būti galimas į įvairias duomenų bazių valdymo sistemas
Kitos sistemos, su kuriomis sąveikauja sistema vykdydama PA	Vartotojo pasirinkta duomenų bazių valdymo sistema. *OWL* modelio valdiklis
Ryšiai su kitais PA	Detalizuojantis panaudojimo atvejis: „Sukurti ontologijos RDB modelį“, kuris apibendrina atvejus: „Transformuoti elementus RDB “ „Transformuoti elementų egzempliorius RDB “ „Įrašyti metaduomenis“

Lentelė 5 PA „Konfigūruoti įrankį“ specifikacija

Panaudojimo atvejis	Konfigūruoti įrankį.
Panaudojimo atvejo aprašymas	Tai panaudojimo atvejis, apibūdinantis transformavimo sistemos konfigūravimo funkciją.
Numeris	PA2
Aktorius	Projektuotojas
Sistema	*OWL* ontologijos modeliavimo įrankis
Prieš sąlyga	Ontologija yra atvaizduojama (aktyvi) modeliavimo įrankyje. Nėra išsaugota aktyvi įrankio konfigūracija. Inicijuojamas ontologijos išsaugojimas RDB
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas naudojasi papildomu modeliavimo įrankio plėtiniu ir pasirenka išsaugoti *OWL* ontologiją į reliacinę duomenų bazę. 2. Vartotojas nurodo prisijungimo prie duomenų bazės parametrus. 3. Vartotojas pasirenka konfigūravimo parametrus.	1. Vartotojui parodomas dialogo langas, kuriame galima nurodyti prisijungimo prie duomenų bazės, konfigūravimo parametrus . 2. Sistema atlieka konfigūracijos išsaugojimą atmintyje
Po sąlyga	Išsaugota aktyvi transformavimo sistemos konfigūracija.
Alternatyvos (nesėkmės atvejai)	1. Nurodyti klaidingi konfigūracijos, prisijungimo duomenys.
Vykdymo variantai	1. Vartotojas pasirenka išsaugoti aktyvią modeliuojamą ontologiją į reliacinę duomenų bazę.
Veiklos taisyklės	1. Vartotojas privalo nurodyti teisingą duomenų bazių valdymo sistemos serverio adresą. 2. Vartotojas privalo nurodyti teisingus konfigūravimo parametrus.
Specialūs (nefunkciniai) reikalavimai	-
Kitos sistemos, su kuriomis sąveikauja sistema vykdydama PA	Vartotojo pasirinkta duomenų bazių valdymo sistema.
Ryšiai su kitais PA	Apibendrinantis panaudojimo atvejis. Apibendrinami PA: „Nurodyti DBVS prisijungimo duomenis“ „Nurodyti DBVS dialektą“ „Nurodyti DDL išvedimo parametrus“

Lentelė 6 PA „Koreguoti ontologiją“ specifikacija

Panaudojimo atvejis	Koreguoti ontologiją.
Panaudojimo atvejo aprašymas	Tai panaudojimo atvejis, apibūdinantis ontologijos koregavimo RDB funkciją.
Numeris	PA3
Aktorius	Projektuotojas
Sistema	*OWL* ontologijos modeliavimo įrankis
Prieš sąlyga	Ontologija yra atvaizduojama (aktyvi) modeliavimo įrankyje. Ontologijos šaltinis – RDB. Inicijuojamas ontologijos išsaugojimas RDB
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas naudojasi modeliavimo įrankiu ir koreguoja aktyvią ontologiją, kurios šaltinis RDB 2. Vartotojas naudojasi papildomu modeliavimo įrankio plėtiniu ir pasirenka išsaugoti *OWL* ontologiją į reliacinę duomenų bazę. 3. Vartotojas nurodo prisijungimo prie duomenų bazės parametrus. 4. Vartotojas pasirenka konfigūravimo parametrus.	1. Vartotojui parodomas dialogo langas, kuriame galima nurodyti prisijungimo prie duomenų bazės, konfigūravimo parametrus . 2. Sistema atlieka sudaryto ontologijos metaduomenų modelio bei pakeitimų *OWL* modelyje sinchronizavimą. Metaduomenų modelyje adaptuojami pakeitimai 3. Sistema atlieka jau transformuotų elementų pakeitimų sinchronizaciją. Sudaromas adaptuotas RDB modelis. 4. Sistema atlieka elementų egzempliorių sinchronizavimą. Sudaromas adaptuotas RDB modelis. 5. Sistema prisijungia prie duomenų bazių valdymo sistemos serverio ir duomenų bazės. 6. Sistema atlieka adaptyvių modelių (elementai,duomenys,metaduomenys) eksportavimo į RDB procesą.
Po sąlyga	Ontologija išsaugota reliacinėje duomenų bazėje.
Alternatyvos (nesėkmės atvejai)	1. Duomenų bazių valdymo sistemos serveris neveikia. 2. Sistemai nesuteiktos teisės prisijungti prie duomenų bazių valdymo sistemos serverio. 3. Vartotojas nurodo neteisingus prisijungimo duomenis 4. Nepavyko sudaryti bent vieno iš modelių (elementų, egzempliorių, metaduomenų)
Vykdymo variantai	1. Vartotojas pasirenka išsaugoti aktyvią modeliuojamą ontologiją į reliacinę duomenų bazę.
Veiklos taisyklės	1. Vartotojas privalo nurodyti teisingą duomenų bazių valdymo sistemos serverio adresą. 2. Sistemai turi būti suteiktos pilnos teisės dirbti su duomenų baze.
Specialūs (nefunkciniai) reikalavimai	Sinchronizavimas turi būti galimas į įvairias duomenų bazių valdymo sistemas
Kitos sistemos, su kuriomis sąveikauja sistema vykdydama PA	Vartotojo pasirinkta duomenų bazių valdymo sistema. *OWL* modelio valdiklis, RDB modelio, RDB metaduomenų modelio valdikliai
Ryšiai su kitais PA	Detalizuojantis panaudojimo atvejis: „Sinchronizuoti *OWL* ir RDB“, kuris apibendrina atvejus: „Sinchronizuoti modelio elementus“ „Sinchronizuoti metaduomenis“ „Sinchronizuoti elementų egzempliorius“

Lentelė 7 PA „Pasirinkti ontologijos užkrovimo šaltinį RDB“ specifikacija

Panaudojimo atvejis	Pasirinkti ontologijos užkrovimo šaltinį RDB.
Panaudojimo atvejo aprašymas	Tai panaudojimo atvejis, apibūdinantis ontologijos,išsaugotos reliacinėje duomenų bazėje, atvaizdavimą modeliavimo įrankyje.
Numeris	PA4
Aktorius	Projektuotojas
Sistema	*OWL* ontologijos modeliavimo įrankis
Prieš sąlyga	Ontologija yra išsaugota reliacinėje duomenų bazėje naudojant transformavimo sistemą
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas naudojasi modeliavimo įrankio ontologijos atidarymo langu. Pasirenka šaltinį RDB 2. Vartotojas nurodo prisijungimo prie duomenų bazės parametrus. 3. Vartotojas pasirenka konfigūravimo parametrus. 4. Vartotojas pasirenka ontologiją	1. Vartotojui parodomas dialogo langas, kuriame galima nurodyti prisijungimo prie duomenų bazės, konfigūravimo parametrus . 2. Sistema atlieka ontologijos metaduomenų modelio nuskaitymą iš RDB. Sudaromas metaduomenų modelis. 3. Sistema atlieka transformuotų RDB elementų nuskaitymą. Sudaromas RDB modelis. 4. Sistema atlieka elementų egzempliorių nuskaitymą.Sudaromas RDB modelis. 5. Sistema atlieka *OWL* modelio sudarymą panaudojant metaduomenų, transformuotų elementų, egzempliorių modelius
Po sąlyga	Ontologija atvaizduojama modeliavimo įrankyje.
Alternatyvos (nesėkmės atvejai)	1. Duomenų bazių valdymo sistemos serveris neveikia. 2. Sistemai nesuteiktos teisės prisijungti prie duomenų bazių valdymo sistemos serverio. 3. Vartotojas nurodo neteisingus prisijungimo duomenis 4. Nepavyko sudaryti bent vieno iš modelių (elementų, egzempliorių, metaduomenų). Duomenų bazėje duomenys sugadinti. 5. Nepavyko sudaryti *OWL* modelio.
Vykdymo variantai	1. Vartotojas pasirenka ontologijos šaltinį RDB.
Veiklos taisyklės	1. Vartotojas privalo nurodyti teisingą duomenų bazių valdymo sistemos serverio adresą. 2. Sistemai turi būti suteiktos pilnos teisės dirbti su duomenų baze.
Specialūs (nefunkciniai) reikalavimai	Nuskaitymas turi būti galimas iš įvairių DBVS
Kitos sistemos, su kuriomis sąveikauja sistema vykdydama PA	Vartotojo pasirinkta duomenų bazių valdymo sistema. *OWL* modelio valdiklis, RDB modelio, RDB metaduomenų modelio valdikliai
Ryšiai su kitais PA	Detalizuojantis panaudojimo atvejis: „Sukurti ontologijos *OWL* modelį“, kuris apibendrina atvejus: „Transformuoti modelio elementus *OWL*“ „Apdoroti metaduomenis“ „Transformuoti elementų egzempliorius *OWL*“

4.2. Reikalavimai vartotojo sąsajai

Transformavimo sistemos reikalavimus vartotojo sąsajai apibūdina navigavimo planas (23 pav.) bei ekranų eskizai.

Pradinis ontologijos modeliavimo įrankio „*Protégé 4.1*“ langas yra ontologijos pasirinkimas (24 pav.) Greta jau egzistuojančių ontologijos užkrovimo šaltinių atvaizduojama užkrovimo iš duomenų bazės galimybė.

Pasirinkus ontologijos šaltinį RDB, atidaromas įrankio konfigūravimo langas (25 pav.) Pateikiama sąsaja prisijungimo prie duomenų bazės bei transformavimo sistemos parametrai.

Įvedami laukai (prisijungimo duomenys):

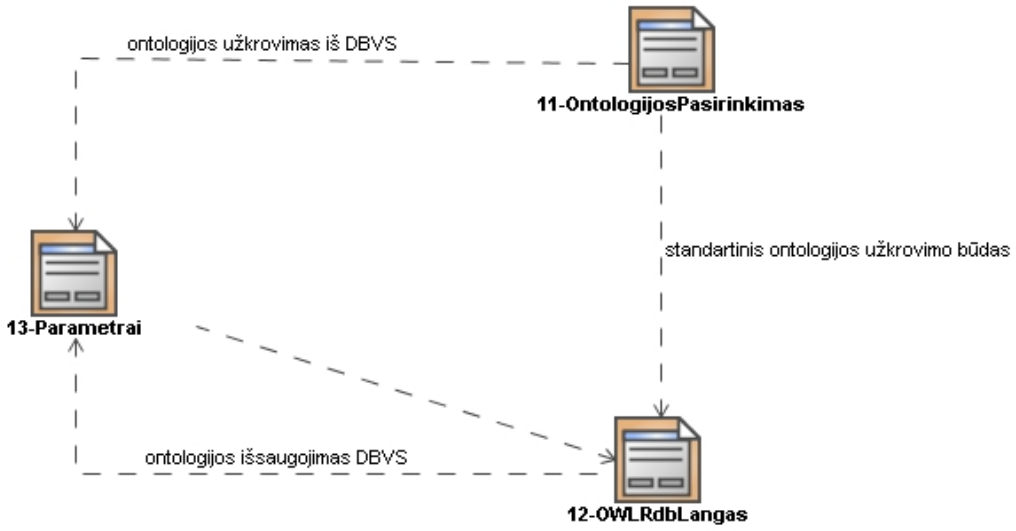
- *Host name* – duomenų bazių valdymo sistemos kompiuterio vardas
- *Port* – dbvs sąsajos portas
- *Service name* – duomenų bazės pavadinimas
- *User name* – vartotojo, schemos vardas
- *Pasword* – slaptažodis
- *Database dialect* – pasirinkimo laukas galimų dbvs tipų (Oracle, MS SQL)
- *Ontology* – duomenų bazėje išsaugotos ontologijos pavadinimas

Įvedami laukai (transformavimo konfigūravimo duomenys):

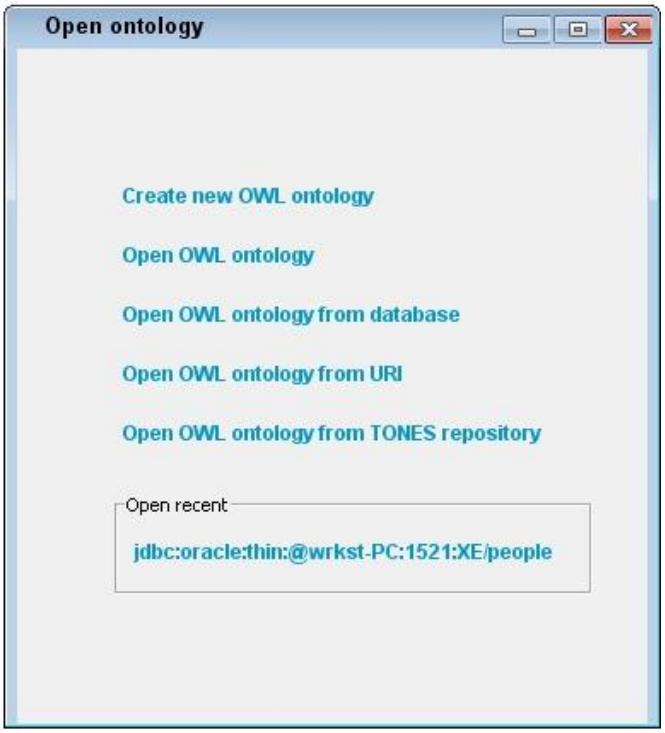
DDL generavimo parametrai: *SQL*(galimi pasirinkimai) : rodyti *SQL*, formatuoti *SQL*, rodyti *SQL* komentarus.

RDB lentelių bei metaduomenų modelio klasių surišimo parametrai: *OWL RDB mapping* (galimi pasirinkimai): naudoti numatytą surišimą, naudoti nurodytą surišimo failą.

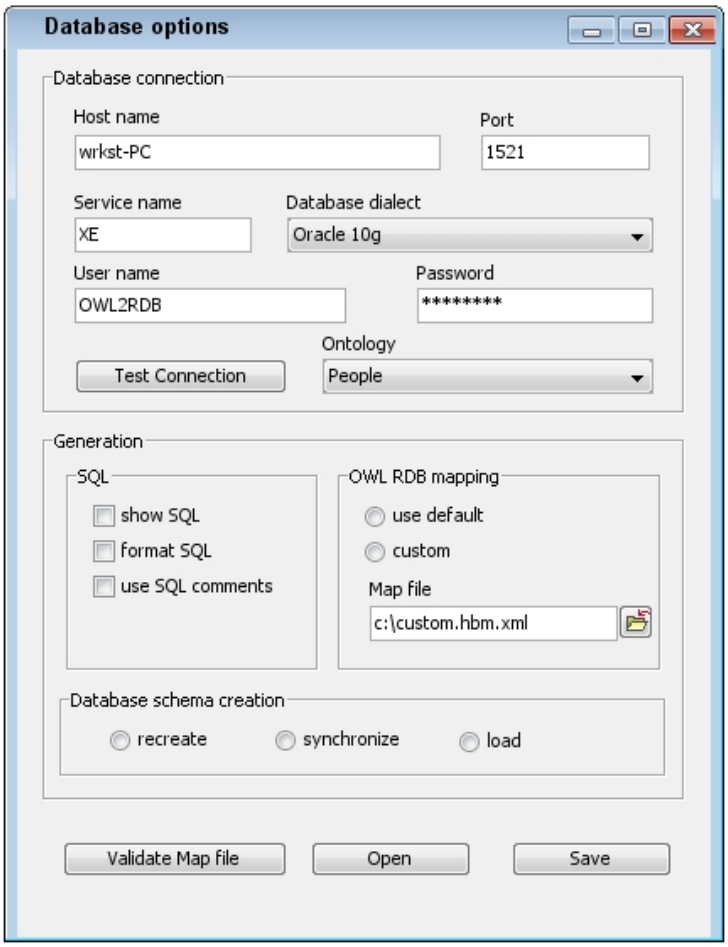
Duomenų bazės kūrimo metodika**:** *Database schema creation* (galimi pasirinkimai): perkūrimas, sinchronizavimas, užkrovimas.



23 pav. Transformavimo sistemos navigavimo planas



24 pav. Ontologijos pasirinkimo langas

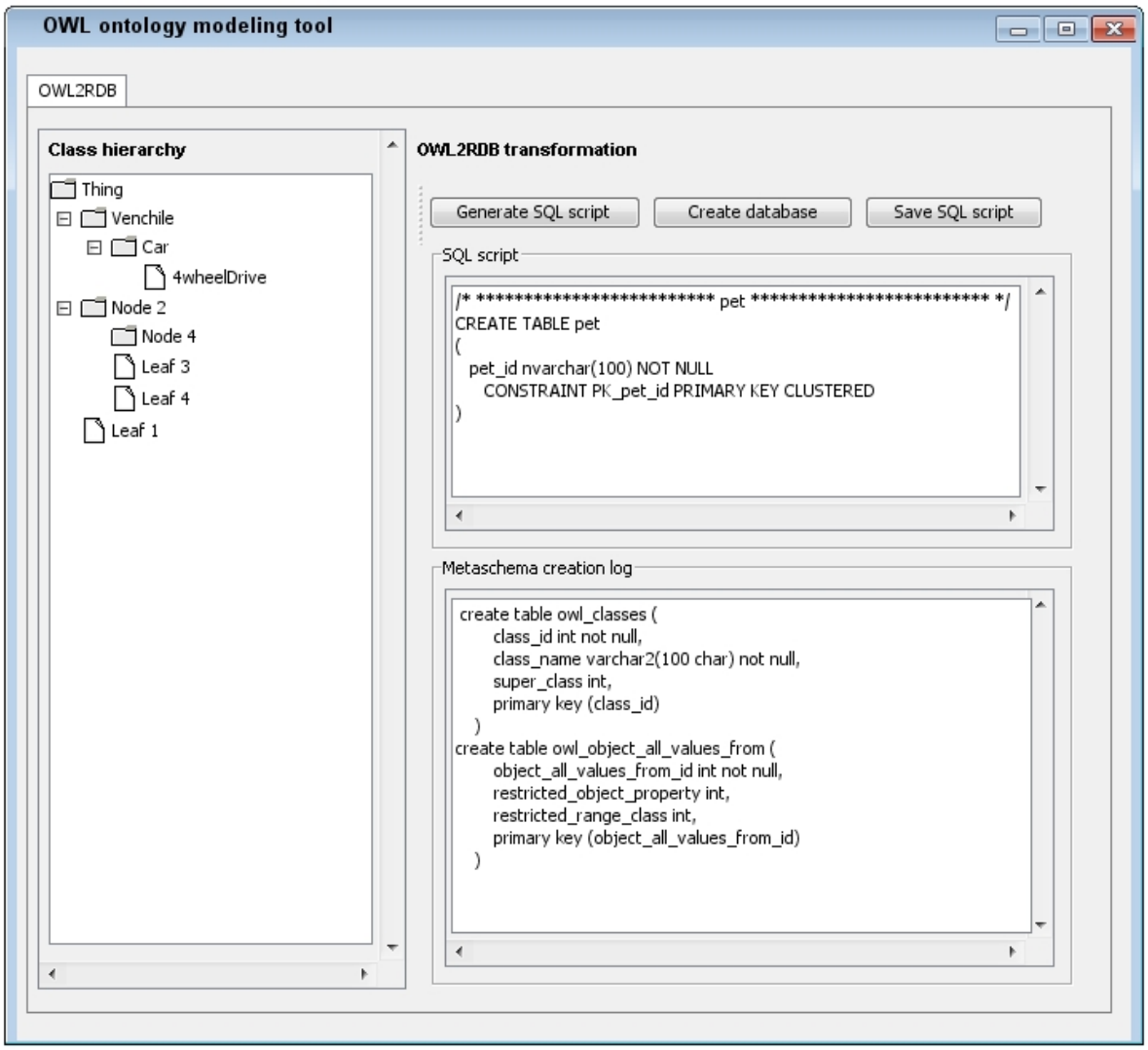


25 pav. Įrankio konfigūravimo langas

Nurodžius įrankio konfigūraciją bei pasirikus „Atidaryti“ vykdomas PA „Pasirinkti ontologijos užkrovimo šaltinį RDB“, kurio metu sukuriamas bei atvaizduojamas *OWL* modelis.

Pasirikus alternatyvų ontologijos šaltinį, į transformavimo sistemos langą galima patekti naviguojant į kortelę „*OWL2ToRDB*“ .(26 pav.)

Kortelės lange pateikiama aktyvios ontologijos klasių hierarchija bei transformavimo sistemos sąsaja.



26 pav. Transformavimo įrankio langas

4.3. Nefunkciniai reikalavimai

Transformavimo įrankiui buvo iškelti šie nefunkciniai reikalavimai:

- Sistema turi veikti patikimai, t.y. jos darbas neturi būti nutraukiamas ir duomenys negali būti sugadinami dėl ontologinio aprašo sintaksės klaidų;
- Sistema privalo veikti nepriklausomai nuo naudojamos platformos, t.y. turi būti galimybė ja naudotis įvairiose operacinėse sistemose;

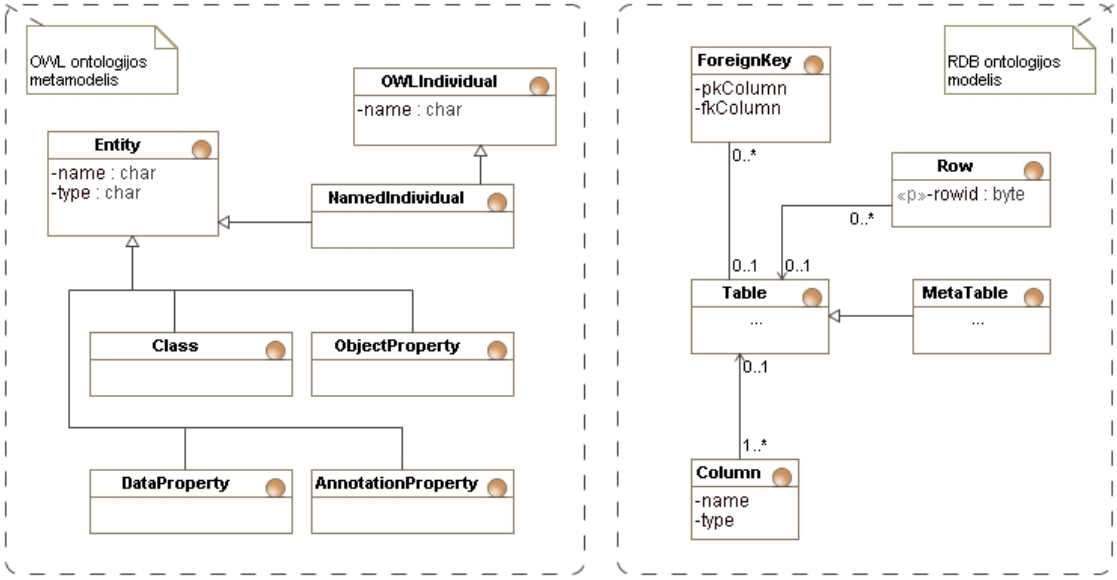
- Sistema turi būti palaikoma, t.y. naujos sistemos galimybės gali būti nesudėtingai įtraukiamos. Turi būti galimybė integruoti transformavimo įrankį su jau egzistuojančiomis ontologijų valdymo sistemomis;
- Sistema privalo teikti pagalbą vartotojui esant nesuprantamai situacijai ar klaidai;
- Sistema turi būti suderinama su įvairiomis duomenų bazių valdymo sistemomis; pvz. *MS SQL Server, Oracle, MySql* ir kitomis.

5. Ontologijos aprašo transformavimo įrankio realizacijos projektas

5.1. Sistemos architektūra

5.1.1. Transformavimo sistemos dalykinės srities esybių klasių modelis

Analizuojant dalykinę sritį nustatyti veiklos objektai, detalizuoti bei peržiūrėti jų metamodeliai. Siekiant apibrėžti reikalavimus transformavimo sistemai, bei sudaryti bendrą, į veiklos tikslus orientuotą elgsenos modelį, naudojamos abstrakcijos arba aukščiausio lygio objektai. Bendrą modelį sudaro *OWL* 2 metamodelis, *OWL* 2 metamodelio bei transformuoti *OWL* 2 elementai reliacinėje duomenų bazėje (27 pav.)



27 pav.Dalykinės srities esybių modelis

5.1.2. Analizės modelis

Analizuojant transformavimo sistemai keliamus reikalavimus, esminius konceptus, sudarytas panaudojimo atvejų analizės klasių modelis – *robastiškumo* diagrama (28 pav.) Pagal vartotojo sąsajai keliamus reikalavimus (sąsajos prototipą) nustatytos ribinės klasės (<<*boundary*>>):

- Ontologijos pasirinkimo langas;
- Parametrų langas;
- *OWL2ToRDB* (sistemos langas);
- Modeliavimo įrankio langas

Pagal numatomą sistemos elgseną bei panaudojimo atvejų specifikaciją identifikuotos reikalingos valdiklių klasės (<<*control*>>):

Vartotojo sąsajos:

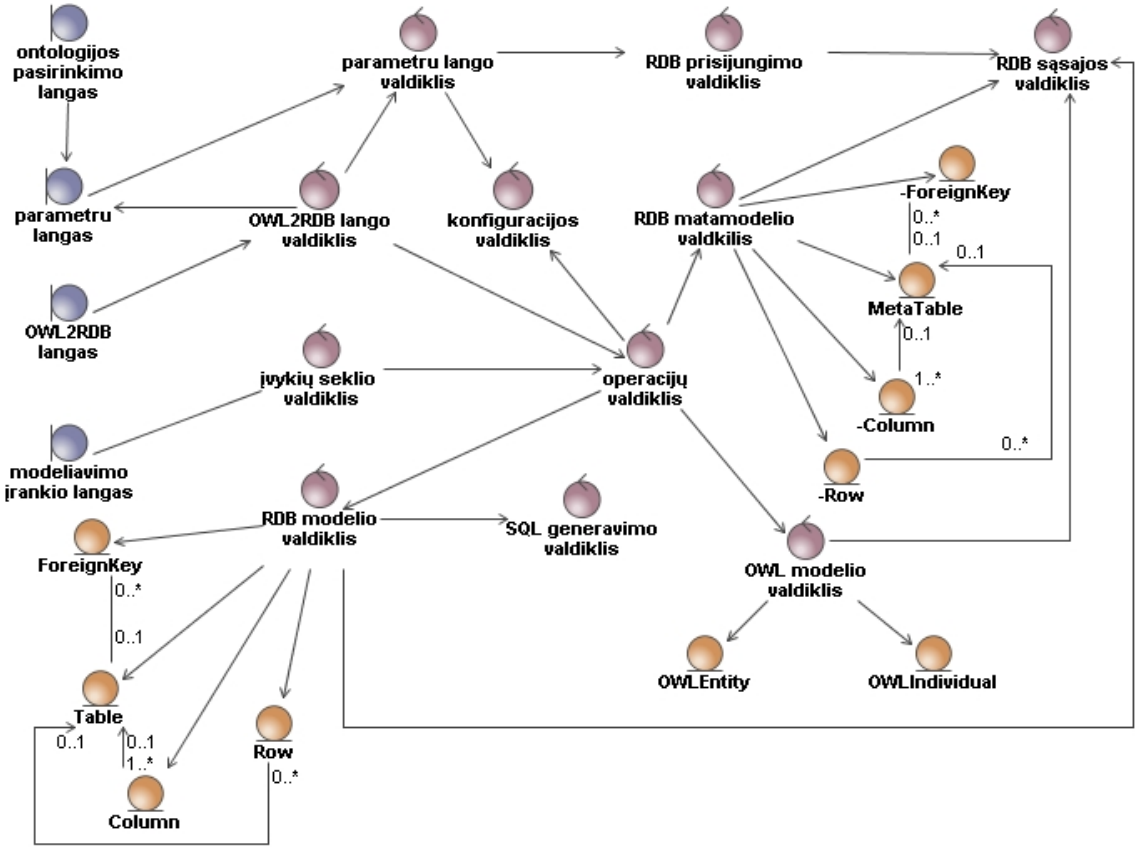
- Parametrų lango valdiklis,
- OWL2ToRDB lango valdiklis.

Veiklos logikos:

- RDB prisijungimo valdiklis;
- RDB sąsajos valdiklis;
- Konfigūracijos valdiklis;
- Įvykio seklio valdiklis;
- Operacijų valdiklis;
- RDB metamodelio valdiklis;
- RDB modelio valdiklis;
- SQL generavimo valdiklis;
- *OWL* modelio valdiklis.

Dalykinės srities objektus vaizduoja esybių klasės (<<*entity*>>)

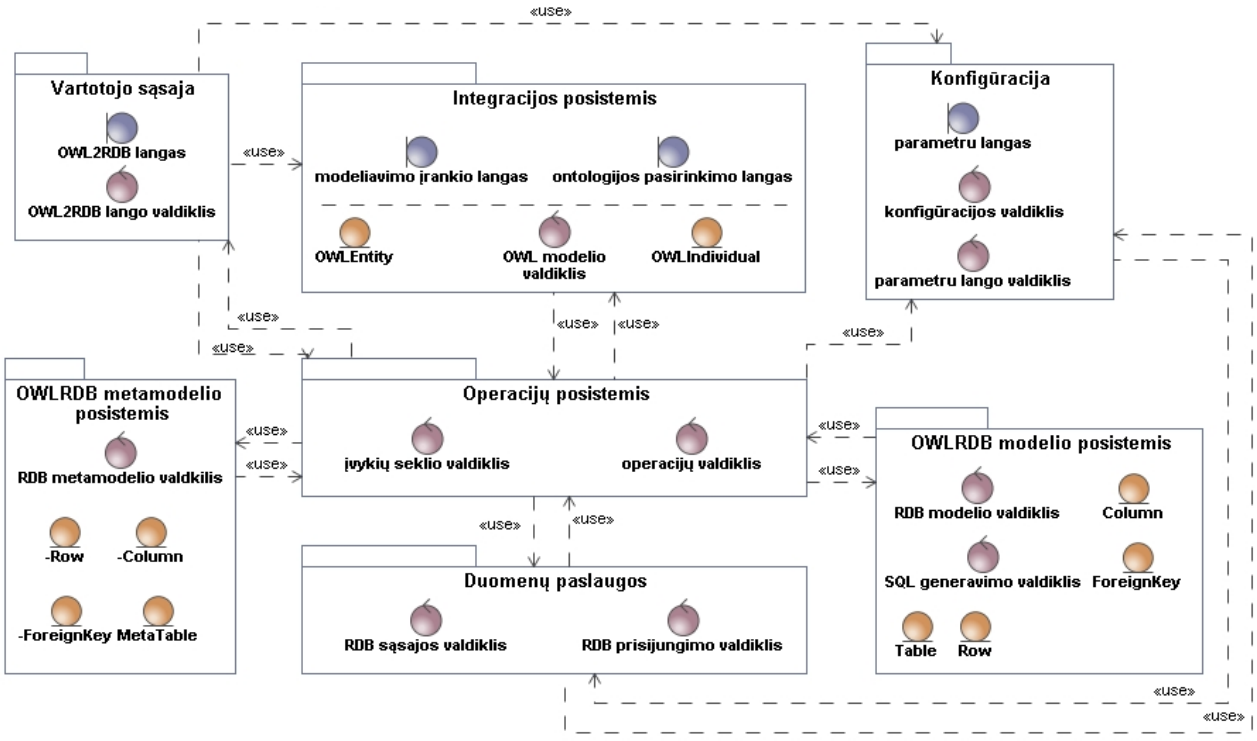
Panaudojimo atvejis „Išsaugoti ontologiją RDB“ (22 pav.) šioje schemoje funkcionuoja taip: Projektuotojas pasirenka aktyvią ontologiją (paprastai tą, su kuria dirbama), transformavimo sistemos lange inicijuoja veiksmą „Išsaugoti“. *OWL2ToRDB* lango valdiklis atidaro parametrų langą. Projektuotojas įveda prisijungimo prie duomenų bazės duomenis bei inicijuoja veiksmą „Išsaugoti“. Parametrų lango valdiklis kreipiasi į RDB prisijungimo valdiklį duomenų bazės sesijos sukūrimui. RDB prisijungimo valdiklis naudodamas atitinkamą RDB sąsajos valdiklį (priklauso nuo pasirinkto duomenų bazės dialekto), inicijuoja prisijungimo sesiją. Sėkmės atveju parametrų lango valdiklis kreipiasi į konfigūracijos valdiklį aktyvios konfigūracijos išsaugojimui. Po sėkmingo parametrų įvedimo *OWL2ToRDB* lango valdiklis kviečia operacijų valdiklį, kuris naudoja RDB metamodelio, *OWL* modelio, RDB modelio valdiklius. Atitinkami modelių valdikliai įgyvendina transformaciją bei sukuria metaduomenis.



28 pav. Analizės modelis

5.1.3. Loginė schema

Analizės klases ribojančių komponentų - posistemių identifikacija pateikiama (29 pav.) Pagrindiniai transformavimo sistemos posistemiai: vartotojo sąsaja, integracijos posistemis, konfigūracija, *OWLRDB* metamodelio posistemis, operacijų posistemis, *OWLRDB* modelio posistemis, duomenų paslaugos. Loginės architektūros modelyje pavaizduotos sąveikos tarp posistemių. Sąveikaujantys posistemiai sujungti ryšiu <<*use*>>.



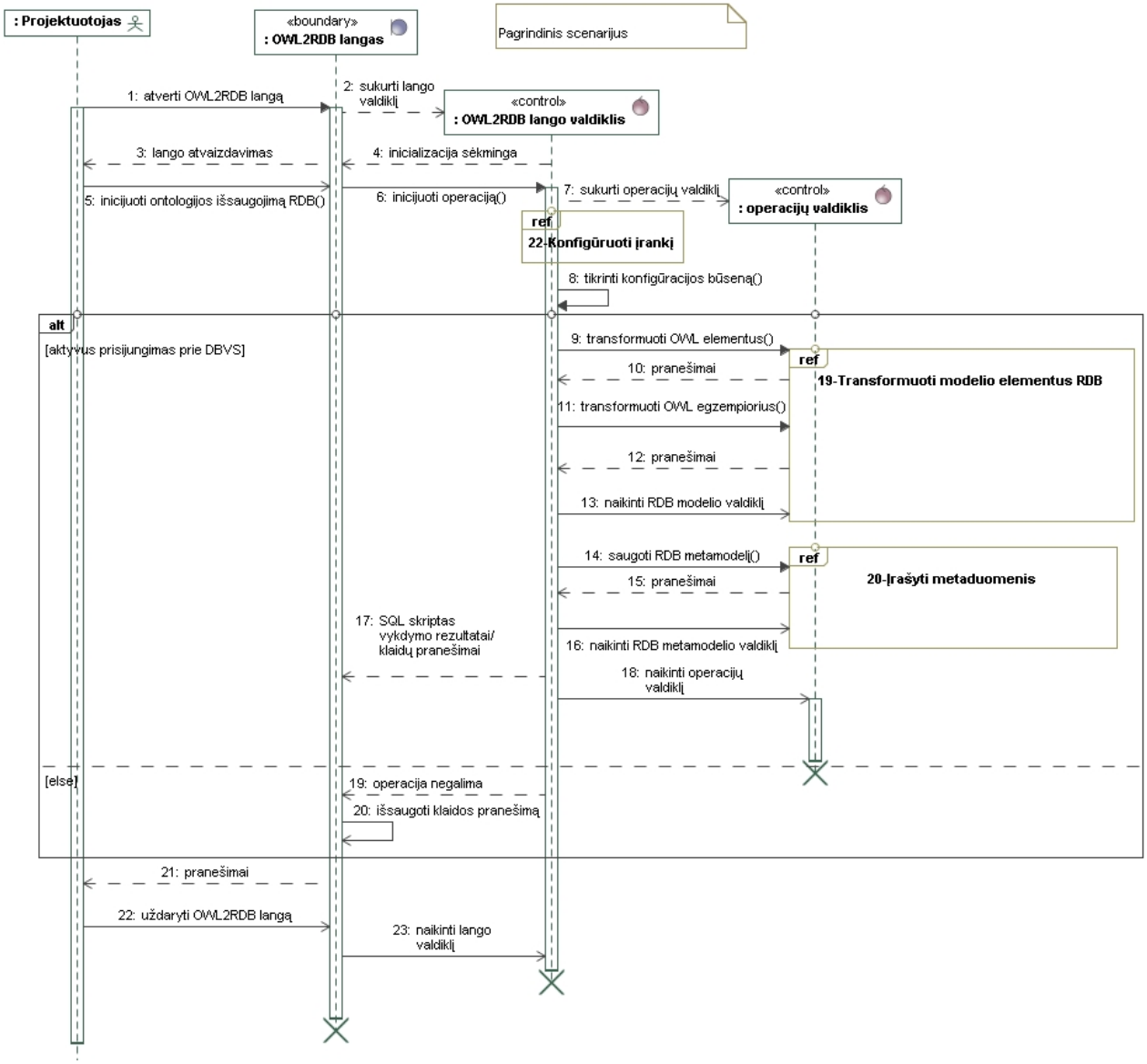
29 pav. Transformavimo sistemos loginė schema

5.2. Reikalavimų realizacijos modelis

5.2.1. Panaudojimo atvejų sekų diagramos

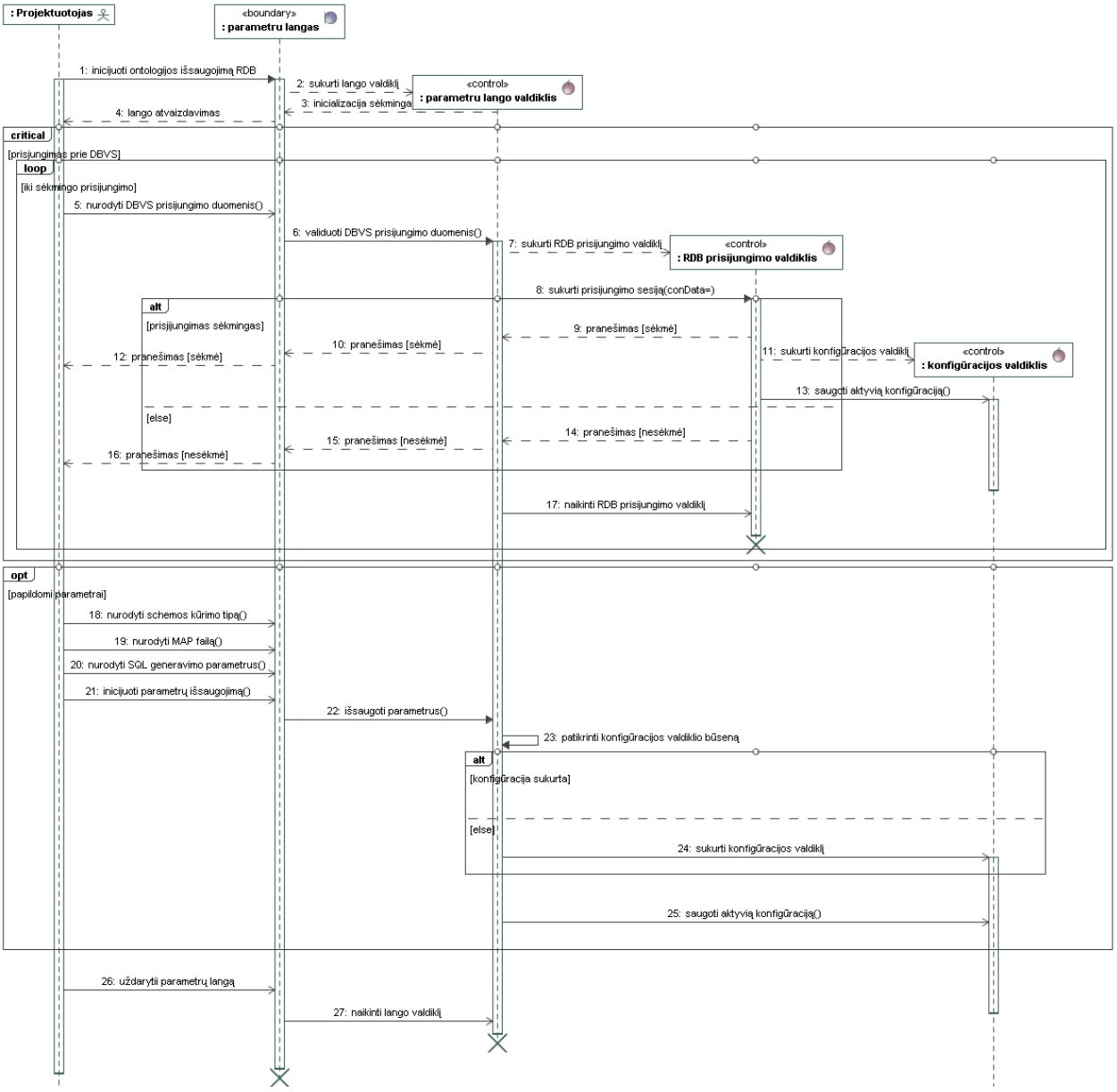
Transformavimo sistemos elgsena modeliuojama sekų diagramomis. Analizuojant panaudojimo atvejų modelį. (22 pav.) bei panaudojimo atvejų specifikacijose. (Lentelė 4 - Lentelė 7) aprašytus žingsnius, nustatytos analizės klasių operacijos bei detalizuota sąveika tarp valdiklių ir vartotojo sąsajos komponentų.

Panaudojimo atvejo „Išsaugoti ontologiją“ realizacijai pateikiama sekų diagrama (30 pav.)

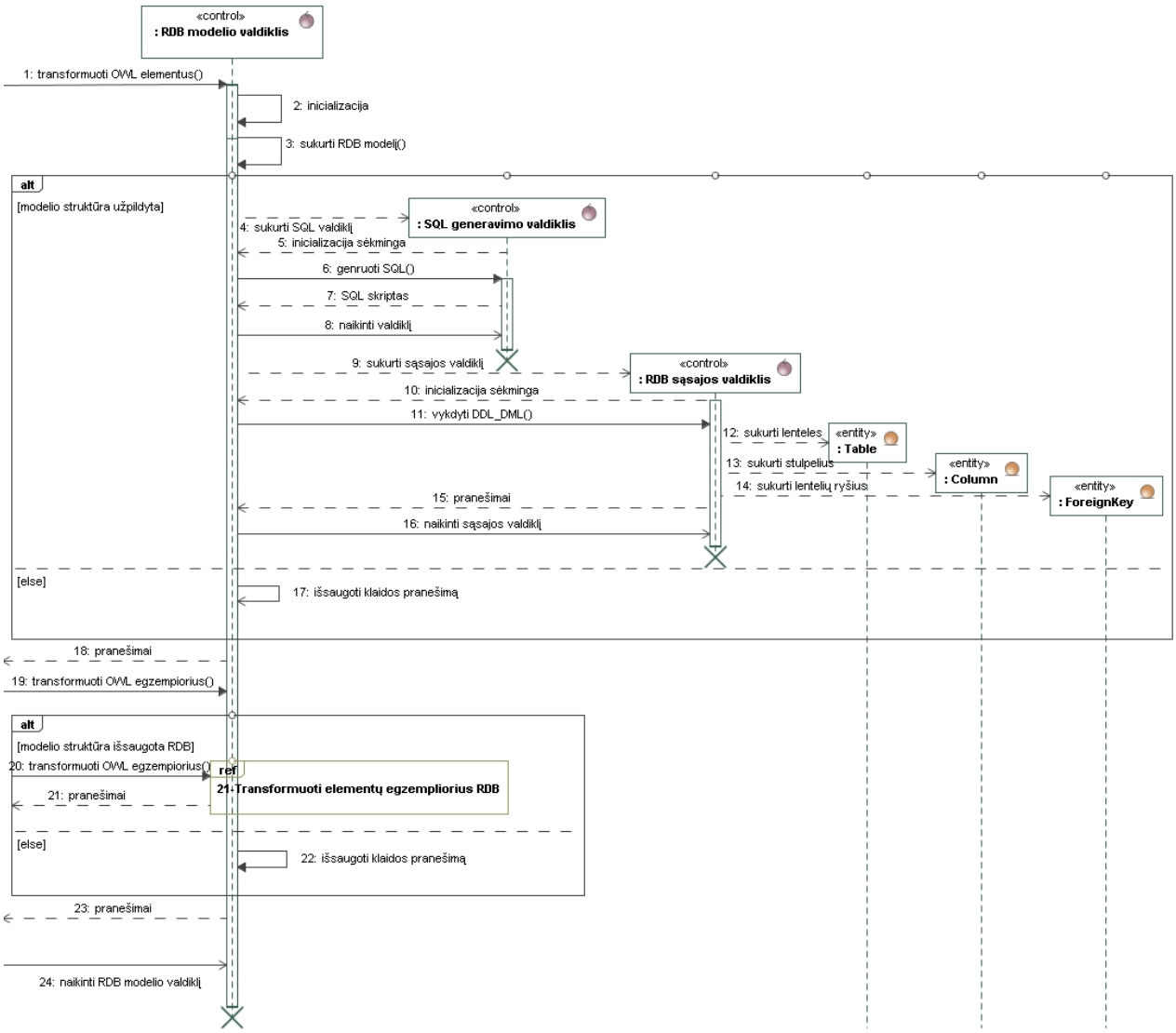


30 pav. Ontologijos išsaugojimo RDB sekų diagrama

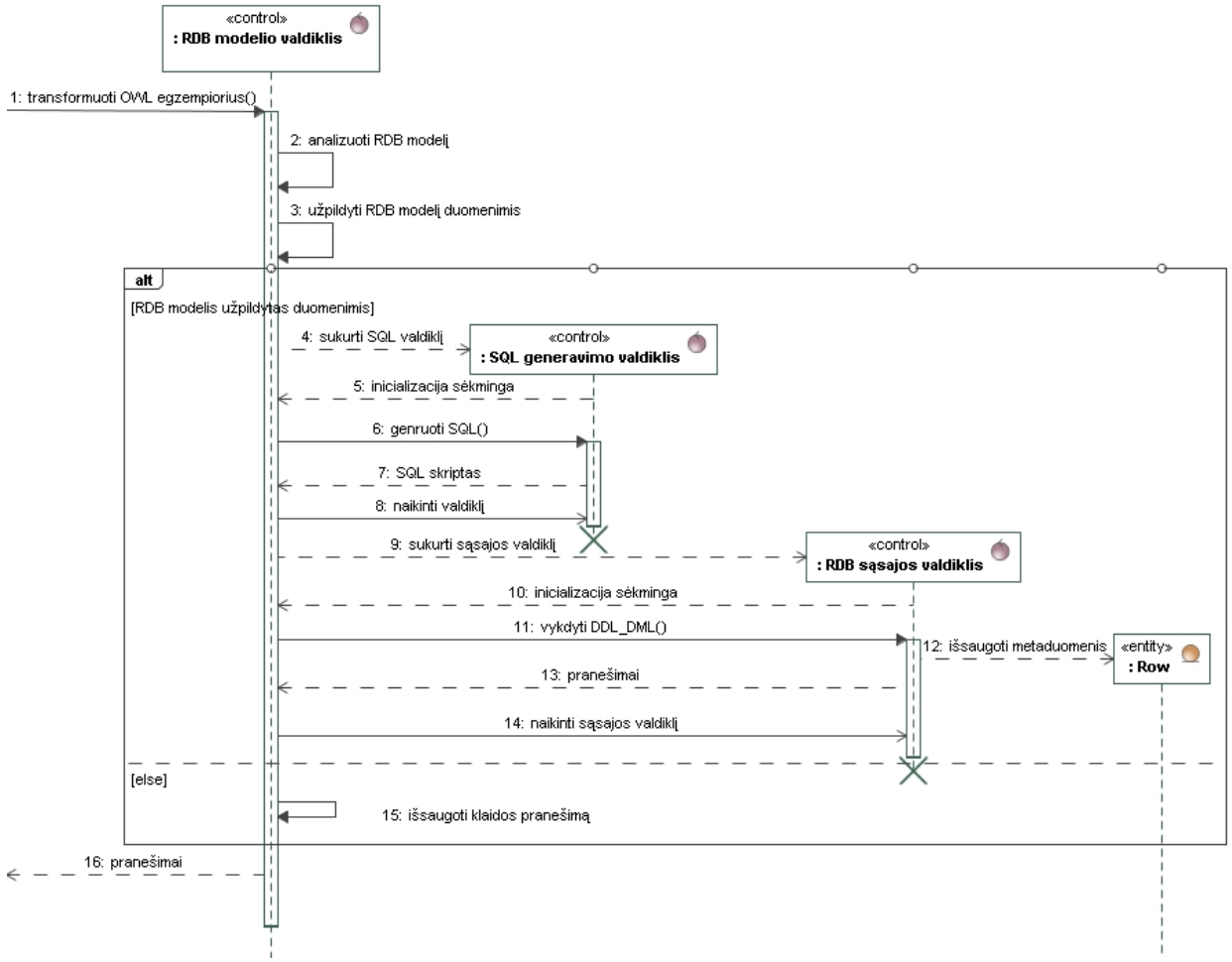
Sekų diagrama aprašo pagrindinį transformavimo sistemos elgsenos scenarijų, kurį sudaro elementai : „Konfigūruoti įrankį“, „Transformuoti modelio elementus RDB“, „Įrašyti metaduomenis“. Valdiklių klasės yra sukuriamos (inicializacija) bei naikinamos pasibaigus inicijuojamai operacijai. Projektuotojui *OWL2ToRDB* lango pagalba inicijavus veiksmą „Išsaugoti ontologiją“, lango valdiklis sukuria operacijų valdiklį bei inicijuoja įrankio konfigūravimą (31 pav.) Patikrinus aktyvios konfigūracijos būseną, nuosekliai inicijuojami operacijų valdiklio metodai: transformuoti *OWL* elementus (32 pav.), transformuoti *OWL* egzempliorius (33 pav.) Atlikus transformaciją, vykdomas metaduomenų modelio sudarymas bei išsaugojimas (34 pav.)



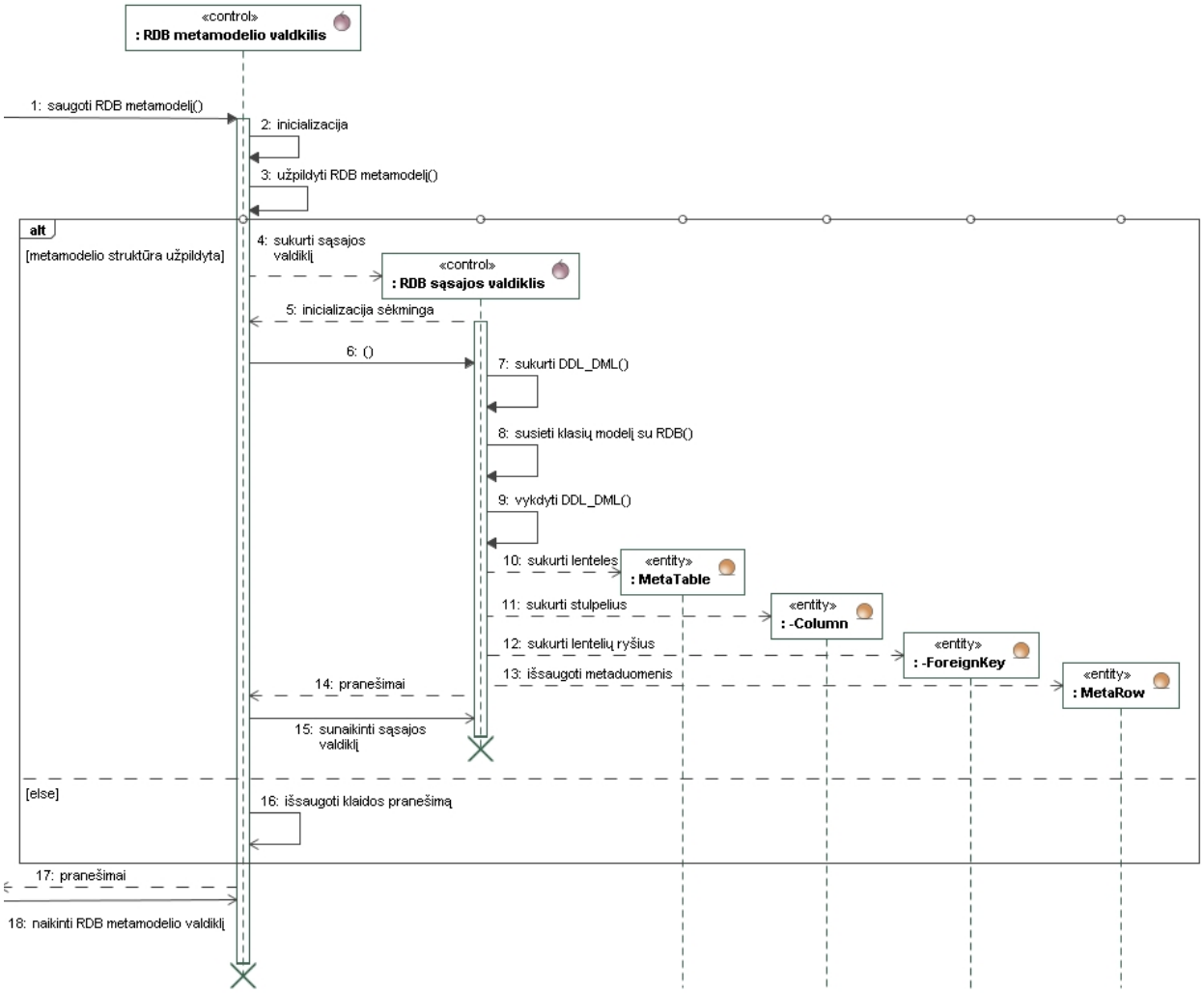
31 pav. Įrankio konfigūravimo sekų diagrama



32 pav. *OWL* elementų transformavimo sekų diagrama



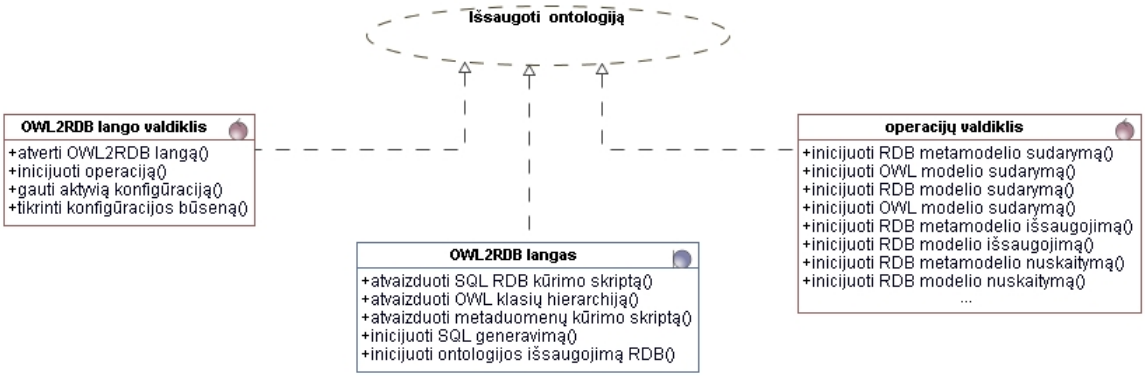
33 pav. *OWL* elementų egzempliorių transformavimo sekų diagrama



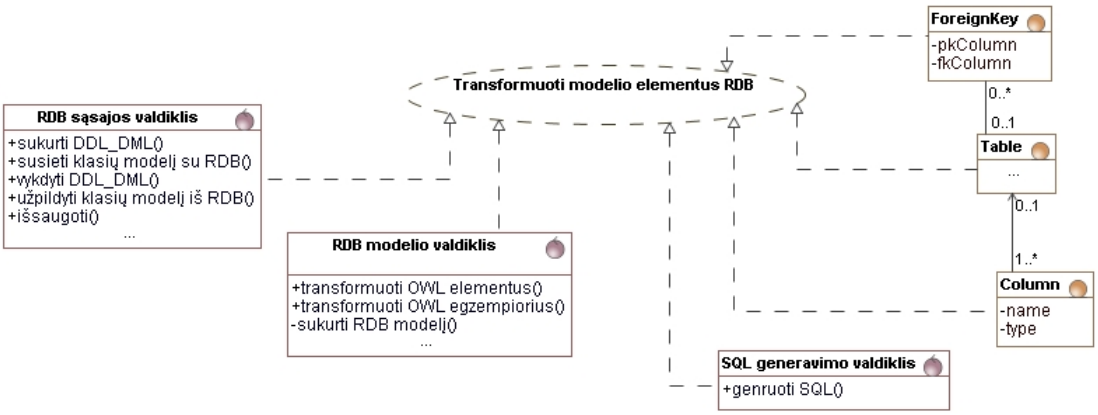
34 pav. Metaduomenų įrašymo sekų diagrama

5.2.2. Panaudojimo atvejų realizacijos

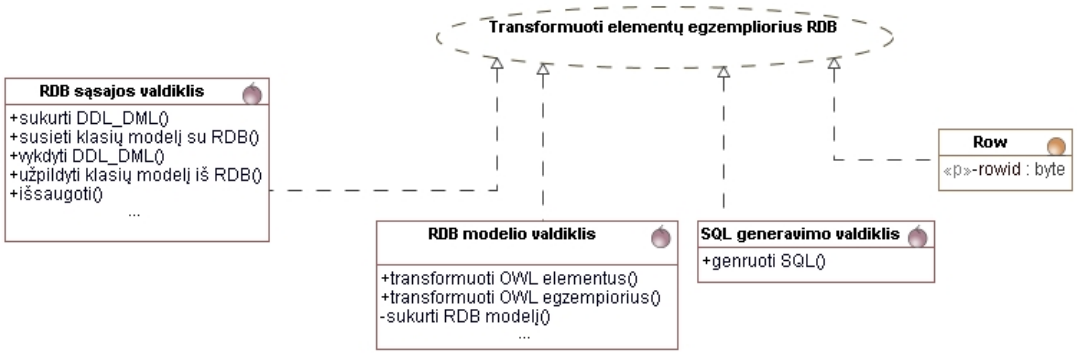
Transformavimo sistemos panaudojimo atvejai realizuojami analogiškomis projekto klasėmis. Panaudojimo atvejų realizavimo pateikiamos sekančiose paveiksluose: (35 pav.- 39 pav.)



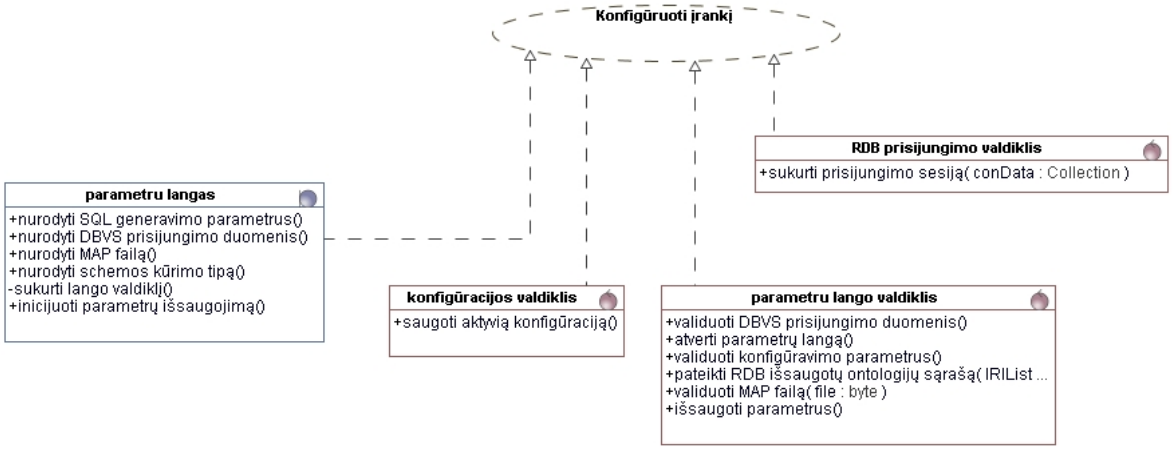
35 pav. Išsaugoti ontologiją panaudojimo atvejo realizacijos klasių diagrama



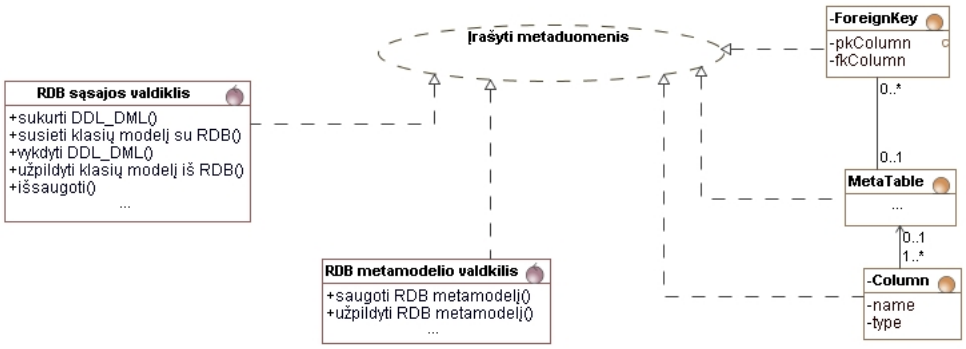
36 pav. Transformuoti modelio elementus RDB panaudojimo atvejo realizacijos klasių diagrama



37 pav. Transformuoti elementų egzempliorius panaudojimo atvejo realizacijos klasių diagrama



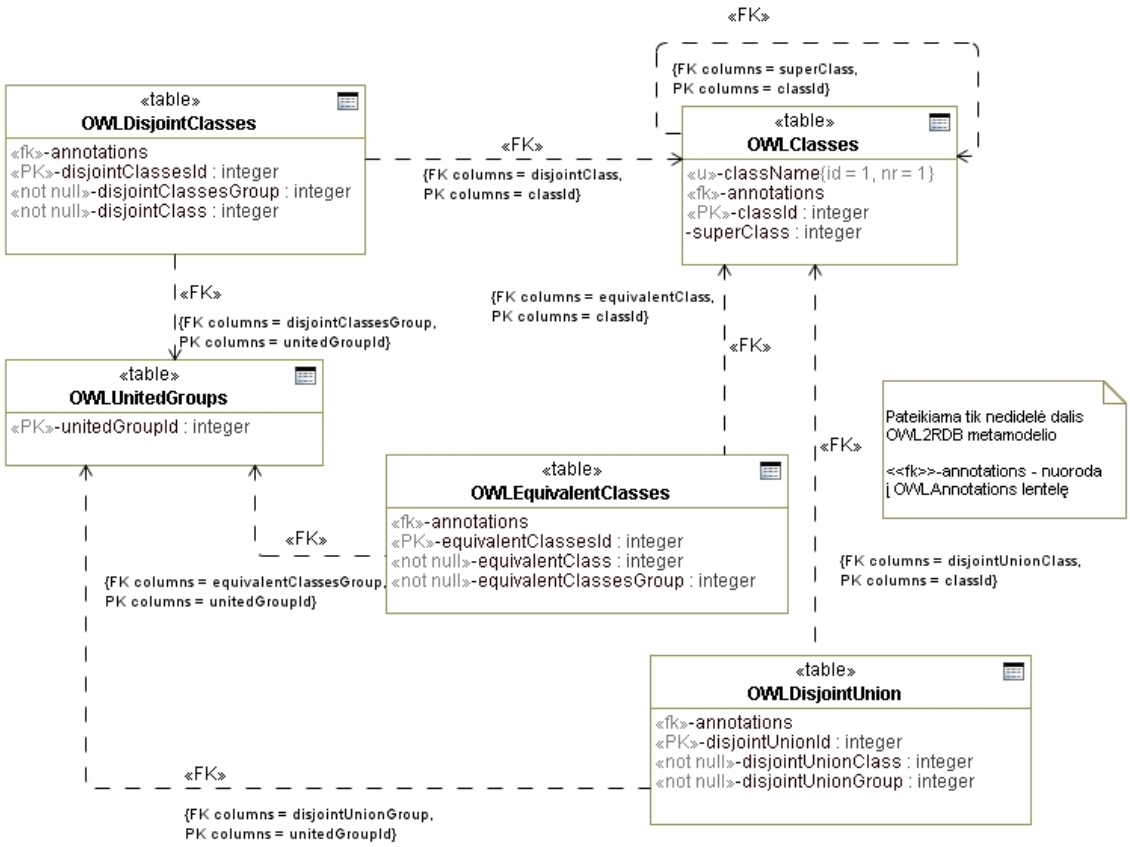
38 pav. Konfigūruoti įrankį panaudojimo atvejo realizacijos klasių diagrama



39 pav. Įrašyti metaduomenis panaudojimo atvejo realizacijos klasių diagrama

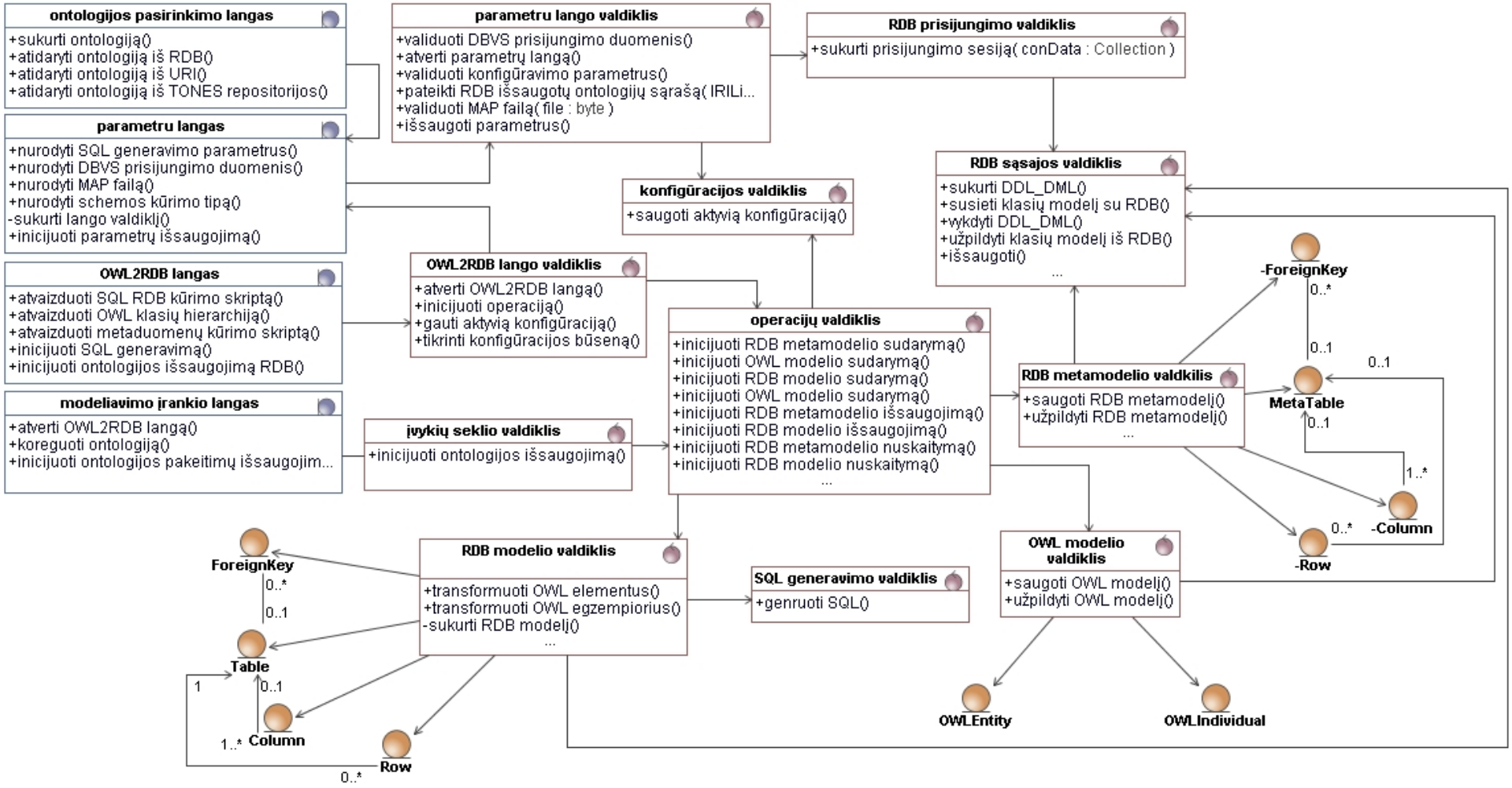
5.3. Duomenų bazės schema

Projektuojant transformavimo sistemą sudaromas statinis ontologijos metaduomenų reliacinėje duomenų bazėje modelis. Modelį sudaro pagrindiniai *OWL* elementai (esybės) (40 pav.) pateikiama dalis metaduomenų modelio.



40 pav. Duomenų bazės schemos fragmentas

5.4. Detalus projektas



41 pav. Transformavimo sistemos projekto klasių diagrama

Trijų lygių projekto klasių diagramoje (41 pav.) galima išskirti vartotojo sąsajos komponentus (elementai pažymėti <<*boundary*>> stereotipu), veiklos komponentus (<<*control*>>), duomenų komponentus (<<*entity*>>). Atitinkami elementų stereotipai identifikuoja klasių diagramos lygius : vartotojo paslaugos, veiklos paslaugos, duomenų paslaugos.

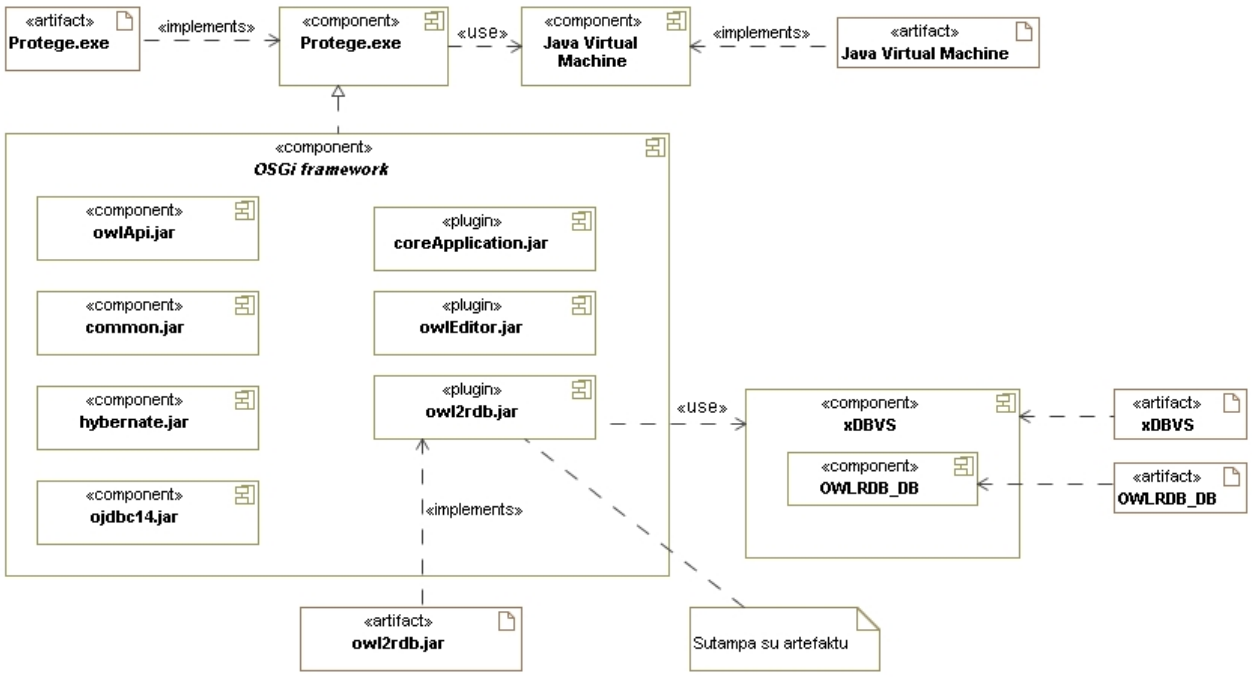
5.5. Realizacijos modelis

5.5.1. Komponentų modelis

Transformavimo įrankio komponentų modelis susideda iš vartotojo sąsajos modulio, „*Protégé*“ ontologijų kūrimo įrankio ir duomenų bazės modulio (42 pav.) Vartotojo sąsajos modulį sudaro *OSGi* aplinkoje integruoti komponentai skirti ontologijos *OWL* modelio manipuliavimui bei leidžia atlikti tokius veiksmus, kaip pasirinkti ontologijos aprašo failą, atvaizduoti ontologiją grafiškai, pasirinkti duomenų bazių serverį ir iššaukti kitas reikalingas komandas.

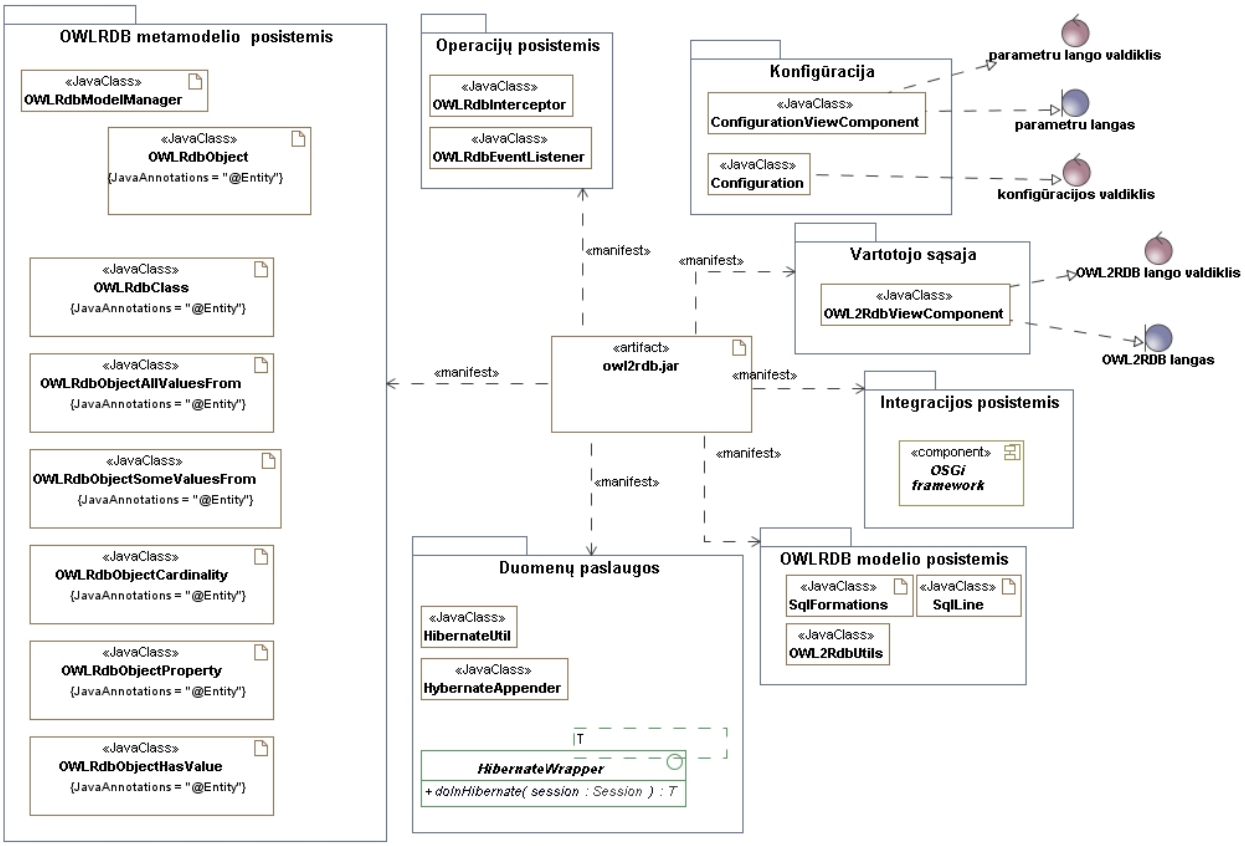
OSGi - standartizuota į komponentus orientuota operavimo aplinka, kurios pagrindinis ypatumas - komponentų paslaugų prieinamumas t.y bet kuris aplinkos komponentas gali naudoti visų kitų komponentų paslaugas. Ryšys <<use>> turėtų būti nukreipiamas nuo kiekvieno komponento į aplinkos komponentą (aplinka atlieka paslaugų saugyklos vaidmenį), bei į kiekvieną paslaugas teikiantį komponentą

Transformavimo įrankio realizacijos klasės sudaro rinkinį *OWL2rdb.jar* (<<*plugin*>>komponentą), kuris integruojamas modeliavimo įrankyje.



42 pav. Transformavimo sistemos komponentų modelis

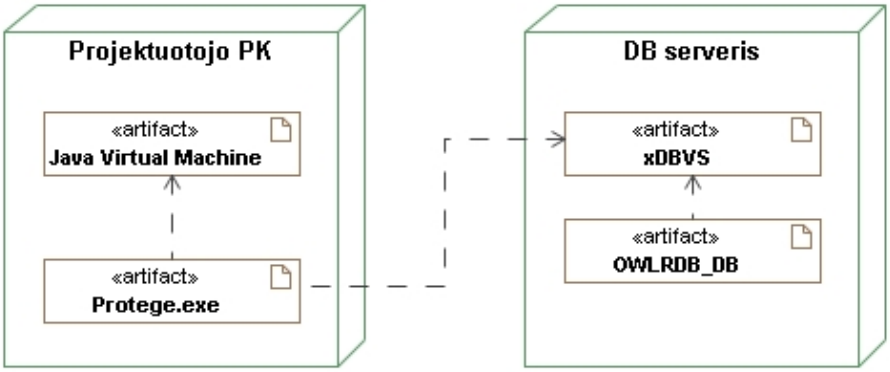
5.5.2. Realizacijos klasių modelis



43 pav. Realizacijos klasių modelis

Transformavimo sistemos realizacijos klasių modelį (43 pav.) sudaro duomenų klasės, langų valdiklių klasės, duomenų paslaugų klasės, operacijų klasės .

5.5.3. Diegimo modelis



44 pav. Artefaktų įdiegimo modelis

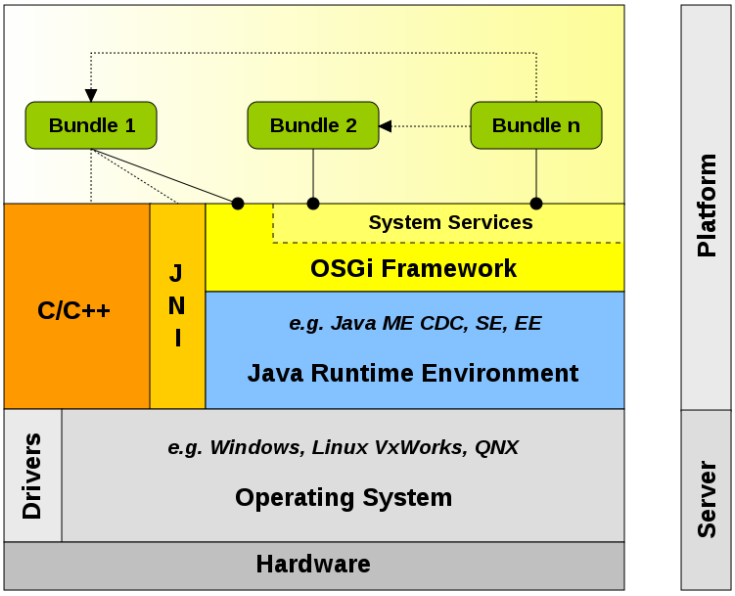
Transformavimo sistemai funkcionuoti reikalingi techninės įrangos komponentai pateikti (44 pav.) Kuriamas komponentas įdiegiamas ontologijų modeliavimo įrankyje *Protégé*, kuris funkcionuoja projektuotojo kompiuteryje. Įrankio naudojama aplinka – Java virtuali mašina. Duomenų bazių serveris gali būti diegiamas atskirai t.y kitame kompiuteryje. Transformuota ontologija saugoma sukurtoje duomenų bazėje *OWL*RDB_DB.

6. Transformavimo įrankio prototipo realizacija

6.1. Transformavimo įrankio realizacijos platforma bei integravimas

Transformavimo įrankiui realizuoti buvo pasirinkta *Java2SE* platforma. Pagrindinės pasirinkimo priežastys yra šios: Java kalbos paprastumas, platformos portabilumas, saugumas, patikimumas. Taip pat naudojant Java programavimo kalbą buvo galima praplėsti egzistuojantį atviro kodo *Protégé* modeliavimo įrankį, skirtą darbui su ontologijomis.

Ontologijų modeliavimo įrankį *Protégé* sudaro pagrindiniai moduliai, kurie skirti darbui su ontologija (*org.editor.protege.owl, org.protege.editor.core.application ir org.semanticweb.owl.owlapi,*) bei eilė papildomų (pasirinktinai įdiegiamų) modulių skirtingoms žinių modelio reprezentacijoms modeliuoti bei atvaizduoti. Visi moduliai integruojami į bendrą platformą *OSGi* – (angl. *Open Services Gateway initiative framework*) (42pav). Bendra *OSGi* karkaso sluoksnių architektūra pateikta (45pav).



45 pav. *OSGi* karkaso integracijos architektūra (*wiki*)

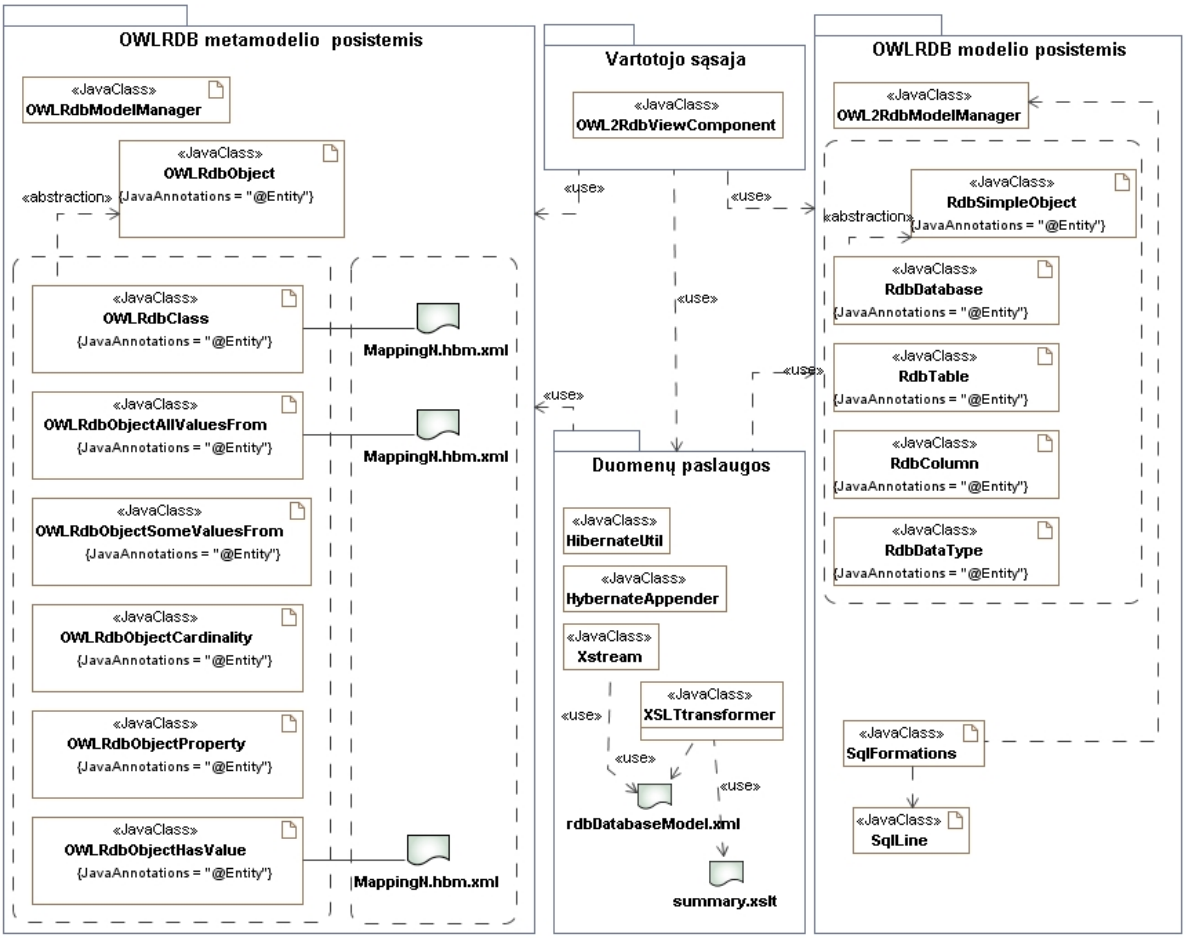
Rinkinys (angl. *bundle*) – grupė java klasių, papildomų konfigūracijos failų (*resources:.xml,.jar,.properties,.xslt*) bei *MANIFEST.MF* failas, kuriame aprašomi visi modulyje naudojami resursai. Transformavimo sistemos modulio konfigūracijos failai aprašomi (2 priede)

6.2. Transformavimo įrankio architektūra

Realizuotas transformavimo įrankis susideda iš vartotojo sąsajos modulio, *OWL* ontologijos metamodelio valdymo modulio, *OWL* ontologijos transformavimo į RDB valdymo modulio ir duomenų paslaugų modulio. Vartotojo sąsajos modulis skirtas palaikyti ryšį tarp projektuotojo ir sistemos, leidžia atlikti tokius veiksmus, kaip pasirinkti ontologijos aprašo failą,

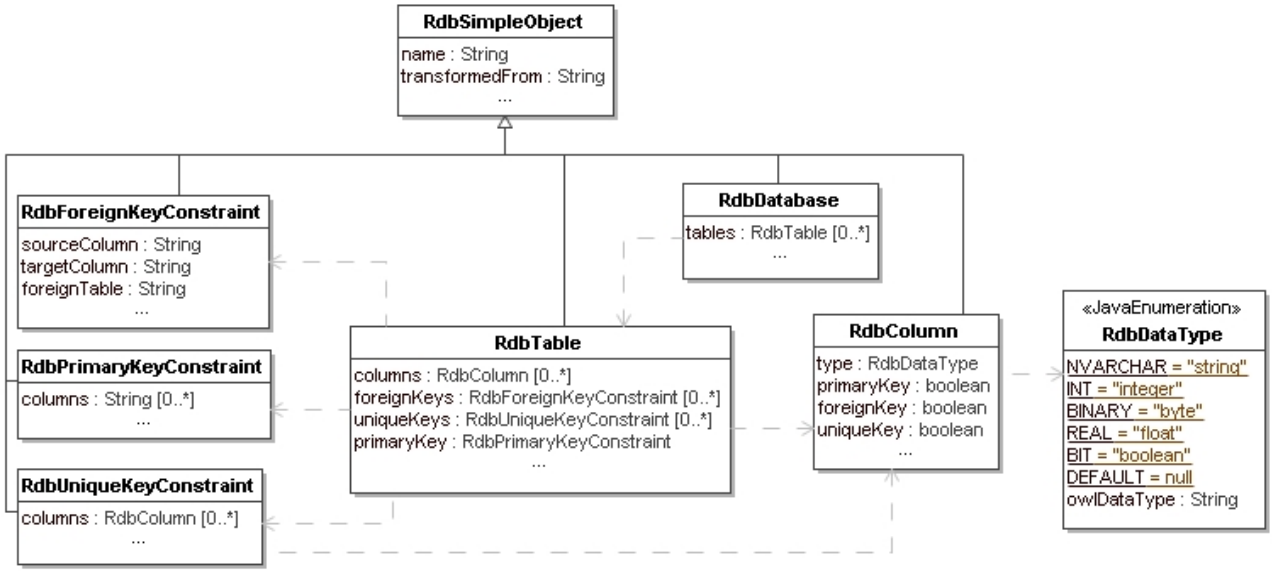
atvaizduoti ontologiją grafiškai, pasirinkti duomenų bazių serverį ir iššaukti kitas reikalingas komandas. *OWL* ontologijos metamodelio valdymo modulis atlieka ontologijos metaduomenų modelio užpildymą. *OWL* ontologijos transformavimo į RDB valdymo modulis nagrinėja ontologijos objektus, suformuoja RDB modelį, formuoja SQL užklausas. Duomenų paslaugų modelis vykdo DDL komandas, sinchronizuoja ontologijos metaduomenų struktūrą RDB, užtikrina sąsają tarp sistemos ir duomenų bazių serverio, pateikia transformavimo rezultatus.

Tam, kad paprasčiau atlikti veiksmus su ontologijos objektais transformavimo moduliai naudoja *Protégé OWL API* bibliotekas. Sistema naudoja Microsoft *JDBC* tvarkyklę, kuri leidžia operuoti su *Microsoft SQL* serveriu standartiniu būdu, naudojant *DDL* komandas. Transformavimo įrankis yra atskirtas nuo *MS JDBC* modulio, kuris veikia nepriklausomai. Naudojant vartotojo sąsają galima pasirinkti norimą *JDBC* tvarkyklę, priklausomai nuo naudojamo duomenų bazių serverio, kuriame ruošiamasi išsaugoti sukurtą duomenų bazę. Eksperimentiniam tyrimui pasirinkta viena iš plačiausiai naudojamų komercinių duomenų bazių valdymo sistemų *MS SQL Server 2008*. Vidinė supaprastinta transformavimo sistemos architektūra pateikta (46pav).



46 pav. Transformavimo sistemos pagrindiniai architektūriniai komponentai

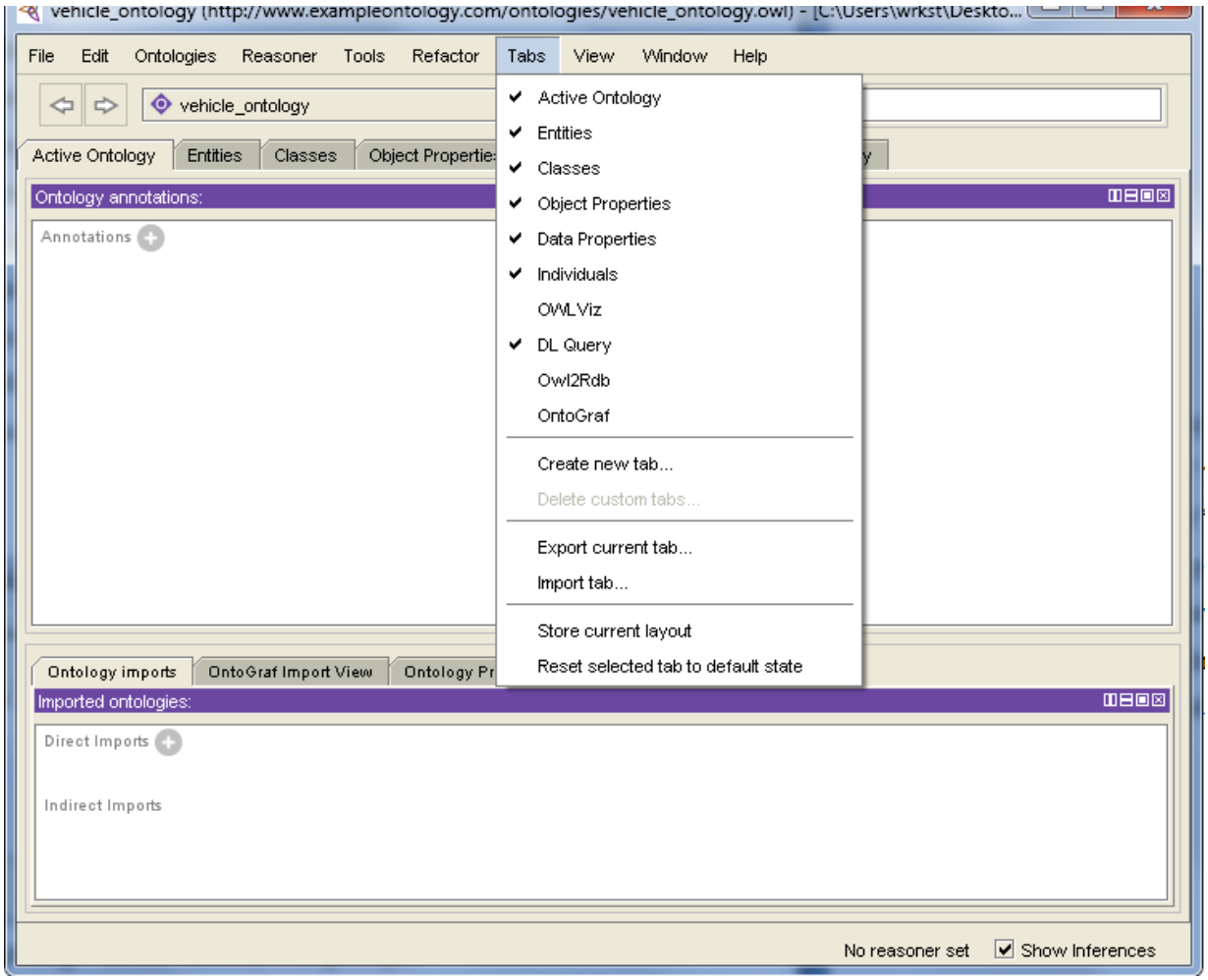
Pradinis transformavimo rezultatas yra saugomas objektinėje klasių struktūroje (47pav). Užpildyta objektais struktūra toliau naudojama *DDL* generavimui bei transformavimo suvestinei formuoti. Objektinė struktūra paverčiama į *XML* dokumentą (Lentelė 39 pateikta 2 priede), inicijuojamas *XSLT* (angl. *Extensible Stylesheet Language Transformations*) procesoriaus seansas, kurio metu naudojant *XSL* (angl. *Extensible Stylesheet Language*) transformacijų instrukcijų failą (Lentelė 40 pateikta 2 priede) gaunama *HTML* (angl. *HyperText Markup Language*) formato suvestinė.



47 pav. RDB klasių modelis

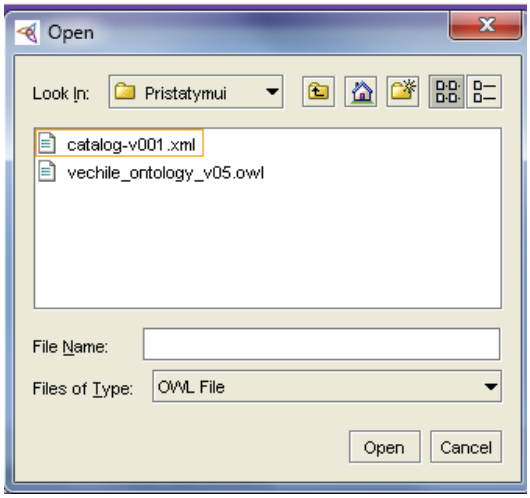
6.3. Transformavimo įrankio realizacijos aprašymas

Pasirinkta realizacijos platforma leidžia kiekvienam vartotojui nesudėtingai integruoti *OWL* transformavimo įrankį į ontologijų pasaulyje plačiausiai naudojamą „*Protege*“ sistemą. Programos bibliotekų rinkinys „*org.coode.OWL22rdb.jar*“ turi būti įkeliamas į „*Protege/plugins*“ direktoriją. Paleidus „*Protege*“ sistemą, pagrindiniame meniu pasirenkamas meniu punktas „*Tabs*“. Meniu pateikiamas visų sistemos matomų ir nematomų kortelės tipo („*tab widget*“) modulių, sąrašas (48pav). Skirtingi moduliai parenkami priklausomai nuo to, su kokio tipo ontologijomis ar jų sritimis dirbama. Sukurtas *OWL* ontologijų transformavimo į reliacines duomenų bazes modulis pavadintas „*OWL2Rdb*“. Jį pasirinkus, sistemos aktyvių kortelių juostoje susikuria nauja kortelė, kuri ir bus naudojama atlikti transformacijas.



48 pav. *Protégé* kortelių aktyvavimas

Bendras darbo su sistema procesas prasideda ontologijos *OWL* aprašo įkėlimu į sistemą. Tam atlikti pagrindiniame meniu pasirenkame punktą „*File->Open*“. Atsivėrusiame failų paieškos lange susirandame ontologijos aprašo failą *.OWL* ir spaudžiame mygtuką „*Open*“ (49pav). Ontologijos modelio informacija užkraunama į operatyviąją atmintį. Šiuos veiksmus atlieka „*Protégé*“ sistemos standartiniai moduliai. Kartu patikrinama ir *OWL* sintaksės bei failo struktūros korektiškumas, taigi dirbama tik su sintaksiškai tvarkingu ontologijos aprašu.

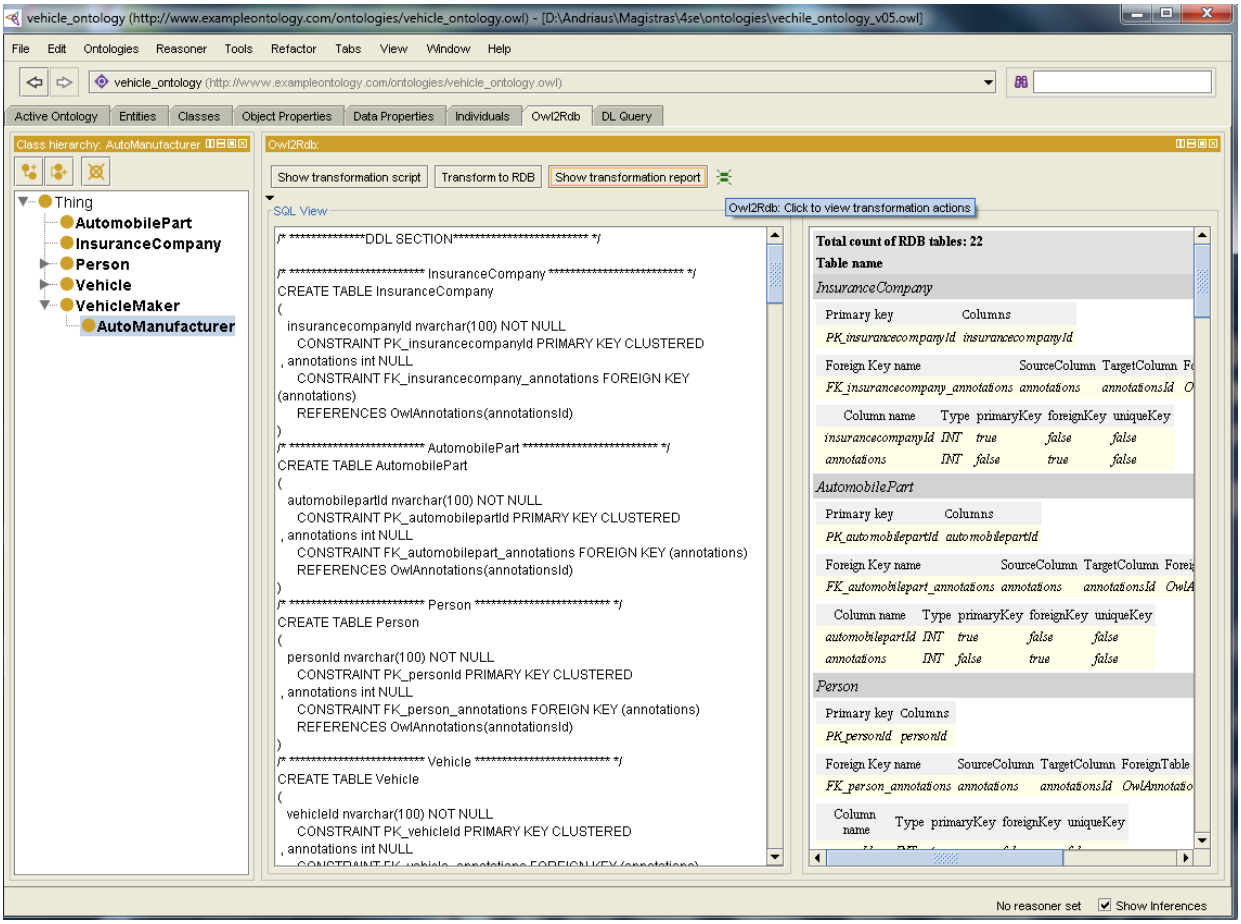


49 pav. *OWL* ontologijos aprašo pasirinkimas

Sistemoje įkeltos ontologijos turinį galima peržiūrėti ir redaguoti naudojant „*Protege.OWL*“ modulius. Tai „*Classes*“ modulis, skirtas darbui su klasėmis, „*Object Properties*“ ir „*Individuals*“ moduliai, skirti savybių ir klasių egzempliorių peržiūrai, redagavimui arba kūrimui. Taip pat yra ir kitų modulių, skirtų darbui su ontologijomis, tačiau, kadangi jie visi neįeina į šio darbo apimtį, smulkiau jų nedetalizuosime.

Pagrindinis transformavimo į RDB įrankio langas pateikiamas (50pav). Jis susideda iš:

- DDL komandų peržiūros lango – išvedamas sugeneruotas *DDL* skriptas;
- Išvesties lango – išvedami klaidų, vykdymo pranešimai;
- Transformavimo suvestinės lango – atvaizduojama transformavimo suvestinė;
- Klasių hierarchijos lango – standartinis „*Classes*“ kortelės modulis.



50 pav. Transformavimo sistemos pagrindinis langas

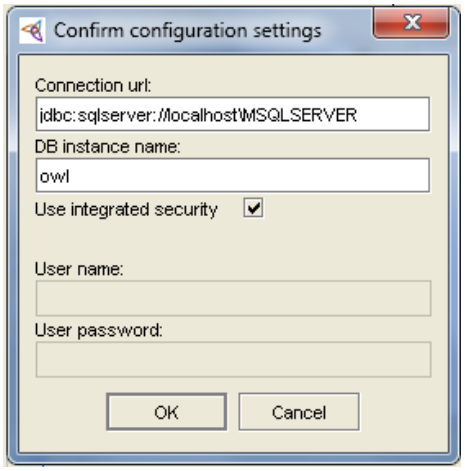
Poklasių naršyklės srityje pateikiama visos ontologijos modelio klasės. Klasės yra išdėstytos hierarchiškai, pagal tėvo-vaiko ryšius, kiekvieną hierarchinį lygmenį galima išskleisti arba sutraukti, paliekant tik tėvinę klasę.

Kitame sistemos lange „*OWL2Rdb*“ pateikiamos transformavimo komandos. Pirma komanda „*Show transformation script*“ atlieka *OWL* aprašo transformaciją į *SQL* kodą, ir jį pateikia lange sistemos lange (50pav). „*Transform to RDB*“ atidaro langą, skirtą atlikti transformacijai tiesiogiai į reliacinių duomenų bazių serverį, sukuriant duomenų bazę.

Paskutinioji komanda „*Show transformation report*“ atlieka transformacijos modelio analizę bei pateikia suvestinę.

Po komandų sąrašo seka transformavimo DDL komandų peržiūros langas. Transformuojama pati klasė, jos turimos savybės, apribojimai bei klasės egzemplioriai.

Norint tiesiogiai prisijungti prie duomenų bazių serverio ir atliktą transformaciją išsaugoti kaip reliacinę duomenų bazę, reikia pasirinkti komandą „*Transform to RDB*“, kuri atveria prisijungimo prie duomenų bazių serverio parametrų langą (51pav). Ontologijos *DDL* transformacijoms išsaugoti galima naudoti bet kokią duomenų bazių valdymo sistemą, kuri yra pasiekiama naudojant *JDBC* tvarkyklę.

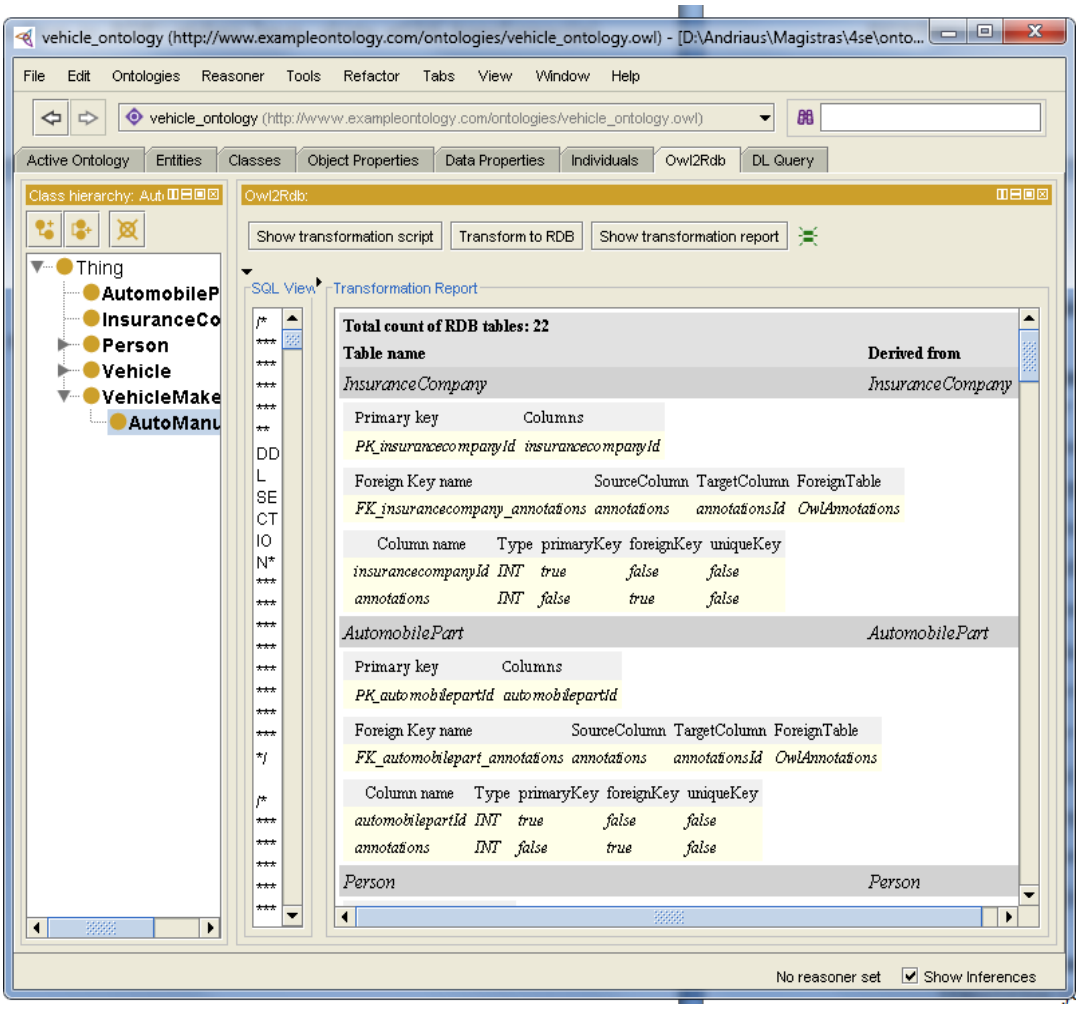


51 pav. Prisijungimo prie duomenų bazės parametrų nurodymas

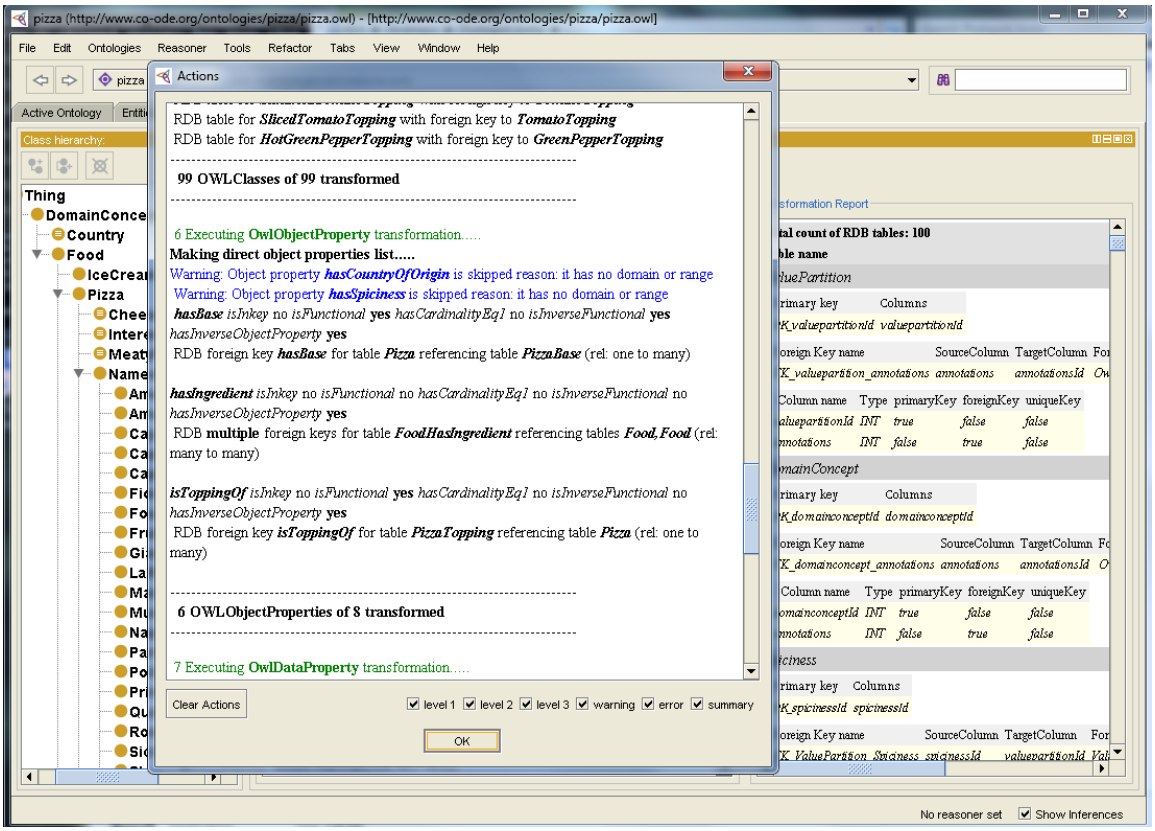
Jungiantis prie duomenų bazių serverio, reikia nurodyti *JDBC* tvarkyklę, kelią iki serverio ir vartotojo vardą bei slaptažodį. Eksperimento metu buvo naudojama „*MS SQL Server 2008*“ duomenų bazių valdymo sistema prie kurios jungiantis galima naudoti ir integruotą autentifikacijos mechanizmą, jei ši savybė yra sukonfigūruota DBVS. Norint naudoti integruotą prisijungimą reikia pažymėti varnelę „*Use integrated security*“.

Testuojant transformavimo įrankį, buvo transformuotas išsamus ontologijos pavyzdys (transporto priemonių ontologija), apimantis visas *OWL* 2 konstrukcijas. Kadangi ši ontologija atitiko transformavimo keliamus reikalavimus, visa jos informacija buvo išsaugota RDB. Atliekant eksperimentus su kitomis ontologijomis (pvz., picų, vynų, *FOAF* ir kt.) paaiškėjo, kad daugelis praktikoje sutinkamų ontologijų šių reikalavimų neatitinka, todėl transformavimo įrankis buvo papildytas sukuriant transformavimo eigos stebėjimo galimybes (53pav. bei priedas 4). Tuomet vartotojas gali sužinoti, kokios ontologijos konstrukcijos buvo netransformuotos ir priimti sprendimą jas koreguoti ar ieškoti kito saugojimo būdo.

Įvykdžius komandą „*Show transformation report*“ ekrane pateikiama transformavimo suvestinė (52pav.). Pilna suvestinė pateikta (3 priede)



52 pav. Transformavimo suvestinė



53 pav. Transformavimo protokolas

7. Eksperimentinis tyrimas ir darbo rezultatų įvertinimas

7.1. Eksperimento tikslas bei planas

Eksperimento tikslas: Įvertinti *OWL2ToRDB* algoritmo efektyvumą informacijos praradimo transformavimo metu atžvilgiu. Eksperimentas bus atliekamas vieno žmogaus, laboratorinėmis sąlygomis, naudojant *Protégé* ontologijų modeliavimo įrankį kartu su įdiegtu *OWL2ToRDB* įskiepiu. Tyrimas vykdomas dviem etapais:

- Pavyzdinės ontologijos, kuri atitinka visus algoritmo keliamus reikalavimui transformavimas;
- Praktikoje sutinkamų ontologijų transformavimo tyrimas;

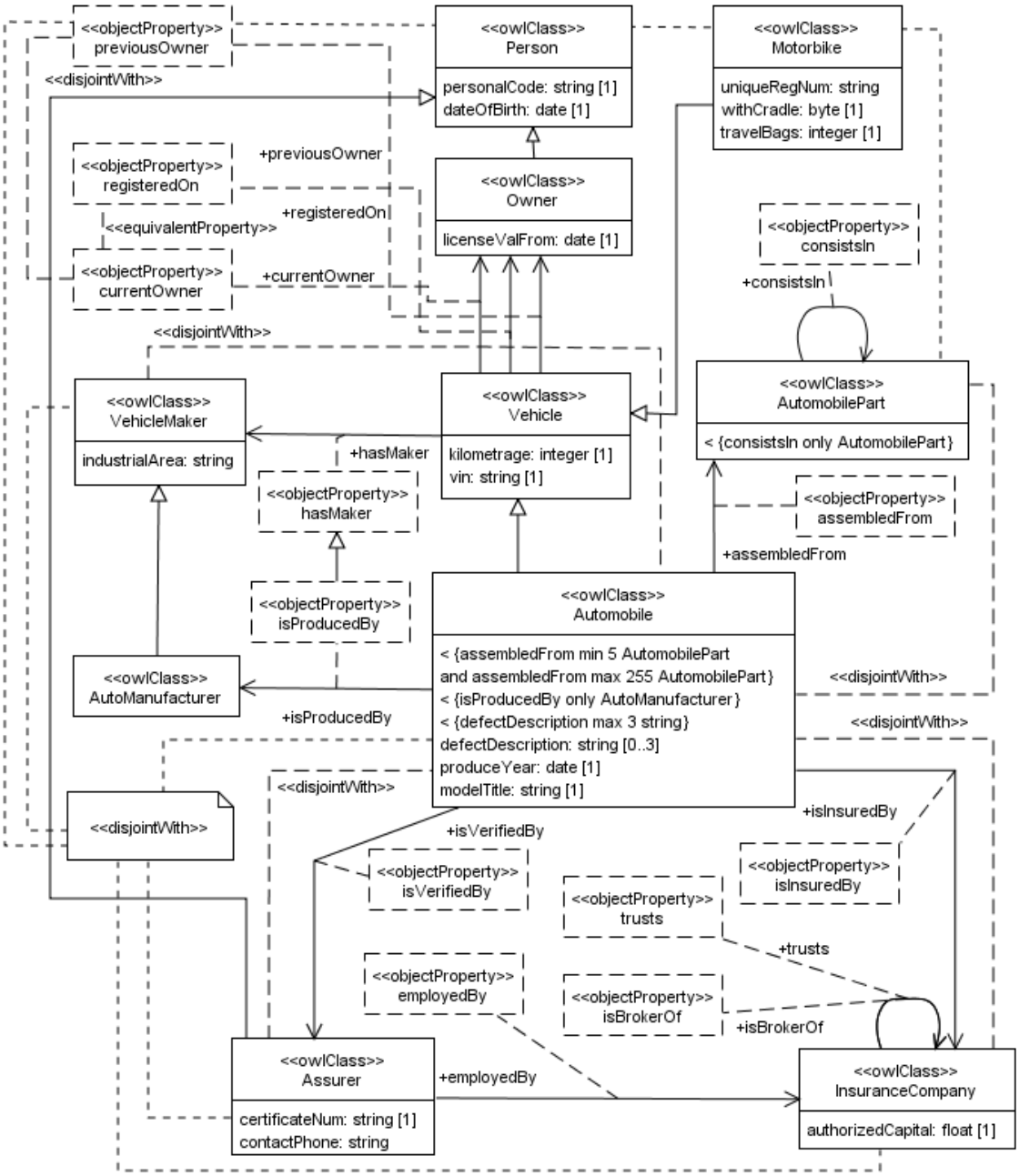
Eksperimento metu siekiama paneigti hipotezę, jog transformavimo metu yra prarandama informacija arba kitaip, transformuotų elementų skaičius nelygus ontologijos elementų skaičiui. Alternatyvi hipotezė, kurią siekiama patvirtinti – informacija yra neprarandama arba transformuotų elementų skaičius yra lygus ontologijos elementų skaičiui. Formaliai eksperimento hipotezės gali būti aprašomos:

- **H0**Elv<>Elt;
- **H1**(Ha): Elv= Elt.

Elv – ontologijos elementų skaičius viso, Elt – transformuotų ontologijos elementų skaičius. Nepriklausomi kintamieji -transformuojamos skirtingos ontologijos, priklausomas kintamasis -ontologijos elementų skaičius.

7.2. Pavyzdinės ontologijos tyrimas

Transformavimo įrankio realizacijos testavimui pasirinkta *OWL* 2 ontologija „*vehicle*“ pavaizduota (54pav). Ontologija sukurta naudojant *Protégé* įrankį. Vizualus modelis gautas naudojant *OWL2UML* plėtinį. Pilnas „*vehicle*“ ontologijos aprašas pateiktas 5 priede.



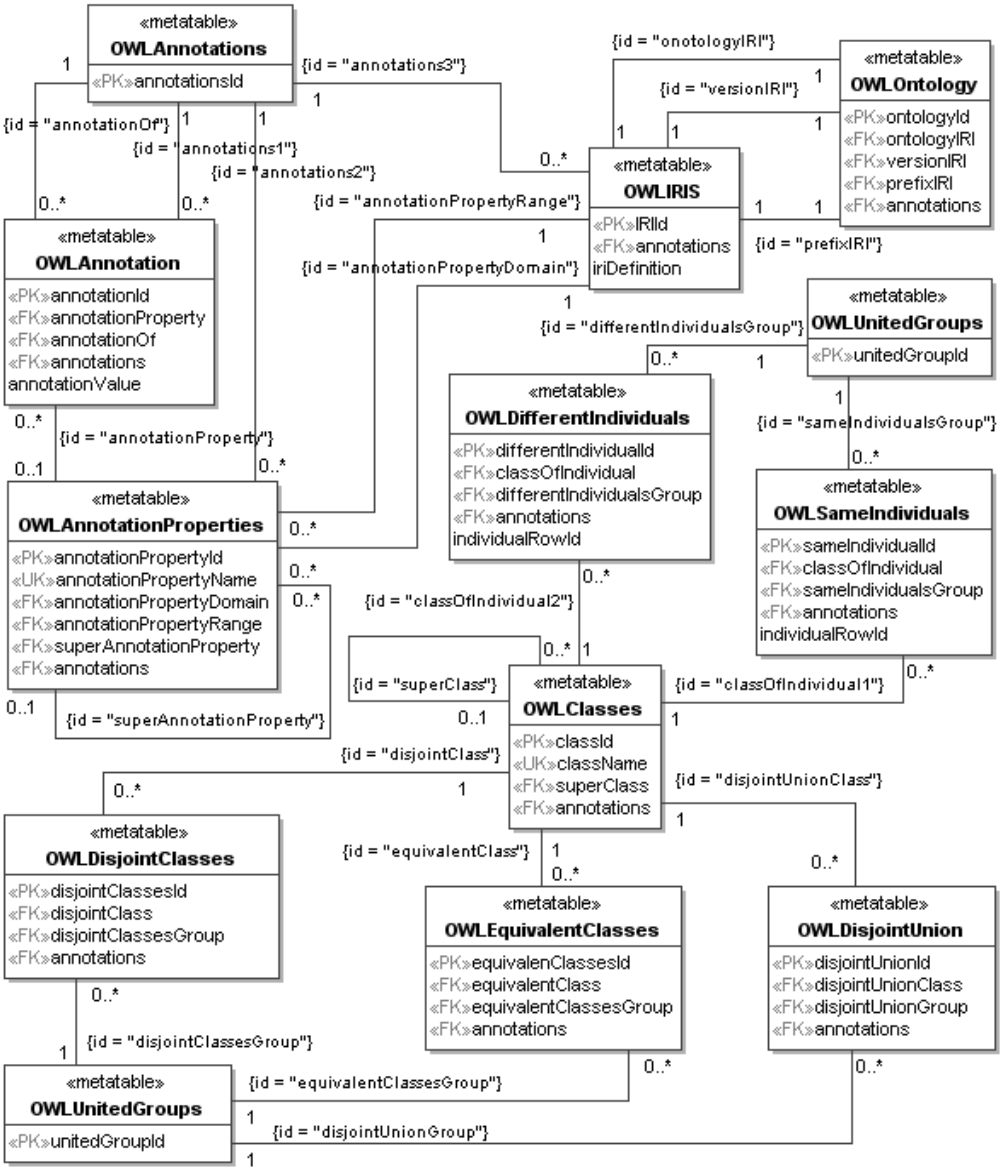
54 pav. Vehicle ontologijos grafinis modelis

7.2.1. Žingsnis 1 – ontologijos *OWL* 2 metaschemos sukūrimas

Pirmasis *OWL* 2 ontologijos transformavimo į reliacinę duomenų bazę transformavimo žingsnis yra RDB metaschemos ontologijai sukūrimas. Ši metaschema yra nekintančios struktūros ir nepriklauso nuo ontologijos sudėtingumo.

OWL 2 ontologijos RDB metaschema, sudaryta iš klasių, klasių aksiomų, *IRIs*, anotacijų ir kit. pavaizduota (55 pav.). Metashemą sudaro 12 automatiškai generuojamų meta lentelių:

OWLOntology, OWLClasses, OWLDisjointClasses, OWLDisjointUnion, OWLEquivalentClasses, OWLUnitedGroups, OWLIRIS, OWLDifferentIndividuals OWLSameIndividuals, OWLAnnotation, OWLAnnotationProperties, OWLAnnotations.



55 pav. *OWL* 2 ontologijos RDB metaschema (I)

Lentelė 8*OWLOntology* metalentelės eilutės

ontologyId	ontologyIRI	versionIRI	prefixIRI	annotations
1	1	2	3	1

Lentelė 9*OWLIRIS* metalentelės eilutės

IRIid	annotations	iriDefinition
1	*Null*	http://www.exampleontology.com/ontologies/vehicle_ontology.*OWL*
2	*Null*	http://www.exampleontology.com/ontologies/vehicle_ontology.*OWL*
3	*Null*	http://www.exampleontology.com/ontologies/vehicle_ontology.*OWL#*

Lentelė 10*OWLClasses* metalentelės eilutės

classId	className	superClass	annotations
1	Vehicle	*Null*	*Null*
2	Automobile	1	*Null*
3	Motorbike	1	*Null*
4	VehicleMaker	*Null*	*Null*
5	AutoManufacturer	4	2
6	Person	*Null*	*Null*
7	Assurer	6	3
8	Owner	6	4
9	AutomobilePart	*Null*	*Null*
10	InsuranceCompany	*Null*	*Null*

Lentelė 11*OWLDisjointClasses* metalentelės eilutės

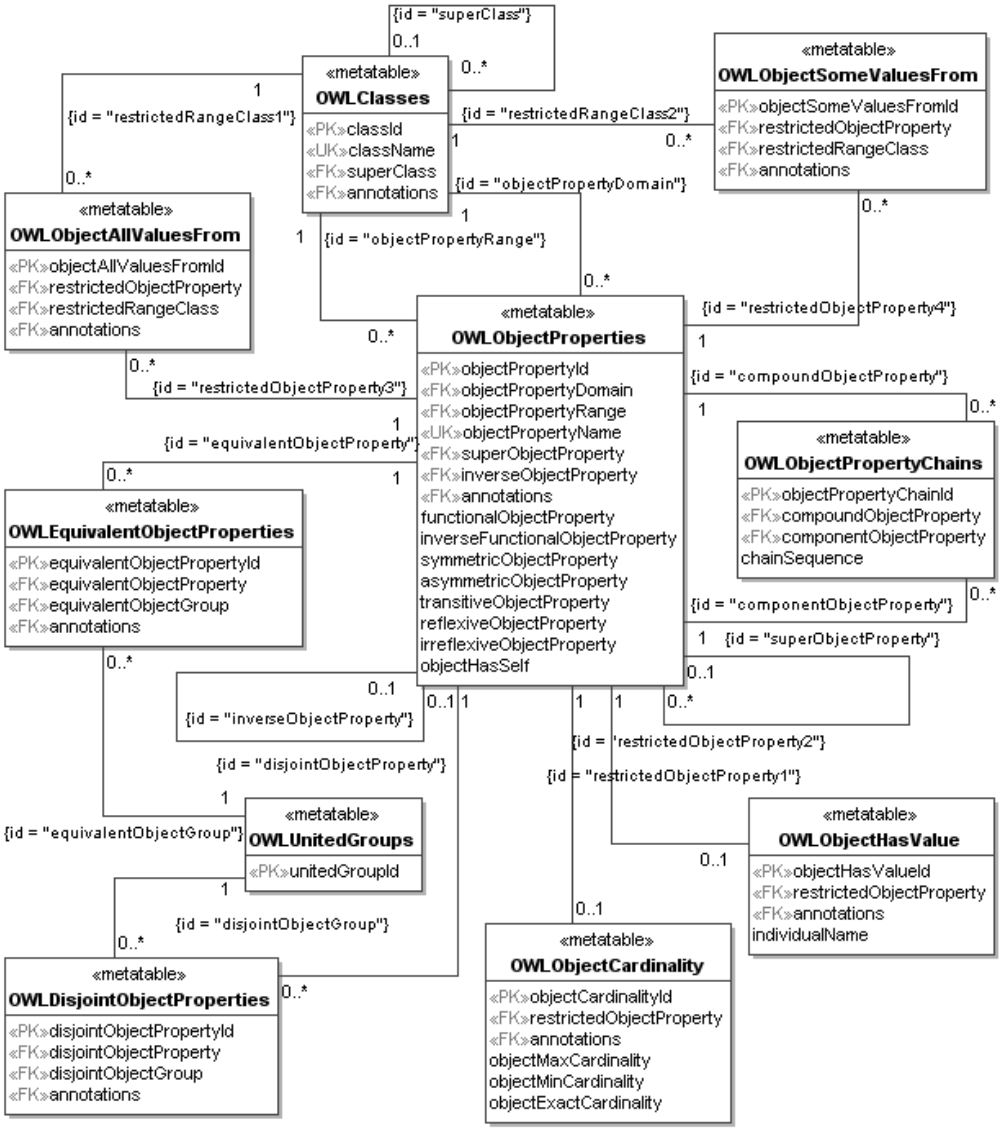
disjointClassesId	disjointClass	disjointClassesGroup	annotations
1	1	1	*Null*
2	4	1	*Null*
3	9	1	*Null*
4	6	1	*Null*
5	10	1	*Null*
6	2	2	*Null*
7	9	2	*Null*
8	3	2	*Null*
9	4	2	*Null*

Lentelė 12*OWLAnnotation*metalentelės eilutės

Annotation Id	annotation Property	annotation Of	annotations	annotationValue
1	*Null*	1	*Null*	*OWL* vehicle ontology
2	*Null*	2	*Null*	Automobile manufacturing company
3	*Null*	3	*Null*	An assurer is a person who insures
4	*Null*	4	*Null*	A person who owns a vehicle
5	*Null*	5	*Null*	A person who currently owns a vehicle
6	*Null*	6	*Null*	Vehicle identification number
7	*Null*	7	*Null*	Can not be more than three defect descriptions
8	*Null*	8	*Null*	The driving license is valid from
9	*Null*	9	*Null*	Unique number of a person
10	*Null*	10	*Null*	This vehicle is an automobile
11	*Null*	11	*Null*	This vehicle is a motorbike
12	*Null*	12	*Null*	The car has motor defect
13	*Null*	13	*Null*	The car has gearbox defect
14	*Null*	14	*Null*	The bike is with a cradle and two bags
15	*Null*	15	*Null*	Motorbikes manufacturing company
16	*Null*	16	*Null*	Japanese manufacturer
17	*Null*	17	*Null*	Korean manufacturer
18	*Null*	18	*Null*	Hyundai Motor Company
19	*Null*	19	*Null*	Mitsubishi Motors
20	*Null*	20	*Null*	This person is an assurer
21	*Null*	21	*Null*	Driving license more than 10 years
22	*Null*	22	*Null*	Works in “ZurichConnect” company
23	*Null*	23	*Null*	Works in “BrokerKing” company

Annotation Id	annotation Property	annotation Of	annotations	annotationValue
24	*Null*	24	*Null*	Is on top of the cylinder block
25	*Null*	25	*Null*	A device in the ignition system
26	*Null*	26	*Null*	Transmission system
27	*Null*	27	*Null*	Swiss insurance company
28	*Null*	28	*Null*	Is a broker of "ZurichConnect" company
29	*Null*	29	*Null*	Cheap car insurance company

OWL 2 ontologijos RDB metaschema, sudaryta iš objektinių savybių, objektinių savybių aksiomų, apribojimų pavaizduota (56pav). Metaschemą sudaro 8 automatiškai generuojamų meta lentelių:*OWLObjectProperties, OWLObjectCardinality, OWLObjectPropertyChains, OWLObjectAllValuesFrom, OWLObjectSomeValuesFrom, OWLObjectHasValue, OWLEquivalentObjectProperties, OWLDisjointObjectProperties.*



56 pav. *OWL* 2 ontologijos RDB metaschema (II)

Lentelė 13*OWLObjectProperties* metalentelės eilutės

OPID	OPD	OPR	OPName	SOP	ANN	FOP	ROP	IROP
1	2	9	assembledFrom	*Null*	*Null*	*Null*	*Null*	*Null*
2	9	9	consistsIn	*Null*	*Null*	True	*Null*	*Null*
3	1	8	currentOwner	*Null*	5	True	*Null*	*Null*
4	7	10	employedBy	*Null*	*Null*	True	*Null*	*Null*
5	1	4	hasMaker	*Null*	*Null*	True	*Null*	*Null*
6	2	5	isProducedBy	5	*Null*	True	*Null*	*Null*
7	10	10	isBrokerOf	*Null*	*Null*	True	*Null*	True
8	2	10	isInsuredBy	*Null*	*Null*	True	*Null*	*Null*
9	2	7	isVerifiedBy	*Null*	*Null*	True	*Null*	*Null*
10	1	8	previousOwner	*Null*	*Null*	*Null*	*Null*	*Null*
11	1	8	registeredOn	*Null*	*Null*	True	*Null*	*Null*
12	10	10	trusts	*Null*	*Null*	*Null*	True	*Null*

Lentelė 14*OWLObjectPropertyChain* metalentelės eilutės

objectPropertyChainId	compoundObjectProperty	componentObjectProperty	chainSequence
1	8	9	1
2	8	4	2

Lentelė 15*OWLEquivalentObjectProperties* metalentelės eilutės

equivalentObjectPropertyId	equivalentObjectProperty	equivalentObjectGroup	annotations
1	3	3	*Null*
2	11	3	*Null*

Lentelė 16*OWLDisjointObjectProperties* metalentelės eilutės

disjointObjectPropertyId	disjointObjectProperty	disjointObjectGroup	annotations
1	3	4	*Null*
2	10	4	*Null*

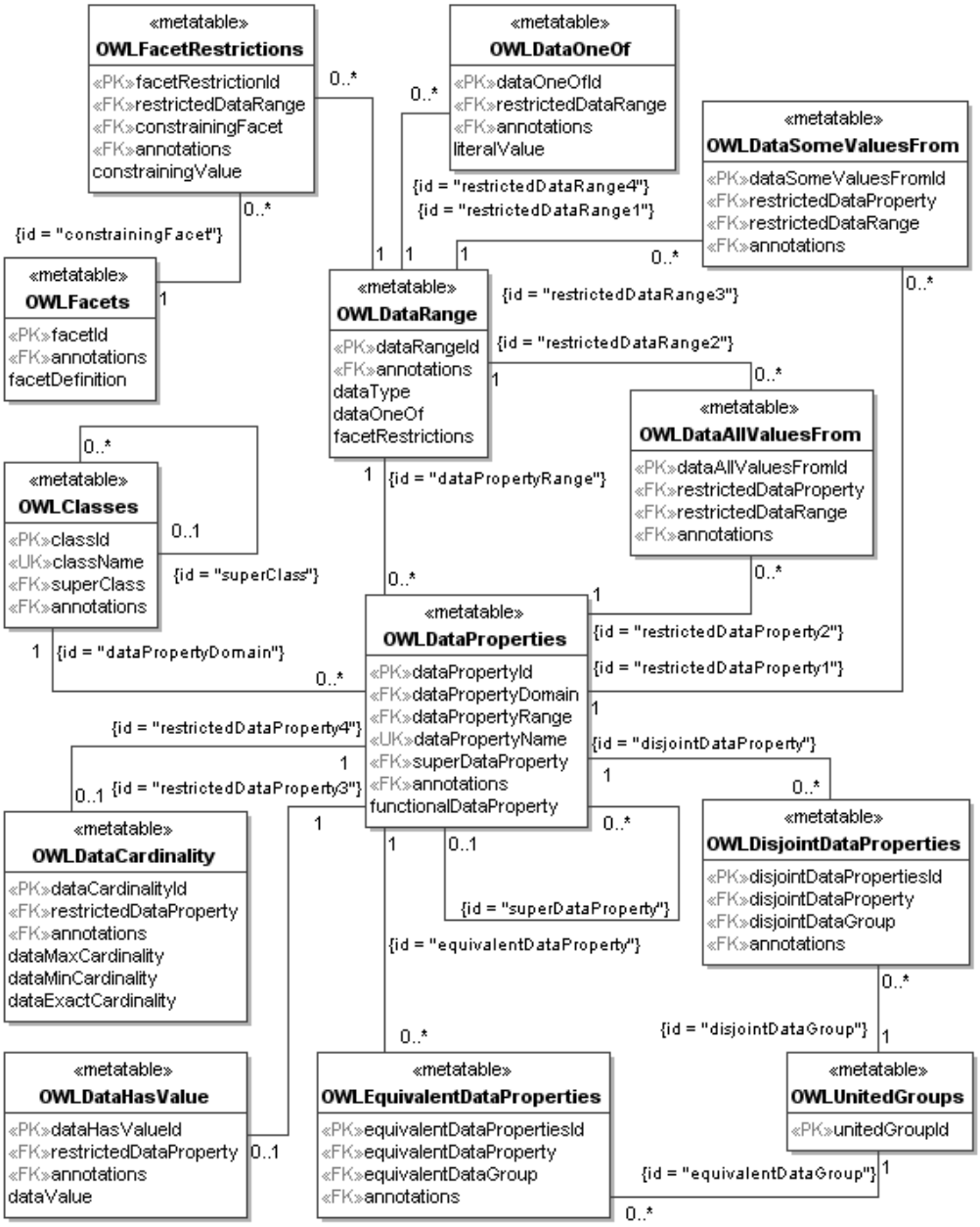
Lentelė 17*OWLObjectAllValuesFrom* metalentelės eilutės

objectAllValuesFromId	restrictedObjectProperty	restrictedRangeClass	annotations
1	6	5	*Null*
2	2	9	*Null*

Lentelė 18*OWLObjectCardinality* metalentelės eilutės

objectCardinalityId	restrictedObjectProperty	annotations	objectMax	objectMin	objectExact
1	1	*Null*	255	5	*Null*

OWL 2 ontologijos RDB metaschema, sudaryta iš duomenų savybių, duomenų savybių aksiomų, apribojimų pavaizduota (57pav). Metashemą sudaro 11 automatiškai generuojamų meta lentelių: *OWLDataProperties, OWLDisjointDataProperties, OWLEquivalentDataProperties, OWLDataHasValue, OWLDataCardinality, OWLDataAllValuesFrom, OWLDataOneOf, OWLDataRange, OWLDataSomeValuesFrom, OWLFacets, OWLFacetRestrictions.*



57 pav. *OWL* 2 ontologijos RDB metaschema (III)

Lentelė 19*OWLDataProperties* metalentelės eilutės

dataPId	dataPDomain	dataPRange	dataPName	superDataP	annotations	FDP
1	10	3	authorizedCapital	*Null*	*Null*	True
2	7	1	certificateNum	*Null*	*Null*	True
3	1	1	vin	*Null*	6	True
4	3	1	uniqueRegNum	4	*Null*	True
5	7	1	contactPhone	*Null*	*Null*	*Null*
6	6	4	dateOfBirth	*Null*	*Null*	True
7	2	1	defectDescription	*Null*	7	*Null*
8	4	1	industrialArea	*Null*	*Null*	*Null*
9	1	2	kilometrage	*Null*	*Null*	True
10	8	4	licenseValFrom	*Null*	8	True
11	2	1	modelTitle	*Null*	*Null*	True
12	6	1	personalCode	*Null*	9	True
13	2	4	produceYear	*Null*	*Null*	True
14	3	2	travelBags	*Null*	*Null*	True
15	3	5	withCradle	*Null*	*Null*	True

Lentelė 20*OWLDataRange* metalentelės eilutės

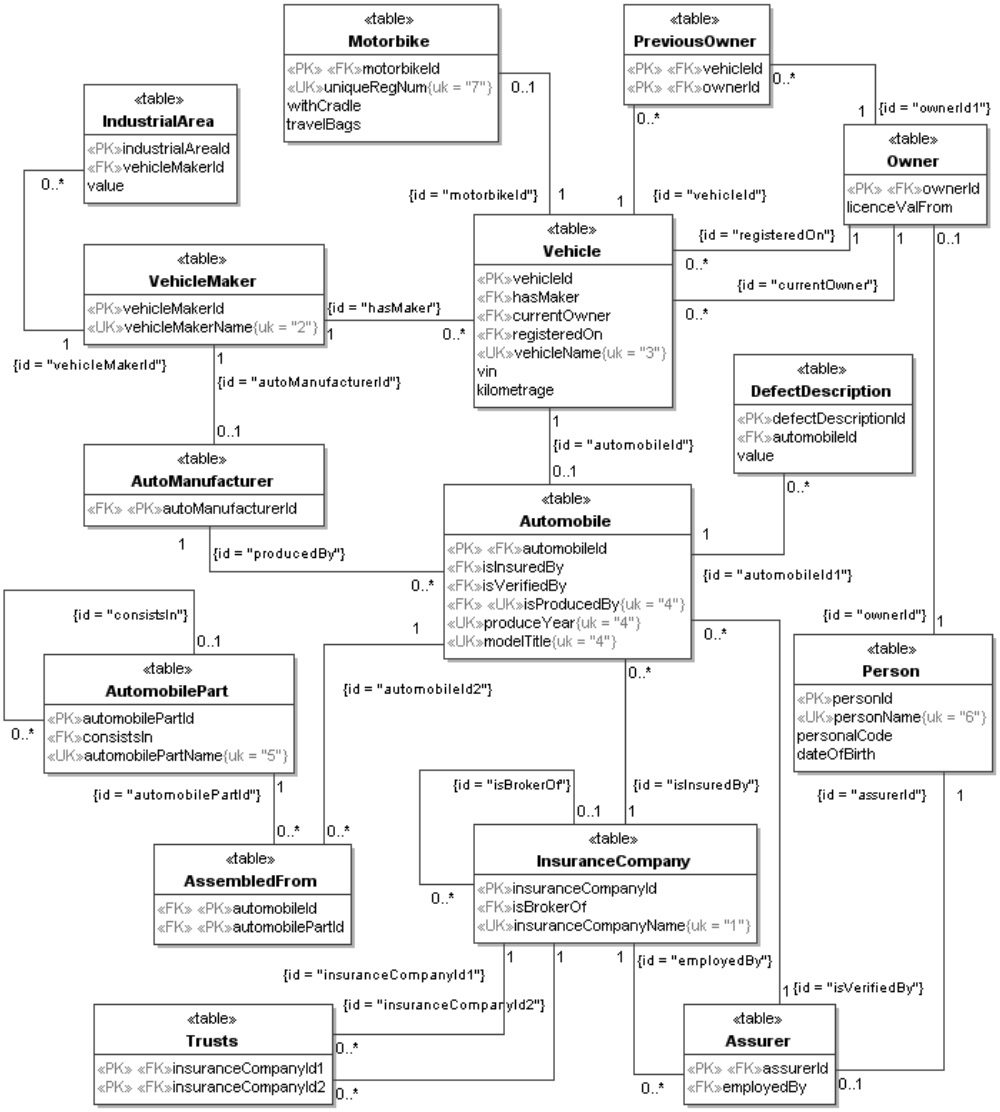
dataRangeId	annotations	dataType	dataOneOf	facetRestrictions
1	*Null*	xsd:string	*Null*	*Null*
2	*Null*	xsd:integer	*Null*	*Null*
3	*Null*	xsd:float	*Null*	*Null*
4	*Null*	xsd:date	*Null*	*Null*
5	*Null*	xsd:byte	*Null*	*Null*

Lentelė 21*OWLDataCardinality* metalentelės eilutės

dataCardinalityId	restrictedDataProperty	annotations	dataMax	dataMin	dataExact
1	7	*Null*	3	*Null*	*Null*

7.2.2. Žingsnis 2 – ontologijos *OWL* 2 konceptų transformavimas į RDB schemą

Po ontologijos metaduomenų išsaugojimo, *OWL* 2 klasės, objektinės bei duomenų savybės yra transformuojamos į reliacinės duomenų bazės schemą, kuri bus naudojama individų bei jų savybių išsaugojimui. Reliacinė schema gauta šio eksperimento metu pavaizduota (58pav).



58 pav. Vehicle ontologijos RDB schema

7.2.3. Žingsnis 3 – RDB schemos užpildymas individais bei jų savybėmis

Paskutinio *OWL* 2 ontologijos transformavimo į reliacinę duomenų bazę žingsnio metu į duomenų bazę yra įrašomi klasių egzemplioriai bei užpildomi objektines bei duomenų savybes atitinkantys lentelių stulpeliai.

Lentelė 22*Vehicle* lentelės eilutės

vehicleId	hasMaker	currentOwner	registeredOn	vehicleName	vin	kilometrage	annotations
1	3	3	3	AutoEBB887	4T1BE46K17U704216	145846	10
2	4	4	4	AutoDOT475	KMHCN46C07U458487	126482	*Null*
3	6	4	4	AutoCKD843	JA32U8FW3BU028408	804965	*Null*
4	5	5	5	AutoGKT274	1N4BL21E67C170979	240473	*Null*
5	3	5	5	AutoRKU418	1NXBU4EE6AZ379083	114726	*Null*
6	2	*Null*	*Null*	Moto568AE	JKAEXMF166DA23239	22473	11
7	2	*Null*	*Null*	Moto968UH	JKBZXNC15AA020752	27912	*Null*
8	1	*Null*	*Null*	Moto675SR	JYAVP11E48A106468	46471	*Null*

Lentelė 23*Automobile* lentelės eilutės

automobileId	insuredBy	isVerifiedBy	isProducedBy	produceYear	modelTitle	annotations
1	1	1	3	2006	Avensis	*Null*
2	2	1	4	2007	Sonata	12
3	3	2	6	2008	Lancer	13
4	4	1	5	2006	Primera	*Null*
5	5	2	3	2006	Corolla	*Null*

Lentelė 24*Motorbike* lentelės eilutės

motorbikeId	uniqueRegNum	withCradle	travelBags	annotations
6	JKAEXMF166DA23239	0	0	*Null*
7	JKBZXNC15AA020752	0	0	*Null*
8	JYAVP11E48A106468	1	2	14

Lentelė 25*Person* lentelės eilutės

personId	personName	personalCode	dateOfBirth	annotations
1	JohnMayer	37405087548	*Null*	20
2	PeterIrving	38304232165	*Null*	*Null*
3	AdamBradley	38104110635	1981-04-11	*Null*
4	EricBrown	37201144521	1972-01-14	21
5	RobertStewart	38610238745	1986-10-23	*Null*

Lentelė 26*Owner* lentelės eilutės

ownerId	licenseValFrom	annotations
3	1999-05-12	*Null*
4	1991-02-24	*Null*
5	2004-11-15	*Null*

Lentelė 27*Assurer* lentelės eilutės

assurerId	employedBy	annotations
1	1	22
2	3	23

Lentelė 28*PreviousOwner* lentelės eilutės

vehicleId	ownerId
1	4
3	5

Lentelė 29*VehicleMaker* lentelės eilutės

vehicleMakerId	vehicleMakerName	annotations
1	Yamaha	*Null*
2	Kawasaki	15
3	Toyota	16
4	Hyundai	17
5	Nissan	*Null*

6	Mitsubishi	*Null*

Lentelė 30*AutoManufacturer* lentelės eilutės

autoManufacturerId	annotations
3	*Null*
4	18
5	*Null*
6	19

Lentelė 31*IndustrialArea* lentelės eilutės

industrialAreaId	vehicleMakerId	value
1	1	Motorbikes
2	2	Motorbikes
3	3	Automobiles
4	4	Automobiles
5	5	Automobiles
6	6	Automobiles

Lentelė 32*InsuranceCompany* lentelės eilutės

insuranceCompanyId	isBrokerOf	insuranceCompanyName	annotations
1	*Null*	ZurichConnect	27
2	*Null*	MotorDirect	*Null*
3	1	BrokerKing	28
4	*Null*	AutoTrader	*Null*
5	*Null*	DivaInsurance	29

Lentelė 33T*rusts* lentelės eilutės

insuranceCompanyId1	insuranceCompanyId2
1	3
5	4

Lentelė 34*AutomobilePart* lentelės eilutės

automobilePartId	consistsIn	automobilePartName	annotations
1	*Null*	SteeringWheel	*Null*
2	*Null*	Engine	*Null*
3	2	CylinderHead	24
4	2	Distributor	25
5	*Null*	Gearbox	26

Lentelė 35*AssembledFrom* lentelės eilutės

automobileId	automobilePartId
5	1
5	2
5	3
5	4
5	5

Lentelė 36*DefectDescription* lentelės eilutės

defectDescriptionId	automobileId	value
1	2	Motor defect
2	3	Gearbox defect

Lentelė 37*ContactPhone* lentelės eilutės

contactPhoneId	assurerId	value
1	1	+37061256536
2	2	+37061198696

7.2.4. Žingsnis 4 – Transformavimo rezultato ataskaita

Užpildyta objektais struktūra toliau naudojama *DDL* generavimui bei transformavimo suvestinei formuoti. Objektinė struktūra paverčiama į *XML* dokumentą, inicijuojamas *XSLT* procesoriaus seansas, kurio metu naudojant *XSL* transformacijų instrukcijų failą (2 priedas Lentelė 40) gaunama *HTML* formato suvestinė, kuri pateikta (3 priede).

7.3. Praktikoje sutinkamų ontologijų tyrimas

Tyrimui atsitiktinai pasiriktos žiniatinklyje publikuojamos ontologijos: *Pizza, Wine, Proton ontology, University benchmark, Word Net* ir kit.Remiantis transformavimo įrankio pateikiamų rezultatų apie transformuotus ontologijos elementus analizės rezultatais hipotezės **H0:**Elv<>Elt paneigti negalima. Tačiau galima išvada, jog pagrindinės informacijos praradimo priežastys yra elementui nenurodyta apibrėžimo ir kitimo sritis arba *OWLThing* naudojimas jas aprašant. Dauguma ontologijų neatitinka *OWL2ToRDB* metodo keliamų reikalavimų. Dėl netransformuojamų atvirkštinių objektinių savybių informacija nėra prarandama kadangi visa ontologijos struktūra yra išsaugoma metalentelėse. Tačiau dėl šios priežasties individams nurodytos atvirkštinės objektinės savybės taip pat negali būti transformuojamos (dabartinėje transformavimo metodo realizacijoje ši problema laikinai yra neišspręsta).

Kaip jau buvo minėta, *OWL2ToRDB* transformavimo metodas yra nepritaikytas sudėtingų ontologijos aksiomų, kurios yra naudojamos išvedimui, išsaugojimui. Tačiau šios ontologijos konstrukcijos neturi reikšmės ontologijoje saugomiems duomenims. Tyrimo rezultatai pateikti lentelėje.

Lentelė 38 Transformavimo rezultatai

Ontologija	Elementas	Elementų skaičius viso	Transformuoti elementai	Procentas	Neatitikimo priežastys
Vehicle					
	Klasės	19	19	1	
	Objektinės savybės	28	28	1	
	Duomenų savybės	27	27	1	
	Individai	42	42	1	
	Individų objektinės savybės	48	48	1	
	Individų duomenų savybės	61	61	1	
	Viso	225	225	1	
Pizza					
	Klasės	99	99	1	
	Objektinės savybės	8	7	0,875	Nėra apibrėžimo arba kitimo srities
	Individai	5	5	1	
	Viso	112	111	0,991	
Wine					
	Klasės	74	74	1	
	Objektinės savybės	12	11	0,917	Apibrėžimo arba kitimo sritis yra *OWLThing*
	Duomenų savybės	1	1	1	
	Individai	161	161	1	
	Individų objektinės savybės	245	179	0,731	Atvirkštinės objektinės savybės yra netransformuojamos
	Individų duomenų savybės	1	1	1	
	Viso	494	427	0,864	
Proton ontology					
	Klasės	275	275	1	
	Objektinės savybės	89	56	0,812	Nėra apibrėžimo arba kitimo srities
	Duomenų savybės	41	38	0,927	Nėra apibrėžimo arba kitimo srities
	Viso	405	369	0,911	
University benchmark					
	Klasės	43	43	1	
	Objektinės savybės	25	19	0,76	Nėra apibrėžimo arba kitimo srities
	Duomenų savybės	7	4	0,571	Nėra apibrėžimo arba kitimo srities
	Viso	75	66	0,880	
Word Net					
	Klasės	14	14	1	
	Objektinės savybės	34	17	0,500	Nėra apibrėžimo arba kitimo srities
	Duomenų savybės	5	3	0,600	Nėra apibrėžimo arba kitimo srities
	Viso	53	34	0,642	
Generations					
	Klasės	18	18	1	
	Objektinės savybės	4	4	1	
	Individai	7	7	1	
	Individų objektinės savybės	9	7	0,778	Atvirkštinės objektinės savybės yra

Ontologija	Elementas	Elementų skaičius viso	Transformuoti elementai	Procentas	Neatitikimo priežastys
					netransformuojamos
	Viso	38	36	0,947	
Dolce Lite					
	Klasės	37	37	1	
	Objektinės savybės	70	62	0,886	Atvirkštinės objektinės savybės yra netransformuojamos
	Viso	107	99	0,925	
Description and Situation					
	Klasės	95	95	1	
	Objektinės savybės	229	44	0,192	Nėra apibrėžimo arba kitimo srities
	Viso	324	139	0,429	

8. Išvados

1. Ontologijų transformavimo į reliacines duomenų bazes metodų ir įrankių analizė parodė, kad nėra tinkamo metodo, kuris leistų neprarasti ontologinės informacijos, o gauta duomenų bazės schema atitiktų dalykinės srities semantiką, būtų suprantama informacinių sistemų kūrėjams, leistų išnaudoti RDB privalumus ir būtų pritaikoma ne tik ontologijoms apdoroti, bet ir kitiems taikomiesiems uždaviniams.
2. Iš esamų transformavimo metodų tinkamiausias buvo ISK katedroje sukurtas hibridinis *OWLToRDB* metodas, tačiau jis buvo skirtas pirmai *OWL* versijai ir neapėmė visų ontologijos elementų, todėl buvo nuspręsta tobulinti šį metodą: įvesti daugiau metaschemos lentelių, papildyti esamus transformavimo algoritmus ir realizuoti juos kaip naujausios *Protégé* versijos įskiepį.
3. Testuojant sukurtą *OWL2ToRDB* transformavimo įrankį, paaiškėjo, kad jis gerai veikia su taisyklingomis ontologijomis, atitinkančiomis ontologijų kūrimo reikalavimus, papildytus reikalavimais transformavimui į RDB. Papildomų reikalavimų transformavimui į RDB yra labai nedaug, tačiau praktikoje naudojamos ontologijos dažnai šių reikalavimų neatitinka, ir tuomet informacija gali būti prarandama.
4. Eksperimentas su aštuoniomis praktikoje naudojamomis ontologijomis parodė, kad transformuojama nuo 40 iki 99 procentų jų elementų ir tik pavyzdinė ontologija transformuojama visa. Dažniausia informacijos praradimo priežastis – nenurodyta objektų savybių apibrėžimo (angl. *domain*) arba kitimo sritis (angl. *range*).
5. Informacijos nuostolius būtų galima sumažinti, papildant transformavimo algoritmą galimybe paveldėti apibrėžimo ir kitimo sritį iš bendresnių objektinių savybių. Kitais atvejais šią problemą turi spręsti ontologijos kūrėjas.
6. Hibridinį ontologijų saugojimo metodą tikslinga taikyti tada, kai esamas ar kuriamas ontologijas norima susieti su taikomosiomis programomis, naudojančiomis reliacines duomenų bazes. Tuomet tikslinga ne tik kurti taisyklingas ontologijas, atitinkančias ontologijų kūrimo reikalavimus, bet ir pritaikyti prie reikalavimų, keliamų saugojimui RDB.

9. Literatūra

1. Bechhofer, S., Horrocks, I., Turi, D. The *OWL* Instance Store: System Description. Proceedings CADE-20; Lecture Notes in Computer Scrience, 2005.
2. Cullot, N., Parent, C., Spaccapietra, S., Vangenot C. Ontologines: A contribution to the DL/DB debate.
3. Dan-Tong, O., Xian-ji, C., Yu-xin, Y. Mapping integrity constraint ontology to relational databases. The Journal of China Universities of Posts and Telecommunications, December 2010, 17(6): 113–121.
4. Geisler, S., Brauers, A., Quix, C., Schmeink, A. Ontology-based system for clinical trial data management. IEE Benelux EMBS Symposium, 2007 December 6-7.
5. Golbreich, C., Wallace, E.K., Patel-Schneider, PF, *OWL* 2 Web Ontology Language New Features and Rationale. W3C Proposed Recommendation 22 September 2009, http://www.w3.org/TR/2009/PR-*OWL*2-new-features-20090922/. Accessed 1 June 2011
6. Goodwin, R., Lee, J., Mikaila, G. A., Stanoi, I. Leveraging Relational Database Systems for Large-Scale Ontology Management. CIDR Conference, 2005.
7. 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
8. Motik, B., Patel-Schneider, P.F., Parsia, B. *OWL*2 Web Ontology Language Structural Specification and Functional-Style Syntax. W3C Proposed Recommendation 22 September 2009, available at http://www.w3.org/TR/2009/REC-*OWL*2-syntax-20091027/.
9. Motik. B., Horrocks. I., Sattler, U. Bridging the Gap Between *OWL* and Relational Databases. In: WWW 2007, International World Wide Web Conference. 2007, 807–816
10. Pan, Z., Heflin, J. DLDB: Extending Relational Databases to Support Semantic Web Queries.
11. Trinkūnas, J., Vasilecas, O. A graph oriented model for ontology transformation into conceptual data model. Information technology and control, 2007, Vol. 36, No. 1A, 126-132.
12. Trinkūnas, J., Vasilecas, O. Building Ontologies from Relational Databases Using Reverse Engineering Methods. International Conference on Computer Systems and Technologies - CompSysTech'07
13. Vysniauskas, E., Nemuraite, L. Mapping of *OWL* ontology concepts to RDB schemas. In: Information Technologies' 2009: proceedings of the 15th International Conference on Information and Software Technologies, IT 2009, Kaunas, Lithuania, April 23-24, 2009, Kaunas University of Technology, Kaunas, Technologija, 2009, 317–327.
14. Vysniauskas, E., Nemuraite, L. Transforming Ontology Representation from *OWL* to Relational Database. Information Technology and Control, Vol. 35(3A), 2006, 333–343.

1. priedas. Straipsnis

Įrankis *OWL 2* ontologijoms transformuoti į reliacines duomenų bazes

Andrius Kondratas, vadovė prof. Lina Nemuraitė

Kauno technologijos universitetas, Informacijos sistemų katedra, Studentų 50-313a. LT-51368Kaunas, Lietuva, andrius2kondratas@gmail.com, lina.nemuraite@ktu.lt

Santrauka. Populiarėjant ontologijoms ir didėjant jose saugomos informacijos apimtims, atsiranda būtinybė saugoti ontologijas reliacinėse duomenų bazėse. Šiame straipsnyje aprašomas sukurtas įrankis, kuris leidžia taisyklingas *OWL 2* ontologijas, atitinkančias keliamus reikalavimus, transformuoti į reliacinių duomenų bazių schemas neprarandant informacijos.

Raktiniai žodžiai:*OWL 2*, ontologija, reliacinės duomenų bazės, transformacija.

1. Įžanga

Ontologijos vis plačiau naudojamos įvairiuose taikymuose verslo procesų ir informacijos integravimui, paieškai ir tvarkymui. Šie taikymai reikalauja greitaveikos ir efektyvaus saugojimo dėl poreikio manipuliuoti didelių ontologijų duomenimis. Kai ontologijomis paremtos sistemos auga sudėtingumu ir apimtimi, įprasti ontologijų saugojimo failuose mechanizmai tampa nebetinkami. Tokiomis aplinkybėmis ontologijų saugojimas reliacinėse duomenų bazėse (RDB) tampa būtina sąlyga Semantinio tinklo ar įmonių informacinėse sistemose. Šiame straipsnyje pristatomas sprendimas ir įrankis, leidžiantis *OWL 2* ontologijas, atitinkančias tam tikrus reikalavimus, transformuoti į reliacinių duomenų bazių schemas neprarandant informacijos.

Straipsnio struktūra: antrame skyriuje analizuojamos ontologijų ir duomenų bazių savybės bei esami ontologijų saugojimo duomenų bazėse metodai; trečiame pateikiama loginė transformavimo įrankio architektūra bei jos realizacija. Straipsnis baigiamas išvadomis.

2. Ontologijų saugojimo reliacinėse duomenų bazėse metodų analizė

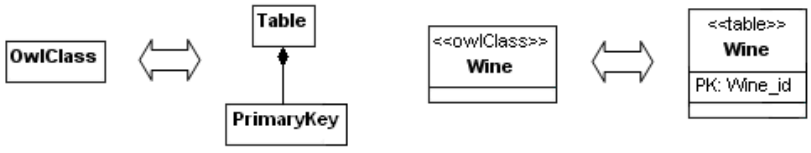
Pagrindinis skirtumas tarp ontologijų kalbos *OWL* 2 ir reliacinių duomenų bazių yra tas, kad ontologijoms taikoma „atviro pasaulio“ prielaida, t. y. jų informacija gali būti nepilna. Dalis ontologijų duomenų išvedama pagal aprašytas aksiomas, ryšiai nėra griežti. Pavyzdžiui, savybė gali būti apibrėžta kaip privaloma, bet ontologijų redaktoriai leidžia išsaugoti egzempliorius, neturinčius tos savybės reikšmių, nes priimama, kad savybė egzistuoja, tik šiuo metu yra nežinoma. Priešingai, duomenų bazės yra uždaros, t. y. jų duomenys turi būti pilni, jose užtikrinamas duomenų vientisumas. Ne visas ontologijų savybes galima tiesiogiai išsaugoti duomenų bazėje, todėl kyla problemų norint išsaugoti sudėtingesnes ontologijas. Sprendžiant iš literatūros šaltinių gausumo, ontologijų saugojimas reliacinėse duomenų bazėse yra aktyvi tyrimo sritis 1.

Esami ontologijų saugojimo RDB metodai turi trūkumų [2, 5]. Juos galima skirstyti į informaciją prarandančius metodus ir neprarandančius. Informaciją prarandantys metodai išsaugo tik ontologijų egzempliorius, o geriausiu atveju tik klases ir savybes. Informacijos neprarandančius metodus galima skirstyti į dvi dalis: metodai, naudojantys primityvią schemą, saugo ontologiją su egzemplioriais vienoje lentelėje, dėl to, augant duomenų apimtims, ontologijų apdorojimas labai sulėtėja. Kiti metodai saugo visą ontologijos metamodelį, tačiau šiuo atveju neišnaudojami RDB privalumai greitai išrinkti informaciją, ir tokio vaizdavimo negalima pritaikyti jau esamoms duomenų bazėms.

Informacijos sistemų katedros sukurto *OWL*ToRDB prototipo pranašumas [4] yra hibridinis saugojimo metodas, kai dalis ontologijos tiesiogiai vaizduojama RDB schema, taip išnaudojant RDB privalumus, o ontologijos elementai, kurie tiesiogiai nepavaizduojami RDB, transformuojami į metaduomenų lenteles. Tačiau prototipas neapima visų ontologijos savybių ir yra skirtas pirmai *OWL* versijai, o šiuo metu atsirado *OWL* 2 versija [3], kuri tapo standartu. ***OWL2RDB* prototipe taikomą hibridinį metodą** sudaro šie žingsniai:

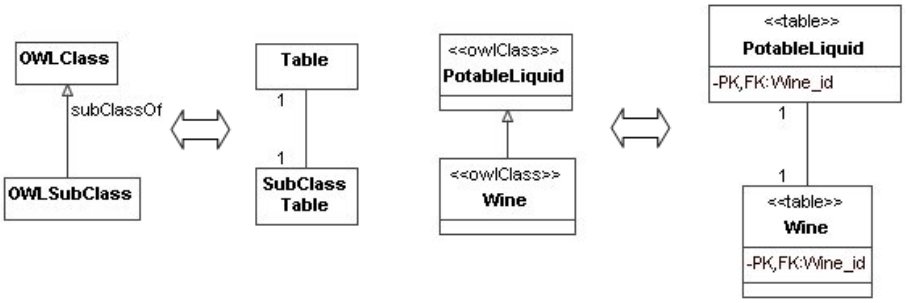
1 žingsnis. *OWL* ontologijos RDB metaschemos sudarymas. *OWL* ontologijos RDB metaschema sudaryta iš 12 automatiškai generuojamų meta lentelių, aprašančių klasių, klasių aksiomų, identifikatorių *IRI*, anotacijų ir kitų elementų savybes [4].

2 žingsnis. Ontologijos klasių transformavimas. Ontologijos klasės transformuojamos į duomenų bazės lenteles (1 pav.), kurios vėliau užpildomos klasių egzemplioriais. Kadangi klasių egzempliorių pavadinimai yra unikalūs, pirminiu raktu paskelbiamas vienas stulpelis, kuriame išsaugomi egzempliorių pavadinimai.



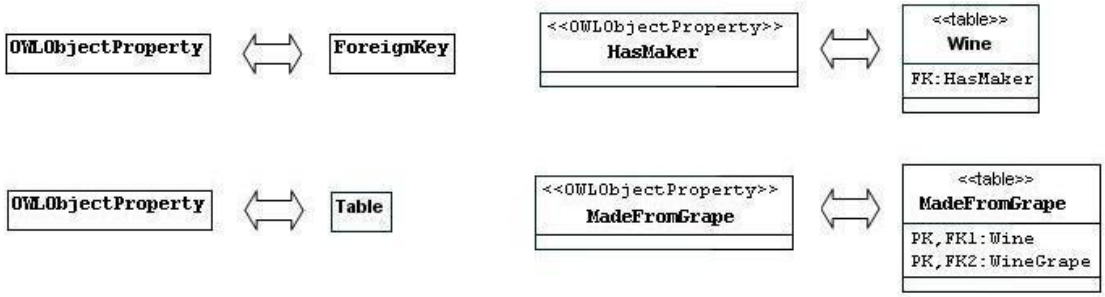
1 pav. Ontologijos klasių transformavimas

Klasių hierarchijai išreikšti naudojami 0..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 (2 pav.).



2 pav. Poklasių transformacija

3 žingsnis. Ontologijos objektinių savybių transformavimas. *OWL* objektinės savybės paverčiamos į ryšius tarp klasių lentelių. Priklausomai nuo jų kardinalumo, naudojamos tarpinės lentelės su išoriniais raktais, kai kardinalumas daug su daug, arba kuriami išoriniai raktai, kai kardinalumas yra vienas su daug (3 pav.)



3 pav. Objektinių savybių transformacija

4 žingsnis. Ontologijos duomenų tipų savybių transformavimas. Duomenų tipų savybės ontologijoje susieja klasių egzempliorius su duomenų tipais. *OWL* duomenų tipų savybės yra paverčiamos naujais stulpeliais lentelėse, kurių klasėms jos taikomos, ir užpildomos atitinkama kiekvieno egzemplioriaus turima reikšme (4 pav.).

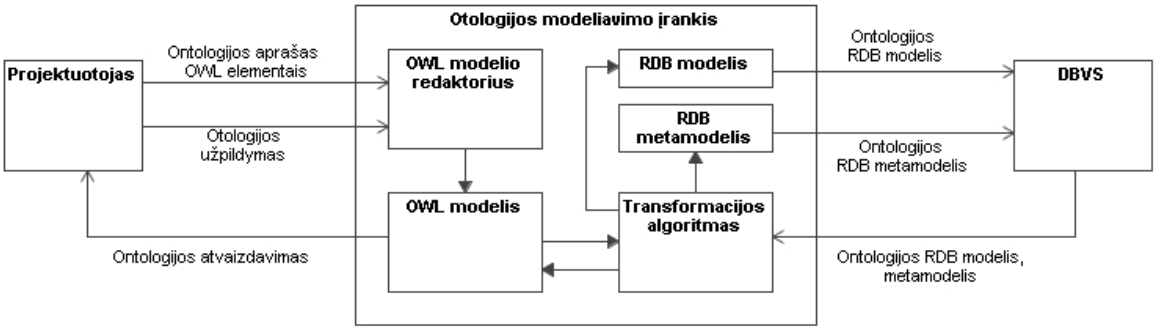
4 pav. Duomenų tipų savybių transformacija

5 žingsnis. Individų transformavimas įrašant į jų tipus atitinkančių klasių lenteles

3. *OWL* 2 transformavimo į RDB įrankis

Naujoji *OWL2ToRDB* transformacija iš esmės taiko tą pačią transformavimo eigą, tačiau ji apima daug daugiau elementų, kurių nebuvo pirmoje *OWL* versijoje, pavyzdžiui, jos metaschema turi 31 lentelę. *OWL2ToRDB* transformacija realizuota kaip ontologijų modeliavimo įrankio *Protegé* įskiepis (5 pav).

OWL2ToRDB transformacija leidžia neprarasti ontologijų informacijos, išsaugodama ontologijos elementų metaduomenis RDB lentelėse, tačiau transformuojamos ontologijos turi atitikti tam tikras taisykles: jos turi būti neprieštaringos, patikrintos ontologijos išvedimo mechanizmu; turi turėti vieną pagrindinę klasių hierarchiją; kiekviena savybė turi turėti apibrėžimo ir kitimo sritį (angl. *domain* ir *range*); kiekvienas individas turi turėti tik vieną tipą. Ne kiekviena ontologija šias sąlygas išpildo, dėl to dalis informacijos gali būti prarandama.



5 pav. Transformavimo sistemos kontekstas

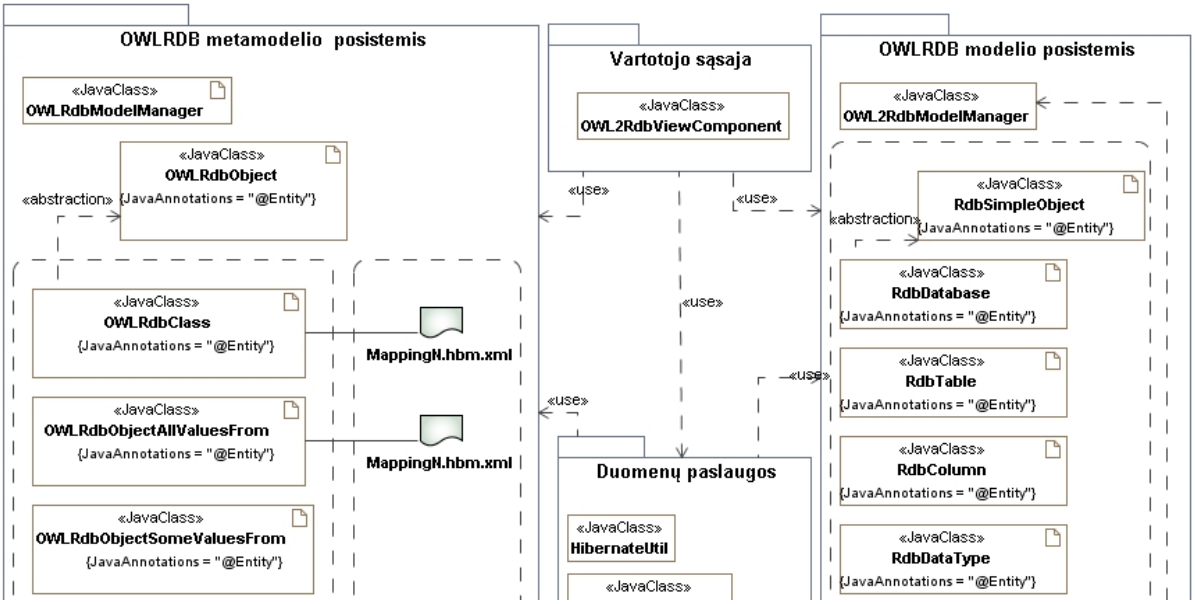
Realizuotas transformavimo įrankis susideda iš vartotojo sąsajos modulio, *OWL 2* ontologijos metamodelio, transformavimo į RDB, duomenų paslaugų modulių (5 pav). Vartotojo sąsajos modulis skirtas palaikyti ryšį tarp projektuotojo ir sistemos, leidžia pasirinkti ontologijos aprašo failą ir duomenų bazių serverį, stebėti transformavimo eigą ir gauti transformavimo ataskaitą. *OWL 2* ontologijos metamodelio valdymo modulis atlieka ontologijos metaduomenų užpildymą. Transformavimo valdymo modulis nagrinėja ontologijos objektus, suformuoja RDB modelį ir *DDL* komandas,kurias vykdo duomenų paslaugų modulis. Pradinis transformavimo rezultatas yra saugomas objektinėje klasių struktūroje, kuri naudojama *DDL* generavimui ir transformavimo

suvestinei formuoti. Ši struktūra paverčiama į *XML* dokumentą, inicijuojamas *XSLT* procesoriaus seansas, kurio metu naudojant *XSL* transformacijų instrukcijų failą gaunama *HTML* formato suvestinė.

Transformaciją vykdančio įskiepio langas pateiktas (59 pav.) Jis susideda iš 1) vykdymo eigos pranešimų lango – išvedami klaidų, vykdymo pranešimai; 2) DDL komandų peržiūros lango – išvedamas sugeneruotas *DDL* skriptas; 3) prisijungimo prie DBVS lango – pateikiami prisijungimo prie duomenų bazės parametrai; 4) transformavimo suvestinės lango – atvaizduojama transformavimo suvestinė.

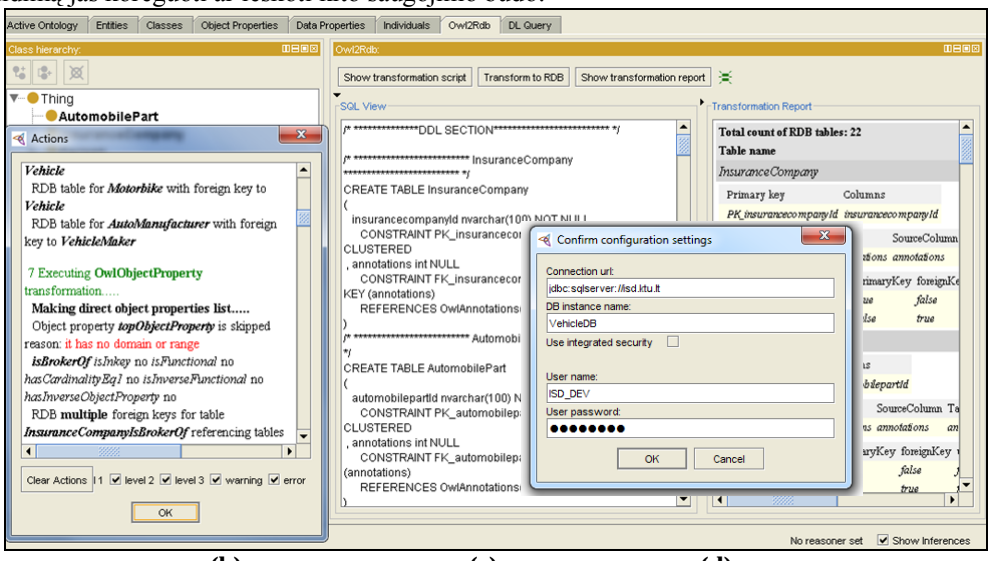
Transformavimo komandos. Pirma komanda „*Show transformation script*“ atlieka *OWL 2* aprašo transformaciją į *SQL* kodą ir jį pateikia lange (6 pav b). „*Transform to RDB*“ atidaro langą (6 pav c), skirtą tiesiogiai atlikti transformaciją į reliacinių duomenų bazių serverį, sukuriant duomenų bazę. Paskutinioji komanda „Show transformation report“ atlieka transformavimo modelio analizę bei pateikia suvestinę (6 pav, d).

Transformaciją galima atlikti tik sėkmingai prisijungus prie duomenų bazių serverio. Pranešimai apie sėkmingai atliktą transformaciją, sukurtus objektus bei įvykusias klaidas pateikiami vykdymo eigos pranešimų lange (6 pav a). Ontologijos *DDL* transformacijoms išsaugoti galima naudoti bet kokią duomenų bazių valdymo sistemą, kuri yra pasiekiama naudojant *JDBC* tvarkyklę.



6 pav.Transformavimo sistemos pagrindiniai architektūriniai komponentai

Testuojant transformavimo įrankį, buvo transformuotas išsamus ontologijos pavyzdys (transporto priemonių ontologija), apimantis visas *OWL 2* konstrukcijas. Kadangi ši ontologija atitiko transformavimo keliamus reikalavimus, visa jos informacija buvo išsaugota RDB. Atliekant eksperimentus su kitomis ontologijomis (pvz., picų, vynų, *FOAF* ir kt.) paaiškėjo, kad daugelis praktikoje sutinkamų ontologijų šių reikalavimų neatitinka, todėl transformavimo įrankis buvo papildytas sukuriant transformavimo eigos stebėjimo galimybes ir transformavimo ataskaitą (6 pava,d). Tuomet vartotojas gali sužinoti, kokios ontologijos konstrukcijos buvo netransformuotos ir priimti sprendimą jas koreguoti ar ieškoti kito saugojimo būdo.



(a) (b) (c) (d)

59 pav.Transformavimo langas *Protege:* vykdymo eiga (a),DDL komandos (b),prisijungimo prie DBVS (c),ataskaita (d)

4. Išvados

Hibridiniu metodu sukurto *OWL2* transformavimo į reliacinę duomenų bazę įrankio bandymai parodė, kad jis gerai veikia su taisyklingomis, reikalavimus atitinkančiomis ontologijomis ir leidžia neprarasti informacijos, tačiau praktikoje sutinkamos ontologijos šių reikalavimų dažnai netenkina, todėl dalis ontologijų informacijos gali būti prarasta. Įrankis buvo papildytas transformavimo eigos stebėjimo ir ataskaitos gavimo galimybėmis, kurios leidžia nustatyti netransformuotas konstrukcijas.

Atliktas tyrimas leidžia daryti išvadą, kad hibridinį ontologijų saugojimo metodą tikslinga taikyti tada, kai esamas ar kuriamas ontologijas norima susieti su taikomosiomis programomis, naudojančiomis reliacines duomenų bazes, nes šiuo metodu sukurta RDB schema yra suprantama taikomųjų programų kūrėjams, leidžia išnaudoti RDB privalumus, greitai išrinkti informaciją ir neprarasti ontologijų informacijos. Esant tokiems tikslams, ontologijas tikslinga pritaikyti prie reikalavimų, keliamų saugojimui RDB.

Literatūros sąrašas

1 **Konstantinou, N., Spanos, D. M., Nikolas, M.** Ontology and database mapping: a survey of current implementations and future directions. *Journal of Web Engineering, Vol. 7(1),* 2008, 001–024.

2 **Motik. B., Horrocks. I., Sattler, U**. Bridging the Gap Between *OWL* and Relational Databases. *In: WWW 2007, International World Wide Web Conference*, 2007, 807–816.

3 **Motik, B., Patel-Schneider, P.F., Parsia, B**. *OWL* 2 Web Ontology Language Structural Specification and Functional-Style Syntax. *W3C Proposed Recommendation 22 September 2009,* available at http://www.w3.org/TR/2009/PR-*OWL*2-syntax-20090922/.

4 **Vyšniauskas, E., Nemuraitė, L., Butleris, R., Paradauskas, B**. Reversible lossless transformation from *OWL* 2 ontologies into relational databases. *Information technology and control, 40(4),* 2011, 293–306.

5 **Vysniauskas, E., Nemuraite, L**. Mapping of *OWL* ontology concepts to RDB schemas. *In Information Technologies' 2009: proceedings of the 15th International Conference on Information and Software Technologies, IT 2009, Kaunas, Lithuania, April 23-24, 2009, Kaunas University of Technology, Kaunas, Technologija*, 2009, 317–327.

Tool for transforming *OWL* 2 ontologies into relational databases

Due to growing importance of ontologies and amounts of ontological information the neccisity arises for storing ontologies in databases. The paper introduces a method which manages transforming of well-formed *OWL* 2 ontologies meeting certain requirements into relational databases without lossing information.

2. priedas. Transformavimo įrankio konfigūracijos failai

Transformavimo sistemos modulio *MANIFEST.MF* failo turinio fragmentas pateiktas žemiau.

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: OWL2Rdb
Bundle-SymbolicName: org.coode.OWL22rdb;singleton:=true
Bundle-Version: 1.0.0.qualifier
Bundle-ClassPath: .,
 lib/antlr-2.7.6.jar,
 lib/c3p0-0.9.1.2.jar,
 lib/commons-collections-3.2.jar,
 lib/commons-logging-1.1.1.jar,
...
Bundle-Activator: org.protege.editor.core.plugin.DefaultPluginActivator
Export-Package: org.coode.OWL22rdb
Require-Bundle:
org.eclipse.equinox.registry,org.eclipse.equinox.common,org.protege.common,or
g.protege.editor.core.application,org.protege.editor.OWL,org.semanticweb.OWL.
OWLapi
Import-Package: javax.net.ssl,
 javax.security.auth,
 javax.swing,
 javax.swing.border,
...
Bundle-DocURL: http://www.isd.ktu.lt/
Bundle-Description: A Plugin allowing export ontologies to RDB using Protégé
OWL Editor
Bundle-RequiredExecutionEnvironment: JavaSE-1.6
Bundle-Category: protege
```

Žymių paaiškinimai:

- **Bundle-Name:** Nurodo modulio pavadinimą;
- **Bundle-SymbolicName:** Vienintelis privalomas atributas, kuris nurodo modulio identifikatorių;
- **Bundle-Description:** Modulio funkcionalumo, paskirties aprašymas;
- **Bundle-ManifestVersion:** Nurodymas OSGi platformai, kokia implementacija turėtų būti naudojama;
- **Bundle-Version:** Versijos numeris;
- **Bundle-Activator:** Nurodo klasės pavadinimą į kurią bus kreipiamasi modulio aktyvavimo metu;
- **Export-Package:** Modulio klasių sąrašas, kurios bus prieinamos kitiems OSGi platformoje veikiantiems moduliams;
- **Import-Package:** Nurodo kokios išorinės klasės yra reikalingos modulio funkcijoms atlikti.

Sekantis taip pat svarbus konfigūracijos failas yra *plugin.xml*, kuris yra naudojamas ne OSGi platformos o pačio *Protégé* modulių paleidimo karkase. Kiekvienas modulis yra atpažįstamas pagal individualiai aprašytus *plugin.xml* failus. Faile nurodomas *plugin identifier*, pavadinimas kuris bus rodomas GUI (angl. *Graphical user interface*), nuoroda į realizuojančią klasę (mūsų atveju į *OWL2RdbViewComponent*). Taip pat keletas plugino savybių tokių kaip, kur jis bus atvaizduojamas, kokios spalvos antraštė. Toliau pateiktas transformavimo sistemos *plugin.xml* failo fragmentas.

```
<?xml version="1.0" ?>
<plugin>
<extensionid="OWL2RdbTab"
point="org.protege.editor.core.application.WorkspaceTab">
<labelvalue="OWL2Rdb"/>
<classvalue="org.protege.editor.OWL.ui.OWLWorkspaceViewsTab"/>
<editorKitIdvalue="OWLEditorKit"/>
<indexvalue="Q"/>
<defaultViewConfigFileNamevalue="viewconfig-OWL22rdbtab.xml"/>
```

```
</extension>

<extensionid="OWL2RdbViewComponent"
   point="org.protege.editor.core.application.ViewComponent">
<labelvalue="OWL2Rdb"/>
<classvalue="org.coode.OWL22rdb.OWL2RdbViewComponent"/>
<headerColorvalue="@org.protege.classcolor"/>
</extension>
</plugin>
```

Faile plugin.xml nurodomas ir papildomas konfigūracijos failas, kuriame aprašoma modulio pagrindinio lango sudėtis. Realizuojamas *Protégé* plėtinys yra suprojektuotas kaip kortelės tipo (*tabwidget*) UI (*User interface*) komponentas, kuris gali būti įvairiai modifikuojamas (pvz. pridedamos papildomos panelės) darbo metu. Norėdami iš anksto apibrėžti UI komponentų rinkinį, kuris bus rodomas ekrane aktyvavus modulį, sukuriame konfigūracijos failą pavadinimu *viewconfig-„plugin pavadinimas“.xml*. Pavyzdys pateiktas žemiau.

```
<?xmlversion="1.0"encoding="UTF-8"?>
<layout>
<VSNodesplits="0.3 0.7">
<CNode>
<Componentlabel="Asserted hierarchy">
<Propertyid="pluginId"value="org.protege.editor.OWL.OWLAssertedClassHierarchy
"/>
</Component>
</CNode>
<CNode>
<Componentlabel="OWL2Rdb">
<Propertyid="pluginId"value="org.coode.OWL22rdb.OWL2RdbViewComponent"/>
</Component>
</CNode>
</VSNode>
      </layout>
```

Lentelė 39 XML dokumento pavyzdys

```
org.coode.OWL22rdb.model.rdb.RdbDatabase>
<name>http://www.exampleontology.com/ontologies/null</name>
<transformedFrom>http://www.exampleontology.com/ontologies/wine example.OWL#
</transformedFrom>
<tables>
<org.coode.OWL22rdb.model.rdb.RdbTable>
<name>InsuranceCompany</name>
<transformedFrom>InsuranceCompany</transformedFrom>
<columns>
<org.coode.OWL22rdb.model.rdb.RdbColumn>
<name>insurancecompanyId</name>
<transformedFrom>InsuranceCompany</transformedFrom>
<type>INT</type>
<primaryKey>true</primaryKey>
<foreignKey>false</foreignKey>
<uniqueKey>false</uniqueKey>
</org.coode.OWL22rdb.model.rdb.RdbColumn>
<org.coode.OWL22rdb.model.rdb.RdbColumn>
<name>annotations</name>
<transformedFrom>Annotations</transformedFrom>
<type>INT</type>
<primaryKey>false</primaryKey>
<foreignKey>true</foreignKey>
<uniqueKey>false</uniqueKey>
</org.coode.OWL22rdb.model.rdb.RdbColumn>
</columns>
<foreignKeys>
<org.coode.OWL22rdb.model.rdb.RdbForeignKeyConstraint>
```

```
<name>FK_insurancecompany_annotations</name>
<transformedFrom>Annotations</transformedFrom>
<sourceColumn>annotations</sourceColumn>
<targetColumn>annotationsId</targetColumn>
<foreignTable>OWLAnnotations</foreignTable>
</org.coode.OWL22rdb.model.rdb.RdbForeignKeyConstraint>
</foreignKeys>
<uniqueKeys/>
<primaryKey>
<name>PK_insurancecompanyId</name>
<transformedFrom>InsuranceCompany</transformedFrom>
<columns>
<string>insurancecompanyId</string>
</columns>
</primaryKey>
</org.coode.OWL22rdb.model.rdb.RdbTable>
```

Lentelė 40 XSL stilių fragmentas

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<table
style="font-family:Times New Roman; font-size:small;
margin-bottom:0; margin-left:0; margin-right:0;"
border="0">
<thead>
<tr bgcolor="#e1e1e1">
      <th colspan="2" style="text-align:left;">
            Total count of RDB tables:
            <xsl:value-of
                  select="count( org.coode.OWL22rdb.model.rdb.RdbDatabase
                  /tables/org.coode.OWL22rdb.model.rdb.RdbTable )" />
      </th>
</tr>
<tr bgcolor="#e1e1e1">
      <th style="text-align:left">Table name</th>
      <th style="text-align:left">Derived from</th>
</tr>
</thead>
<tbody>
<xsl:for-each select="org.coode.OWL22rdb.model.rdb.RdbDatabase">
<xsl:for-each select="tables">
<xsl:for-each select="org.coode.OWL22rdb.model.rdb.RdbTable">
<tr
style="background-color:#d2d2d2; border-bottom-color:black;
 border-bottom-style:solid;">
<td style="font-style:italic; font-weight:bolder; width:5.33in; ">
      <xsl:for-each select="name">
            <xsl:apply-templates />
      </xsl:for-each>
</td>
<td>
      <xsl:for-each select="transformedFrom">
            <xsl:apply-templates />
      </xsl:for-each>
</td>
</tr>
</xsl:for-each>
```

```
</xsl:for-each>
</xsl:for-each>
</tbody>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

3. priedas. Transformavimo suvestinė

Total count of RDB tables: 19

Table name	Derived from
InsuranceCompany	InsuranceCompany

Primary key	Columns
PK_insurancecompanyId	insurancecompanyId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_insurancecompany_annotations	annotations	annotationsId	*OWL*Annotations

Column name	Type	primaryKey	foreignKey	uniqueKey
insurancecompanyId	INT	true	false	false
annotations	INT	false	true	false

Table name	Derived from
AutomobilePart	AutomobilePart

Primary key	Columns
PK_automobilepartId	automobilepartId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_automobilepart_annotations	annotations	annotationsId	*OWL*Annotations

Column name	Type	primaryKey	foreignKey	uniqueKey
automobilepartId	INT	true	false	false
annotations	INT	false	true	false

Table name	Derived from
Person	Person

Primary key	Columns
PK_personId	personId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_person_annotations	annotations	annotationsId	*OWL*Annotations

Column name	Type	primaryKey	foreignKey	uniqueKey
personId	INT	true	false	false
annotations	INT	false	true	false

Table name	Derived from
Vehicle	Vehicle

Primary key	Columns
PK_vehicleId	vehicleId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_vehicle_annotations	annotations	annotationsId	*OWL*Annotations

Column name	Type	primaryKey	foreignKey	uniqueKey
vehicleId	INT	true	false	false
annotations	INT	false	true	false

Table name	Derived from
VehicleMaker	VehicleMaker

Primary key	Columns
PK_vehiclemakerId	vehiclemakerId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_vehiclemaker_annotations	annotations	annotationsId	*OWL*Annotations

Column name	Type	primaryKey	foreignKey	uniqueKey
vehiclemakerId	INT	true	false	false
annotations	INT	false	true	false

Table name	Derived from
Owner	Owner

Primary key	Columns
PK_ownerId	ownerId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_Person_Owner	ownerId	personId	Person

Column name	Type	primaryKey	foreignKey	uniqueKey
ownerId	INT	true	false	false

Table name	Derived from
Assurer	Assurer

Primary key	Columns
PK_assurerId	assurerId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_Person_Assurer	assurerId	personId	Person

Column name	Type	primaryKey	foreignKey	uniqueKey
assurerId	INT	true	false	false

Automobile	Automobile

Primary key	Columns
PK_automobileId	automobileId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_Vehicle_Automobile	automobileId	vehicleId	Vehicle

Column name	Type	primaryKey	foreignKey	uniqueKey
automobileId	INT	true	false	false

Motorbike	Motorbike

Primary key	Columns
PK_motorbikeId	motorbikeId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_Vehicle_Motorbike	motorbikeId	vehicleId	Vehicle

Column name	Type	primaryKey	foreignKey	uniqueKey
motorbikeId	INT	true	false	false

AutoManufacturer	AutoManufacturer

Primary key	Columns
PK_automanufacturerId	automanufacturerId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_VehicleMaker_AutoManufacturer	automanufacturerId	vehiclemakerId	VehicleMaker

Column name	Type	primaryKey	foreignKey	uniqueKey
automanufacturerId	INT	true	false	false

Automobile	isInsuredBy

Primary key	Columns
PK_automobileIsInsuredBy	automobileId insurancecompanyId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_isInsuredBy_InsuranceCompany	insurancecompanyId	insurancecompanyId	InsuranceCompany
FK_isInsuredBy_Automobile	automobileId	automobileId	Automobile

Column name	Type	primaryKey	foreignKey	uniqueKey
automobileId	INT	true	false	false
insurancecompanyId	INT	true	false	false

Assurer	employedBy

Primary key	Columns
PK_assurerEmployedBy	assurerId insurancecompanyId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_employedBy_InsuranceCompany	insurancecompanyId	insurancecompanyId	InsuranceCompany
FK_employedBy_Assurer	assurerId	assurerId	Assurer

Column name	Type	primaryKey	foreignKey	uniqueKey
assurerId	INT	true	false	false
insurancecompanyId	INT	true	false	false

Vehicle	hasMaker

Primary key	Columns
PK_vehicleHasMaker	vehicleId vehiclemakerId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_hasMaker_VehicleMaker	vehiclemakerId	vehiclemakerId	VehicleMaker
FK_hasMaker_Vehicle	vehicleId	vehicleId	Vehicle

Column name	Type	primaryKey	foreignKey	uniqueKey
vehicleId	INT	true	false	false

vehiclemakerId	INT	true	false	false

AutomobilePart — consistsIn

Primary key	Columns
PK_automobilepartConsistsIn	automobilepart1Id automobilepart2Id

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_consistsIn_AutomobilePart1	automobilepart1Id	automobilepartId	AutomobilePart
FK_consistsIn_AutomobilePart2	automobilepart2Id	automobilepartId	AutomobilePart

Column name	Type	primaryKey	foreignKey	uniqueKey
automobilepart1Id	INT	true	false	false
automobilepart2Id	INT	true	false	false

Automobile — isProducedBy

Primary key	Columns
PK_automobileIsProducedBy	automobileId automanufacturerId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_isProducedBy_AutoManufacturer	automanufacturerId	automanufacturerId	AutoManufacturer
FK_isProducedBy_Automobile	automobileId	automobileId	Automobile

Column name	Type	primaryKey	foreignKey	uniqueKey
automobileId	INT	true	false	false
automanufacturerId	INT	true	false	false

Vehicle — registeredOn

Primary key	Columns
PK_vehicleRegisteredOn	vehicleId ownerId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_registeredOn_Vehicle	vehicleId	vehicleId	Vehicle
FK_registeredOn_Owner	ownerId	ownerId	Owner

Column name	Type	primaryKey	foreignKey	uniqueKey
vehicleId	INT	true	false	false
ownerId	INT	true	false	false

InsuranceCompany — isBrokerOf

Primary key	Columns
PK_insurancecompanyIsBrokerOf	insurancecompany1Id insurancecompany2Id

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_isBrokerOf_InsuranceCompany2	insurancecompany2Id	insurancecompanyId	InsuranceCompany
FK_isBrokerOf_InsuranceCompany1	insurancecompany1Id	insurancecompanyId	InsuranceCompany

Column name	Type	primaryKey	foreignKey	uniqueKey
insurancecompany1Id	INT	true	false	false
insurancecompany2Id	INT	true	false	false

Vehicle — previousOwner

Primary key	Columns
PK_vehiclePreviousOwner	vehicleId ownerId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_previousOwner_Owner	ownerId	ownerId	Owner
FK_previousOwner_Vehicle	vehicleId	vehicleId	Vehicle

Column name	Type	primaryKey	foreignKey	uniqueKey
vehicleId	INT	true	false	false
ownerId	INT	true	false	false

Vehicle — currentOwner

Primary key	Columns
PK_vehicleCurrentOwner	vehicleId ownerId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable

FK_currentOwner_Owner	ownerId	ownerId	Owner
FK_currentOwner_Vehicle	vehicleId	vehicleId	Vehicle

Column name	Type	primaryKey	foreignKey	uniqueKey
vehicleId	INT	true	false	false
ownerId	INT	true	false	false

Automobile assembledFrom

Primary key	Columns
PK_automobileAssembledFrom	automobileId automobilepartId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_assembledFrom_Automobile	automobileId	automobileId	Automobile
FK_assembledFrom_AutomobilePart	automobilepartId	automobilepartId	AutomobilePart

Column name	Type	primaryKey	foreignKey	uniqueKey
automobileId	INT	true	false	false
automobilepartId	INT	true	false	false

Automobile isVerifiedBy

Primary key	Columns
PK_automobileIsVerifiedBy	automobileId assurerId

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_isVerifiedBy_Assurer	assurerId	assurerId	Assurer
FK_isVerifiedBy_Automobile	automobileId	automobileId	Automobile

Column name	Type	primaryKey	foreignKey	uniqueKey
automobileId	INT	true	false	false
assurerId	INT	true	false	false

InsuranceCompany trusts

Primary key	Columns
PK_insurancecompanyTrusts	insurancecompany1Id insurancecompany2Id

Foreign Key name	SourceColumn	TargetColumn	ForeignTable
FK_trusts_InsuranceCompany2	insurancecompany2Id	insurancecompanyId	InsuranceCompany
FK_trusts_InsuranceCompany1	insurancecompany1Id	insurancecompanyId	InsuranceCompany

Column name	Type	primaryKey	foreignKey	uniqueKey
insurancecompany1Id	INT	true	false	false
insurancecompany2Id	INT	true	false	false

4. priedas. Transformavimo protokolas

1 logged at 19:32:38-422 Start creating metamodel.....

---

2 logged at 19:32:38-600 Finnished creating metamodel.....

---

3 logged at 19:32:38-600 Start executing transformation.....

---

4 Building ***OWL*Class** queue.....

5 Executing ***OWL*Class** transformation.....

RDB table for ***ValuePartition*** with foreign key to ***OWLAnnotations***
RDB table for ***DomainConcept*** with foreign key to ***OWLAnnotations***
RDB table for ***Spiciness*** with foreign key to ***ValuePartition***
RDB table for ***Food*** with foreign key to ***DomainConcept***
RDB table for ***Country*** with foreign key to ***DomainConcept***
RDB table for ***Medium*** with foreign key to ***Spiciness***
RDB table for ***Hot*** with foreign key to ***Spiciness***
RDB table for ***Mild*** with foreign key to ***Spiciness***
RDB table for ***IceCream*** with foreign key to ***Food***
RDB table for ***Pizza*** with foreign key to ***Food***
RDB table for ***PizzaTopping*** with foreign key to ***Food***
RDB table for ***PizzaBase*** with foreign key to ***Food***
RDB table for ***NamedPizza*** with foreign key to ***Pizza***
RDB table for ***MeatyPizza*** with foreign key to ***Pizza***
RDB table for ***InterestingPizza*** with foreign key to ***Pizza***
RDB table for ***VegetarianPizzaEquivalent1*** with foreign key to ***Pizza***
RDB table for ***ThinAndCrispyPizza*** with foreign key to ***Pizza***
RDB table for ***CheeseyPizza*** with foreign key to ***Pizza***
RDB table for ***VegetarianPizzaEquivalent2*** with foreign key to ***Pizza***
RDB table for ***SpicyPizza*** with foreign key to ***Pizza***
RDB table for ***NonVegetarianPizza*** with foreign key to ***Pizza***
RDB table for ***RealItalianPizza*** with foreign key to ***Pizza***
RDB table for ***SpicyPizzaEquivalent*** with foreign key to ***Pizza***
RDB table for ***VegetarianPizza*** with foreign key to ***Pizza***
RDB table for ***FruitTopping*** with foreign key to ***PizzaTopping***
RDB table for ***VegetarianTopping*** with foreign key to ***PizzaTopping***
RDB table for ***SpicyTopping*** with foreign key to ***PizzaTopping***
RDB table for ***CheeseTopping*** with foreign key to ***PizzaTopping***
RDB table for ***SauceTopping*** with foreign key to ***PizzaTopping***
RDB table for ***HerbSpiceTopping*** with foreign key to ***PizzaTopping***
RDB table for ***MeatTopping*** with foreign key to ***PizzaTopping***
RDB table for ***VegetableTopping*** with foreign key to ***PizzaTopping***
RDB table for ***NutTopping*** with foreign key to ***PizzaTopping***
RDB table for ***FishTopping*** with foreign key to ***PizzaTopping***
RDB table for ***DeepPanBase*** with foreign key to ***PizzaBase***
RDB table for ***ThinAndCrispyBase*** with foreign key to ***PizzaBase***
RDB table for ***Mushroom*** with foreign key to ***NamedPizza***
RDB table for ***AmericanHot*** with foreign key to ***NamedPizza***
RDB table for ***LaReine*** with foreign key to ***NamedPizza***
RDB table for ***Napoletana*** with foreign key to ***NamedPizza***
RDB table for ***SloppyGiuseppe*** with foreign key to ***NamedPizza***
RDB table for ***Giardiniera*** with foreign key to ***NamedPizza***
RDB table for ***PolloAdAstra*** with foreign key to ***NamedPizza***
RDB table for ***FourSeasons*** with foreign key to ***NamedPizza***
RDB table for ***UnclosedPizza*** with foreign key to ***NamedPizza***
RDB table for ***Cajun*** with foreign key to ***NamedPizza***
RDB table for ***QuattroFormaggi*** with foreign key to ***NamedPizza***
RDB table for ***Capricciosa*** with foreign key to ***NamedPizza***

RDB table for ***Veneziana*** with foreign key to ***NamedPizza***
RDB table for ***Parmense*** with foreign key to ***NamedPizza***
RDB table for ***Soho*** with foreign key to ***NamedPizza***
RDB table for ***PrinceCarlo*** with foreign key to ***NamedPizza***
RDB table for ***Rosa*** with foreign key to ***NamedPizza***
RDB table for ***American*** with foreign key to ***NamedPizza***
RDB table for ***Fiorentina*** with foreign key to ***NamedPizza***
RDB table for ***Siciliana*** with foreign key to ***NamedPizza***
RDB table for ***FruttiDiMare*** with foreign key to ***NamedPizza***
RDB table for ***Caprina*** with foreign key to ***NamedPizza***
RDB table for ***Margherita*** with foreign key to ***NamedPizza***
RDB table for ***SultanaTopping*** with foreign key to ***FruitTopping***
RDB table for ***CheeseyVegetableTopping*** with foreign key to ***CheeseTopping***
RDB table for ***MozzarellaTopping*** with foreign key to ***CheeseTopping***
RDB table for ***GorgonzolaTopping*** with foreign key to ***CheeseTopping***
RDB table for ***GoatsCheeseTopping*** with foreign key to ***CheeseTopping***
RDB table for ***ParmesanTopping*** with foreign key to ***CheeseTopping***
RDB table for ***FourCheesesTopping*** with foreign key to ***CheeseTopping***
RDB table for ***TobascoPepperSauce*** with foreign key to ***SauceTopping***
RDB table for ***CajunSpiceTopping*** with foreign key to ***HerbSpiceTopping***
RDB table for ***RosemaryTopping*** with foreign key to ***HerbSpiceTopping***
RDB table for ***HotSpicedBeefTopping*** with foreign key to ***MeatTopping***
RDB table for ***HamTopping*** with foreign key to ***MeatTopping***
RDB table for ***PeperoniSausageTopping*** with foreign key to ***MeatTopping***
RDB table for ***ChickenTopping*** with foreign key to ***MeatTopping***
RDB table for ***PetitPoisTopping*** with foreign key to ***VegetableTopping***
RDB table for ***CaperTopping*** with foreign key to ***VegetableTopping***
RDB table for ***ArtichokeTopping*** with foreign key to ***VegetableTopping***
RDB table for ***RocketTopping*** with foreign key to ***VegetableTopping***
RDB table for ***LeekTopping*** with foreign key to ***VegetableTopping***
RDB table for ***PepperTopping*** with foreign key to ***VegetableTopping***
RDB table for ***GarlicTopping*** with foreign key to ***VegetableTopping***
RDB table for ***AsparagusTopping*** with foreign key to ***VegetableTopping***
RDB table for ***SpinachTopping*** with foreign key to ***VegetableTopping***
RDB table for ***OliveTopping*** with foreign key to ***VegetableTopping***
RDB table for ***OnionTopping*** with foreign key to ***VegetableTopping***
RDB table for ***MushroomTopping*** with foreign key to ***VegetableTopping***
RDB table for ***TomatoTopping*** with foreign key to ***VegetableTopping***
RDB table for ***PineKernels*** with foreign key to ***NutTopping***
RDB table for ***AnchoviesTopping*** with foreign key to ***FishTopping***
RDB table for ***MixedSeafoodTopping*** with foreign key to ***FishTopping***
RDB table for ***PrawnsTopping*** with foreign key to ***FishTopping***
RDB table for ***ParmaHamTopping*** with foreign key to ***HamTopping***
RDB table for ***JalapenoPepperTopping*** with foreign key to ***PepperTopping***
RDB table for ***PeperonataTopping*** with foreign key to ***PepperTopping***
RDB table for ***SweetPepperTopping*** with foreign key to ***PepperTopping***
RDB table for ***GreenPepperTopping*** with foreign key to ***PepperTopping***
RDB table for ***RedOnionTopping*** with foreign key to ***OnionTopping***
RDB table for ***SundriedTomatoTopping*** with foreign key to ***TomatoTopping***
RDB table for ***SlicedTomatoTopping*** with foreign key to ***TomatoTopping***
RDB table for ***HotGreenPepperTopping*** with foreign key to ***GreenPepperTopping***
---
99 *OWL*Classes of 99 transformed
---

6 Executing ***OWL*ObjectProperty** transformation.....

Making direct object properties list.....
Warning: Object property ***hasCountryOfOrigin*** is skipped reason: it has no domain or range
Warning: Object property ***hasSpiciness*** is skipped reason: it has no domain or range
hasBase*isInkey* no *isFunctional***yes***hasCardinalityEq1* no *isInverseFunctional***yes***hasInverseObjectProperty***yes**
RDB foreign key ***hasBase*** for table ***Pizza*** referencing table ***PizzaBase*** (rel: one to many)

hasIngredient*isInkey* no *isFunctional* no *hasCardinalityEq1* no *isInverseFunctional* no *hasInverseObjectProperty***yes**
RDB **multiple** foreign keys for table ***FoodHasIngredient*** referencing tables ***Food,Food*** (rel: many to many)

isToppingOf*isInkey* no *isFunctional***yes***hasCardinalityEq1* no *isInverseFunctional* no *hasInverseObjectProperty***yes**
RDB foreign key ***isToppingOf*** for table ***PizzaTopping*** referencing table ***Pizza*** (rel: one to many)

---

6 *OWL*ObjectProperties of 8 transformed

---

7 Executing ***OWL*DataProperty** transformation.....

---

0 *OWL*DataProperties of 0 transformed

---

8 Executing ***OWL*HasKey** transformation.....

---

0 *OWL*HasKey of 0 transformed

---

9 Executing ***OWL*Individual** classAssertion transformation.....

Warning: Individual ***France*** is skipped reason: has more that one ClassAssertionAxiom
Warning: Individual ***England*** is skipped reason: has more that one ClassAssertionAxiom
Warning: Individual ***England*** ClassAssertionAxiom is skipped reason: it's *OWL*Thing
Warning: Individual ***Italy*** is skipped reason: has more that one ClassAssertionAxiom
Warning: Individual ***Germany*** is skipped reason: has more that one ClassAssertionAxiom
Warning: Individual ***Germany*** ClassAssertionAxiom is skipped reason: it's *OWL*Thing
Warning: Individual ***America*** is skipped reason: has more that one ClassAssertionAxiom
Warning: Individual ***America*** ClassAssertionAxiom is skipped reason: it's *OWL*Thing

---

2 *OWL*Individual classAssertions of 10 transformed

---

10 Executing ***OWL*Individual** object properties transformation.....

---

0 *OWL*Individual object properties of 0 transformed

---

11 Executing ***OWL*Individual** data properties transformation.....

---

0 *OWL*Individual data properties of 0 transformed

---

12 logged at 19:32:39-27 End executing transformation.....

---

13 logged at 19:32:39-72 Saving transformation to RDB.....

---

14 Synchronizing metadata structure.....

15 Saving matadata.....

16 Executing DDL script.....

17 Executing DML script.....

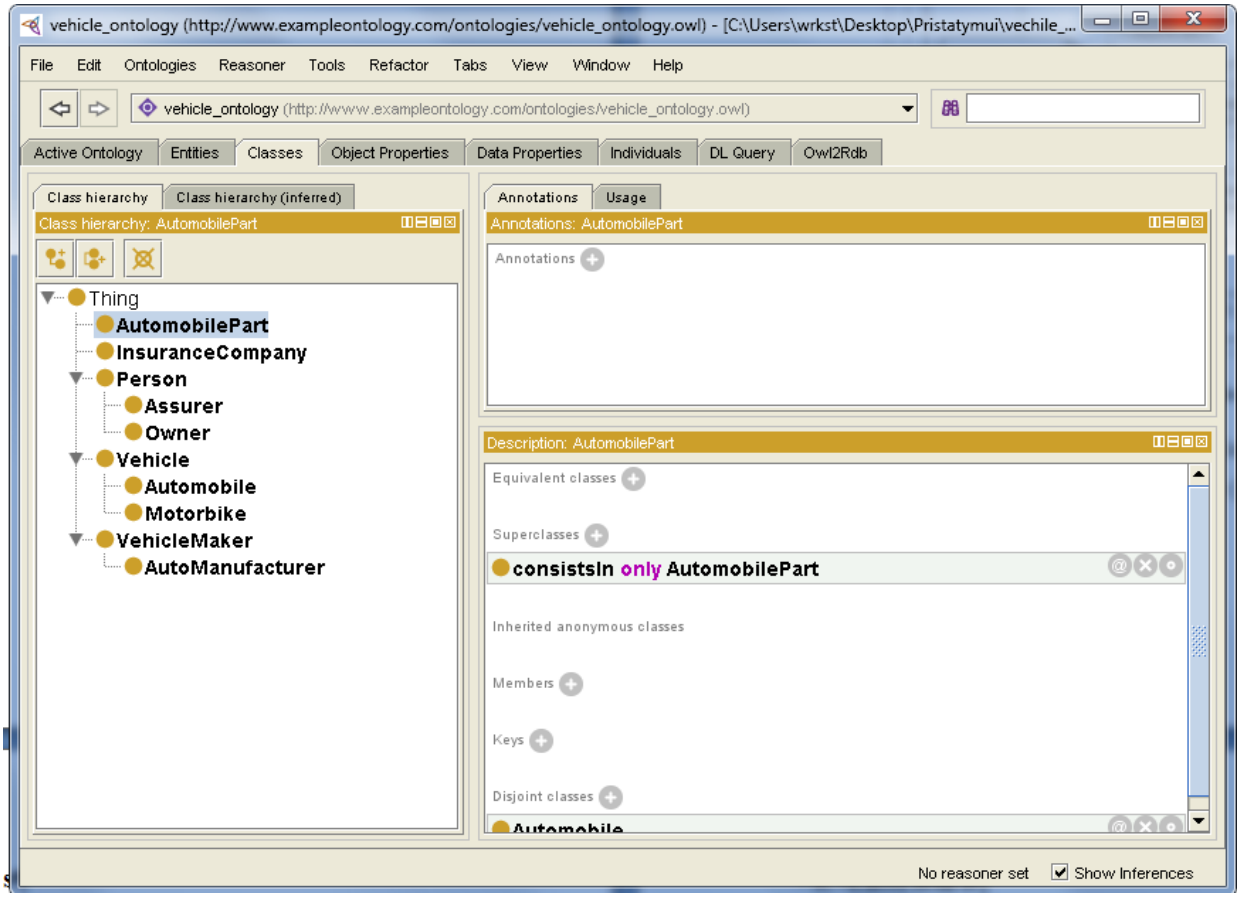
18 Adding UK constraints.....

19 logged at 19:32:50-473 Transformation successfully saved to RDB.....

---

5. priedas. Vehicle ontologijos aprašymas

Vehicle ontologija sudaryta iš 10 klasių. Įkėlus ontologiją į *Protégé*, visos klases yra atvaizduojamos klasių hierarchijos lange (60pav). Klasių aprašas pateikiamas (Lentelė 41)

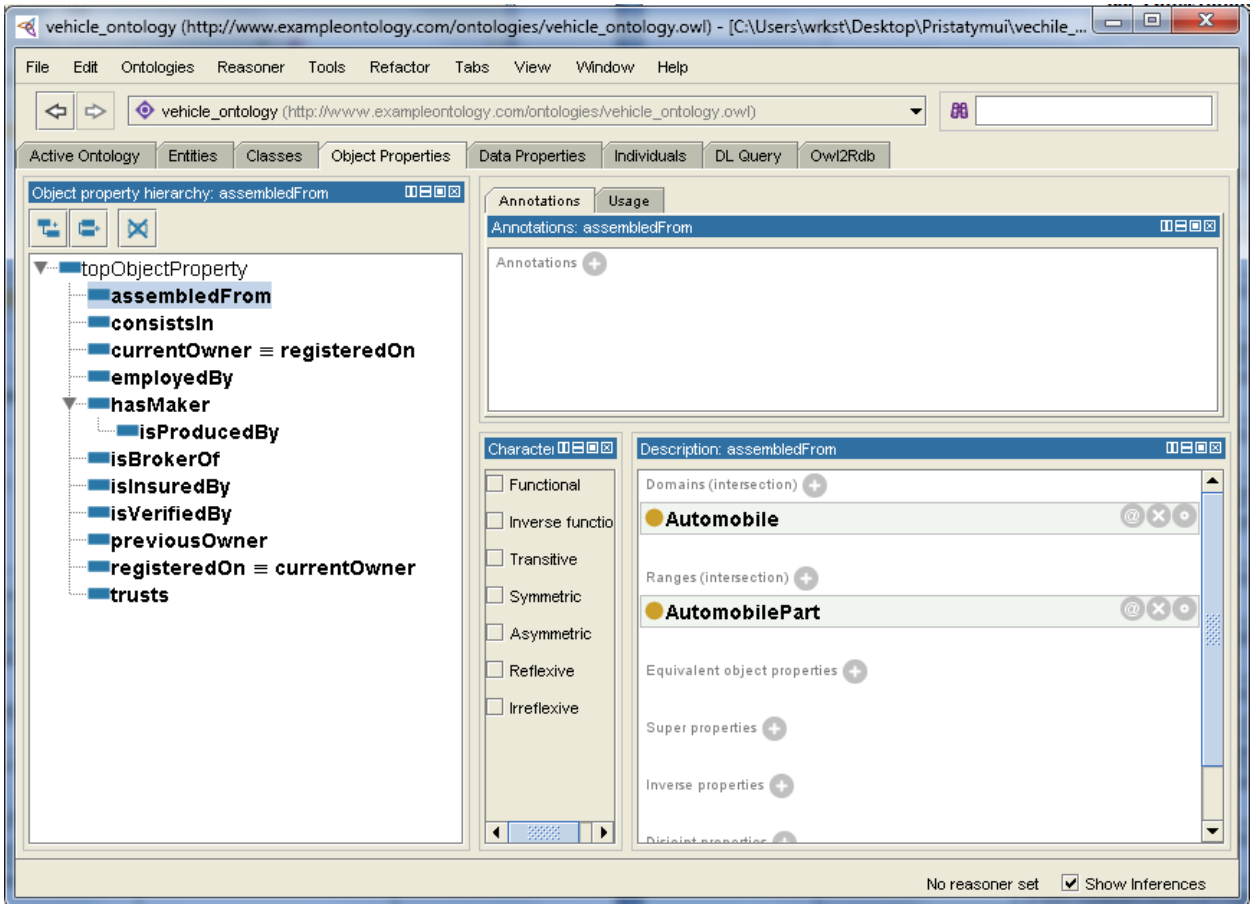


60 pav. Klasių hierarchija

Lentelė 41Venchile ontologijos *OWL*2 klasių aprašas

***OWL* class**	**Class axioms**	**Annotations**
a:Vehicle	DisjointClasses(*a:Vehicle a:VehicleMaker a:AutomobilePart a:Person a:InsuranceCompany*)	---
a:Automobile	SubClassOf(*a:Automobile a:Vehicle*)	---
	DisjointClasses(*a:Automobile a:AutomobilePart a:Motorbike a:VehicleMaker*)	
a:Motorbike	SubClassOf(*a:Motorbike a:Vehicle*)	---
a:VehicleMaker	---	---
a:AutoManufacturer	SubClassOf(*a:AutoManufacturer a:VehicleMaker*)	"*Automobile manufacturing company*"
a:Person	---	
a:Assurer	SubClassOf(*a:Assuer a:Person*)	"*An assurer is a person who insures*"
a:Owner	SubClassOf(*a:Owner a:Person*)	"*A person who owns a vehicle*"
a:AutomobilePart	---	---
a:InsuranceCompany	---	---

Pavyzdinėje ontologijoje taip pat aprašytos 12 objektinių savybių. (61pav) ir (Lentelė 42)



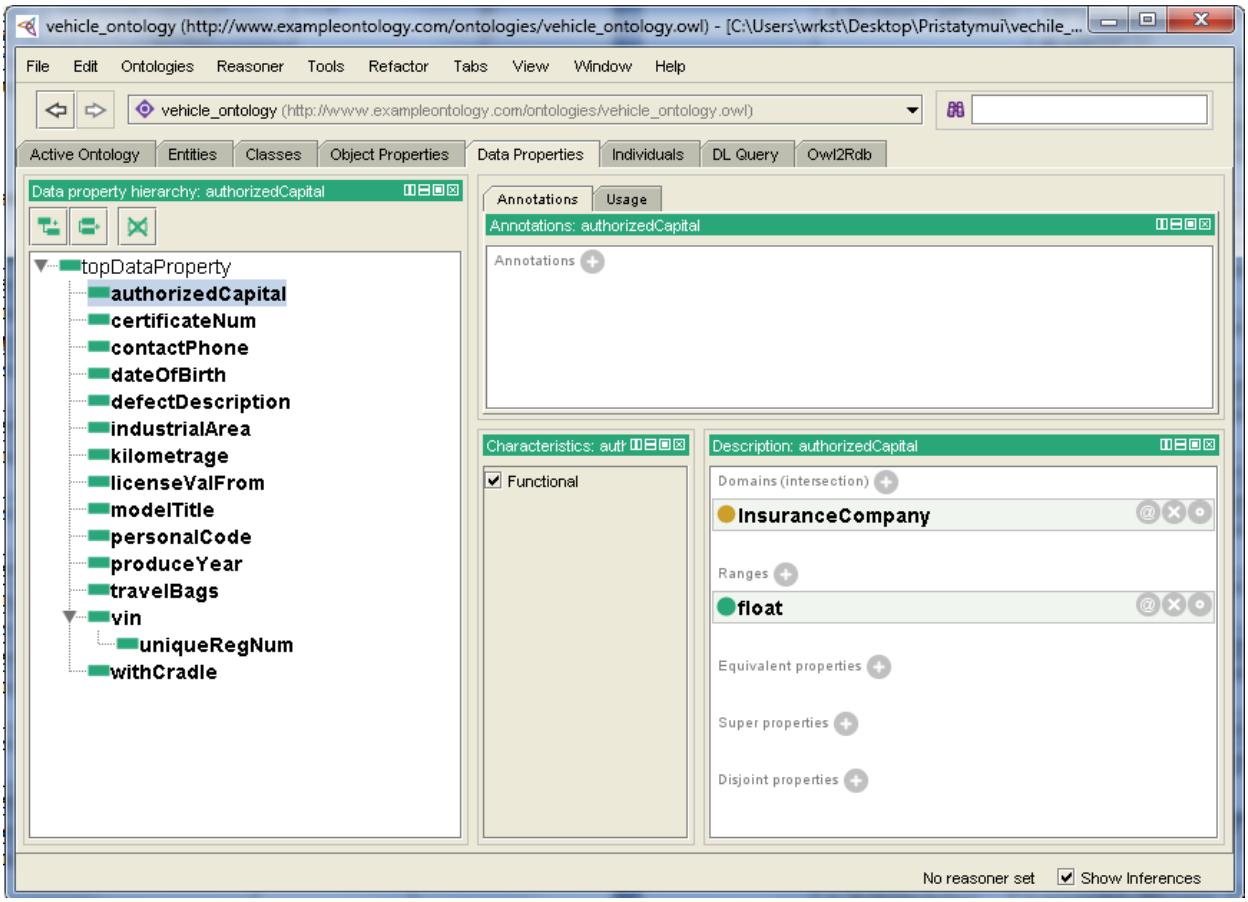
61 pav. Objektinių savybių hierarchijos langas

Lentelė 42Venchile ontologijos *OWL*2 objektinių savybių aprašas

Object property	Object property axioms	**Annotations**
a:assembledFrom	ObjectPropertyDomain(*a:assembledFrom a:Automobile*)	---
	ObjectPropertyRange(*a:assembledFrom a:AutomobilePart*)	
a:consistsIn	ObjectPropertyDomain(*a:consistsIn a:AutomobilePart*)	---
	ObjectPropertyRange(*a:consistsIn a:AutomobilePart*)	
	FunctionalObjectProperty(*a:consistsIn*)	
a:currentOwner	ObjectPropertyDomain(*a:currentOwner a:Vehicle*)	*"A person who currently own a vehicle"*
	ObjectPropertyRange(*a:currentOwner a:Owner*)	
	FunctionalObjectProperty(*a:currentOwner*)	
	EquivalentObjectProperties(*a:currentOwner a:registeredOn*)	
	DisjointObjectProperties(*a:currentOwner a:previousOwner*)	
a:employedBy	ObjectPropertyDomain(*a:employedBy a:Assurer*)	---
	ObjectPropertyRange(*a:employedBy a:InsuranceCompany*)	
	FunctionalObjectProperty(*a:employedBy*)	
a:hasMaker	ObjectPropertyDomain(*a:hasMaker a:Vehicle*)	---
	ObjectPropertyRange(*a:hasMaker a:VehicleMaker*)	
	FunctionalObjectProperty(*a:hasMaker*)	
a:isProducedBy	ObjectPropertyDomain(*a:isProducedBy a:Automobile*)	---
	ObjectPropertyRange(*a:isProducedBy a:AutoManufacturer*)	
	SubObjectPropertyOf(*a:isProducedBy a:hasMaker*)	
	FunctionalObjectProperty(*a:isProducedBy*)	
a:isBrokerOf	ObjectPropertyDomain(*a:isBrokerOf a:InsuranceCompany*)	---
	ObjectPropertyRange(*a:isBrokerOf a:InsuranceCompany*)	
	FunctionalObjectProperty(*a:isBrokerOf*)	
	IrreflexiveObjectProperty(*a:isBrokerOf*)	
a:isInsuredBy	ObjectPropertyDomain(*a:isInsuredBy a:Automobile*)	*"An automobile is insured by the insurance company"*
	ObjectPropertyRange(*a:isInsuredBy a:InsuranceCompany*)	
	SubObjectPropertyOf(ObjectPropertyChain(*a:isVerifiedBy a:employedBy*)	

Object property	Object property axioms	**Annotations**
	a:isInsuredBy)	
	FunctionalObjectProperty(*a:isInsuredBy*)	
a:isVerifiedBy	ObjectPropertyDomain(*a:isVerifiedBy a:Automobile*)	---
	ObjectPropertyRange(*a:isVerifiedBy a:Assurer*)	
	FunctionalObjectProperty(*a:isVerifiedBy*)	
a:previousOwner	ObjectPropertyDomain(*a:previousOwner a:Vehicle*)	*"A previous owner of the vehicle"*
	ObjectPropertyRange(*a:previousOwner a:Owner*)	
	DisjointObjectProperties(*a:previousOwner a:currentOwner*)	
a:registeredOn	ObjectPropertyDomain(*a:registeredOn a:Vehicle*)	*"A vehicle is registered on a person"*
	ObjectPropertyRange(*a:registeredOn a:Owner*)	
	FunctionalObjectProperty(*a:registeredOn*)	
	EquivalentObjectProperties(*a:registeredOn a:currentOwner*)	
a:trusts	ObjectPropertyDomain(*a:trusts a:InsuranceCompany*)	---
	ObjectPropertyRange(*a:trusts a:InsuranceCompany*)	
	ReflexiveObjectProperty(*a:trusts*)	

Vehicle ontologija turi penkiolika duomenų savybių, kurios atvaizduojamos duomenų savybių hierarchijos lange (62pav) ir (Lentelė 43)



62 pav. Duomenų savybių hierarchijos langas

Lentelė 43Venchile ontologijos *OWL*2 duomenų savybių aprašas

Data property	**Data property axioms**	**Annotations**
a:authorizedCapital	DataPropertyDomain(a:*authorizedCapital a:InsuranceCompany*)	---
	DataPropertyRange(*a:authorizedCapital xsd:float*)	
	FunctionalDataProperty(*a:authorizedCapital*)	
a:certificateNum	DataPropertyDomain(a:*certificateNum a:Assurer*)	---
	DataPropertyRange(*a:certificateNum xsd:string*)	
	FunctionalDataProperty(*a:certificateNum*)	
a:vin	DataPropertyDomain(*a:vin a:Vehicle*)	*"Vehicle identification*
	DataPropertyRange(*a:vin xsd:string*)	

Data property	Data property axioms	Annotations
	FunctionalDataProperty(*a:vin*)	*number"*
a:uniqueRegNum	DataPropertyDomain(*a:uniqueRegNum a:Motorbike*)	---
	DataPropertyRange(*a:uniqueRegNum xsd:string*)	
	FunctionalDataProperty(*a:uniqueRegNum*)	
	SubDataPropertyOf(*a:uniqueRegNum a:vin*)	
a:contactPhone	DataPropertyDomain(*a:contactPhone a:Assurer*)	---
	DataPropertyRange(*a:contactPhone xsd:string*)	
a:dateOfBirth	DataPropertyDomain(*a:dateOfBirth a:Person*)	---
	DataPropertyRange(*a:dateOfBirth xsd:date*)	
	FunctionalDataProperty(*a:dateOfBirth*)	
a:defectDescription	DataPropertyDomain(*a:defectDescription a:Automobile*)	*"Can not be more than three defect descriptions"*
	DataPropertyRange(*a:defectDescription xsd:string*)	
a:industrialArea	DataPropertyDomain(*a:industrialArea a:VehicleMaker*)	---
	DataPropertyRange(*a:industrialArea xsd:string*)	
a:kilometrage	DataPropertyDomain(*a:kilometrage a:Vehicle*)	---
	DataPropertyRange(*a:kilometrage xsd:integer*)	
	FunctionalDataProperty(*a:kilometrage*)	
a:licenseValFrom	DataPropertyDomain(*a:licenseValFrom a:Owner*)	*"The driving license is valid from"*
	DataPropertyRange(*a:licenseValFrom xsd:date*)	
	FunctionalDataProperty(*a:licenseValFrom*)	
a:modelTitle	DataPropertyDomain(*a:modelTitle a:Automobile*)	---
	DataPropertyRange(*a:modelTitle xsd:string*)	
	FunctionalDataProperty(*a:modelTitle*)	
a:personalCode	DataPropertyDomain(*a:personalCode a:Person*)	*"Unique number of a person"*
	DataPropertyRange(*a:personalCode xsd:string*)	
	FunctionalDataProperty(*a:personalCode*)	
a:produceYear	DataPropertyDomain(*a:produceYear a:Automobile*)	---
	DataPropertyRange(*a:produceYear xsd:date*)	
	FunctionalDataProperty(*a:produceYear*)	
a:travelBags	DataPropertyDomain(*a:travelBags a:Motorbike*)	---
	DataPropertyRange(*a:travelBags xsd:integer*)	
	FunctionalDataProperty(*a:travelBags*)	
a:withCradle	DataPropertyDomain(*a:withCradle a:Motorbike*)	---
	DataPropertyRange(*a:withCradle xsd:byte*)	
	FunctionalDataProperty(*a:withCradle*)	

Lentelė 44Venchile ontologijos *OWL*2 apribojimų aprašas

OWL class	Object/data property	Restrictions
a:Automobile	a:assembledFrom	ObjectMinCardinality(*5 a:assembledFrom a:AutomobilePart*)
		ObjectMaxCardinality(*255 a:assembledFrom a:AutomobilePart*)
	a:isProducedBy	ObjectAllValuesFrom(*a:isProducedBy a:AutoManufacturer*)
	a:defectDescription	DataMaxCardinality(*3 a:defectDescription xsd:string*)
a:AutomobilePart	a:consistsIn	ObjectAllValuesFrom(*a:consistsIn a:AutomobilePart*)

Lentelė 45Venchile ontologijos *OWL*2 klasių egzempliorių (individų) aprašas

***OWL* class**	**Individuals**	**Annotations**
a:Vehicle	a:AutoEBB887	*"This vehicle is an automobile"*
	a:AutoDOT475	---

OWL class	Individuals	Annotations
	a:AutoCKD843	---
	a:AutoGKT274	---
	a:AutoRKU418	---
	a:Moto568AE	*"This vehicle is a motorbike"*
	a:Moto968UH	---
	a:Moto675SR	---
a:Automobile	a:AutoEBB887	---
	a:AutoDOT475	*"The car has motor defect"*
	a:AutoCKD843	*"The car has gearbox defect"*
	a:AutoGKT274	---
	a:AutoRKU418	---
a:Motorbike	a:Moto568AE	---
	a:Moto968UH	---
	a:Moto675SR	*"The bike is with a cradle and two bags"*
a:VehicleMaker	a:Yamaha	---
	a:Kawasaki	*"Motorbikes manufacturing company"*
	a:Toyota	*"Japanese manufacturer"*
	a:Hyundai	*"Korean manufacturer"*
	a:Nissan	---
	a:Mitsubishi	---
a:AutoManufacturer	a:Toyota	---
	a:Hyundai	*"Hyundai Motor Company"*
	a:Nissan	---
	a:Mitsubishi	*"Mitsubishi Motors"*
a:Person	a:JohnMayer	*"This person is an assurer"*
	a:PeterIrving	---
	a:AdamBradley	---
	a:EricBrown	*"Driving license more than 10 years"*
	a:RobertStewart	---
a:Assurer	a:JohnMayer	*"Works in "ZurichConnect" company"*
	a:PeterIrving	*"Works in "BrokerKing" company"*
a:Owner	a:AdamBradley	---
	a:EricBrown	---
	a:RobinStewart	---
a:AutomobilePart	a:SteeringWheel	---
	a:Engine	---
	a:CylinderHead	*"Is on top of the cylinder block"*
	a:Distributor	*"A device in the ignition system"*
	a:Gearbox	*"Transmission system"*
a:InsuranceCompany	a:ZurichConnect	*"Swiss insurance company"*
	a:MotorDirect	---
	a:BrokerKing	*"Is a broker of "ZurichConnect" company"*
	a:AutoTrader	---
	a:DivaInsurance	*"Cheap car insurance company"*

Lentelė 46Venchile ontologijos *OWL*2 individų objektinių savybių (assertion) aprašas

Object property	Individuals	Assertions
a:consistsIn	a:CylinderHead	ObjectPropertyAssertion(*a:consistsIn a:CylinderHead a:Engine*)
	a:Distributor	ObjectPropertyAssertion(a*:consistsIn a:Distributor a:Engine*)
a:currentOwner	a:AutoEBB887	ObjectPropertyAssertion(*a:currentOwner a:AutoEBB887 a:AdamBradley*)
	a:AutoDOT475	ObjectPropertyAssertion(*a:currentOwner a:AutoDOT475 a:EricBrown*)
	a:AutoCKD843	ObjectPropertyAssertion(*a:currentOwner a:AutoCKD843 a:EricBrown*)
	a:AutoGKT274	ObjectPropertyAssertion(*a:currentOwner*

Object property	Individuals	Assertions
		a:AutoGKT274 a:RobinStewart)
	a:AutoRKU418	ObjectPropertyAssertion(*a:currentOwner a:AutoRKU418 a:RobinStewart*)
a:employedBy	a:JohnMayer	ObjectPropertyAssertion(*a:employedBy a:JohnMayer a:ZurichConnect*)
	a:PeterIrving	ObjectPropertyAssertion(*a:employedBy a:PeterIrving a:BrokerKing*)
a:hasMaker	a:AutoEBB887	ObjectPropertyAssertion(*a:hasMaker a:AutoEBB887 a:Toyota*)
	a:AutoDOT475	ObjectPropertyAssertion(*a:hasMaker a:AutoDOT475 a:Hyundai*)
	a:AutoCKD843	ObjectPropertyAssertion(*a:hasMaker a:AutoCKD843 a:Mitsubishi*)
	a:AutoGKT274	ObjectPropertyAssertion(*a:hasMaker a:AutoGKT274 a:Nissan*)
	a:AutoRKU418	ObjectPropertyAssertion(*a:hasMaker a:AutoRKU418 a:Toyota*)
	a:Moto568AE	ObjectPropertyAssertion(*a:hasMaker a:Moto568AE a:Kawasaki*)
	a:Moto968UH	ObjectPropertyAssertion(*a:hasMaker a:Moto968UH a:Kawasaki*)
	a:Moto675SR	ObjectPropertyAssertion(*a:hasMaker a:Moto675SR a:Yamaha*)
a:isProducedBy	a:AutoEBB887	ObjectPropertyAssertion(*a:isProducedBy a:AutoEBB887 a:Toyota*)
	a:AutoDOT475	ObjectPropertyAssertion(*a:isProducedBy a:AutoDOT475 a:Hyundai*)
	a:AutoCKD843	ObjectPropertyAssertion(*a:isProducedBy a:AutoCKD843 a:Mitsubishi*)
	a:AutoGKT274	ObjectPropertyAssertion(*a:isProducedBy a:AutoGKT274 a:Nissan*)
	a:AutoRKU418	ObjectPropertyAssertion(*a:isProducedBy a:AutoRKU418 a:Toyota*)
a:isBrokerOf	a:BrokerKing	ObjectPropertyAssertion(*a:isBrokerOf a:BrokerKing a:ZurichConnect*)
a:isInsuredBy	a:AutoEBB887	ObjectPropertyAssertion(*a:isInsuredBy a:AutoEBB887 a:ZurichConnect*)
	a:AutoDOT475	ObjectPropertyAssertion(*a:isInsuredBy a:AutoDOT475 a:MotorDirect*)
	a:AutoCKD843	ObjectPropertyAssertion(*a:isInsuredBy a:AutoCKD843 a:BrokerKing*)
	a:AutoGKT274	ObjectPropertyAssertion(*a:isInsuredBy a:AutoGKT274 a:AutoTrader*)
	a:AutoRKU418	ObjectPropertyAssertion(*a:isInsuredBy a:AutoRKU418 a:DivaInsurance*)
a:isVerifiedBy	a:AutoEBB887	ObjectPropertyAssertion(*a:isVerifiedBy a:AutoEBB887 a:JohnMayer*)
	a:AutoDOT475	ObjectPropertyAssertion(*a:isVerifiedBy a:AutoDOT475 a:JohnMayer*)
	a:AutoCKD843	ObjectPropertyAssertion(*a:isVerifiedBy a:AutoCKD843 a:PeterIrving*)
	a:AutoGKT274	ObjectPropertyAssertion(*a:isVerifiedBy a:AutoGKT274 a:JohnMayer*)
	a:AutoRKU418	ObjectPropertyAssertion(*a:isVerifiedBy a:AutoRKU418 a:PeterIrving*)
a:previousOwner	a:AutoEBB887	ObjectPropertyAssertion(*a:previousOwner a:AutoEBB887 a:EricBrown*)
	a:AutoCKD843	ObjectPropertyAssertion(*a:previousOwner a:AutoCKD843 a:RobinStewart)*
a:registeredOn	a:AutoEBB887	ObjectPropertyAssertion(*a:registeredOn*

Object property	Individuals	Assertions
		a:AutoEBB887 a:AdamBradley)
	a:AutoDOT475	ObjectPropertyAssertion(*a:registeredOn a:AutoDOT475 a:EricBrown*)
	a:AutoCKD843	ObjectPropertyAssertion(*a:registeredOn a:AutoCKD843 a:EricBrown*)
	a:AutoGKT274	ObjectPropertyAssertion(*a:registeredOn a:AutoGKT274 a:RobinStewart*)
	a:AutoRKU418	ObjectPropertyAssertion(*a:registeredOn a:AutoRKU418 a:RobinStewart*)
a:assembledFrom	a:AutoRKU418	ObjectPropertyAssertion(*a:assembledFrom a:AutoRKU418 a:SteeringWheel*)
		ObjectPropertyAssertion(*a:assembledFrom a:AutoRKU418 a:Engine*)
		ObjectPropertyAssertion(*a:assembledFrom a:AutoRKU418 a:CylinderHead*)
		ObjectPropertyAssertion(*a:assembledFrom a:AutoRKU418 a:Distributor*)
		ObjectPropertyAssertion(*a:assembledFrom a:AutoRKU418 a:GearBox*)
a:trusts	a:ZurichConnect	ObjectPropertyAssertion(a:trusts a:ZurichConnect a:BrokerKing)
	a:DivaInsurance	ObjectPropertyAssertion(a:trusts a:DivaInsurance a:AutoTrader)

Lentelė 47Venchile ontologijos *OWL*2 individų duomenų savybių (assertion) aprašas

Data property	Individuals	Assertions
a:authorizedCapital	a:ZurichConnect	DataPropertyAssertion(*a:authorizedCapital a:ZurichConnect "84100000"^^xsd:float*)
	a:MotorDirect	DataPropertyAssertion(*a:authorizedCapital a:MotorDirect "5280000"^^xsd:float*)
	a:BrokerKing	DataPropertyAssertion(*a:authorizedCapital a:BrokerKing "3650000"^^xsd:float*)
	a:AutoTrader	DataPropertyAssertion(*a:authorizedCapital a:AutoTrader "120640,50"^^xsd:float*)
	a:DivaInsurance	DataPropertyAssertion(*a:authorizedCapital a:DivaInsurance "34800000"^^xsd:float*)
a:certificateNum	a:JohnMayer	DataPropertyAssertion(*a:certificateNum a:JohnMayer "AS-97-623-1"^^xsd:string*)
	a:PeterIrving	DataPropertyAssertion(*a:certificateNum a:PeterIrving "AS-48-122-1"^^xsd:string*)
a:vin	a:AutoEBB887	DataPropertyAssertion(*a:vin a:AutoEBB887 "4T1BE46K17U704216"^^xsd:string*)
	a:AutoDOT475	DataPropertyAssertion(*a:vin a:AutoDOT475 "KMHCN46C07U458487"^^xsd:string*)
	a:AutoCKD843	DataPropertyAssertion(*a:vin a:AutoCKD483 "JA32U8FW3BU028408"^^xsd:string*)
	a:AutoGKT274	DataPropertyAssertion(*a:vin a:AutoGKT274 "1N4BL21E67C170979"^^xsd:string*)
	a:AutoRKU418	DataPropertyAssertion(*a:vin a:AutoRKU418 "1NXBU4EE6AZ379083"^^xsd:string*)
	a:Moto568AE	DataPropertyAssertion(*a:vin a:Moto568AE "JKAEXMF166DA23239"^^xsd:string*)
	a:Moto968UH	DataPropertyAssertion(*a:vin a:Moto968UH "JKBZXNC15AA020752"^^xsd:string*)
	a:Moto675SR	DataPropertyAssertion(*a:vin a:Moto675SR "JYAVP11E48A106468"^^xsd:string*)
a:uniqueRegNum	a:Moto568AE	DataPropertyAssertion(*a:vin a:Moto568AE "JKAEXMF166DA23239"^^xsd:string*)
	a:Moto968UH	DataPropertyAssertion(*a:vin a:Moto968UH "JKBZXNC15AA020752"^^xsd:string*)

Data property	Individuals	Assertions
	a:Moto675SR	DataPropertyAssertion(*a:vin a:Moto675SR "JYAVP11E48A106468"^^xsd:string*)
a:contactPhone	a:JohnMayer	DataPropertyAssertion(*a:contactPhone a:JohnMayer "+37061256536"^^xsd:string*)
	a:PeterIrving	DataPropertyAssertion(*a:contactPhone a:PeterIrving "+37061198696"^^xsd:string*)
a:dateOfBirth	a:AdamBradley	DataPropertyAssertion(*a:dateOfBirth a:AdamBradley "1981-04-11"^^xsd:date*)
	a:EricBrown	DataPropertyAssertion(*a:dateOfBirth a:EricBrown "1972-01-14"^^xsd:date*)
	a:RobertStewart	DataPropertyAssertion(*a:dateOfBirth a:RobertStewart "1986-10-23"^^xsd:date*)
a:defectDescription	a:AutoDOT475	DataPropertyAssertion(*a:defectDescription a:AutoDOT475 "Motor defect"^^xsd:string*)
	a:AutoCKD843	DataPropertyAssertion(*a:defectDescription a:AutoCKD843 "Gearbox defect"^^xsd:string*)
a:industrialArea	a:Yamaha	DataPropertyAssertion(*a:industrialArea a:Yamaha "Motorbikes"^^xsd:string*)
	a:Kawasaki	DataPropertyAssertion(*a:industrialArea a:Kawasaki "Motorbikes"^^xsd:string*)
	a:Toyota	DataPropertyAssertion(*a:industrialArea a:Toyota "Automobiles"^^xsd:string*)
	a:Hyundai	DataPropertyAssertion(*a:industrialArea a:Hyundai "Automobiles"^^xsd:string*)
	a:Nissan	DataPropertyAssertion(*a:industrialArea a:Nissan "Automobiles"^^xsd:string*)
	a:Mitsubishi	DataPropertyAssertion(*a:industrialArea a:Mitsubishi "Automobiles"^^xsd:string*)
a:kilometrage	a:AutoEBB887	DataPropertyAssertion(*a:kilometrage a:AutoEBB887 "145846"^^xsd:integer*)
	a:AutoDOT475	DataPropertyAssertion(*a:kilometrage a:AutoDOT475 "126482"^^xsd:integer*)
	a:AutoCKD843	DataPropertyAssertion(*a:kilometrage a:AutoCKD483 "804965"^^xsd:integer*)
	a:AutoGKT274	DataPropertyAssertion(*a:kilometrage a:AutoGKT274 "240473"^^xsd:integer*)
	a:AutoRKU418	DataPropertyAssertion(*a:kilometrage a:AutoRKU418 "114726"^^xsd:integer*)
	a:Moto568AE	DataPropertyAssertion(*a:kilometrage a:Moto568AE "22473"^^xsd:integer*)
	a:Moto968UH	DataPropertyAssertion(*a:kilometrage a:Moto968UH "27912"^^xsd:integer*)
	a:Moto675SR	DataPropertyAssertion(*a:kilometrage a:Moto675SR ""46471^^xsd:integer*)
a:licenseValFrom	a:AdamBradley	DataPropertyAssertion(*a:licenseValFrom a:AdamBradley "1999-05-12"^^xsd:date*)
	a:EricBrown	DataPropertyAssertion(*a:licenseValFrom a:EricBrown "1991-02-24"^^xsd:date*)
	a:RobertStewart	DataPropertyAssertion(*a:licenseValFrom a:RobertStewart "2004-11-15"^^xsd:date*)
a:modelTitle	a:AutoEBB887	DataPropertyAssertion(*a:modelTitle a:AutoEBB887 "Avensis"^^xsd:string*)
	a:AutoDOT475	DataPropertyAssertion(*a:modelTitle a:AutoDOT475 "Sonata"^^xsd:string*)
	a:AutoCKD843	DataPropertyAssertion(*a:modelTitle a:AutoCKD483 "Lancer"^^xsd:string*)
	a:AutoGKT274	DataPropertyAssertion(*a:modelTitle a:AutoGKT274 "Primera"^^xsd:string*)
	a:AutoRKU418	DataPropertyAssertion(*a:modelTitle a:AutoRKU418 "Corolla"^^xsd:string*)

Data property	Individuals	Assertions
a:personalCode	a:JohnMayer	DataPropertyAssertion(*a:personalCode a:JohnMayer "37405087548"^^xsd:string*)
	a:PeterIrving	DataPropertyAssertion(*a:personalCode a:PeterIrving "38304232165"^^xsd:string*)
	a:AdamBradley	DataPropertyAssertion(*a:personalCode a:AdamBradley "38104110635"^^xsd:string*)
	a:EricBrown	DataPropertyAssertion(*a:personalCode a:EricBrown "37201144521"^^xsd:string*)
	a:RobertStewart	DataPropertyAssertion(*a:personalCode a:RobertStewart "38610238745"^^xsd:string*)
a:produceYear	a:AutoEBB887	DataPropertyAssertion(*a:produceYear a:AutoEBB887 "2006"^^xsd:date*)
	a:AutoDOT475	DataPropertyAssertion(*a: produceYear a:AutoDOT475 "2007"^^xsd:date*)
	a:AutoCKD843	DataPropertyAssertion(*a: produceYear a:AutoCKD483 "2008"^^xsd:date*)
	a:AutoGKT274	DataPropertyAssertion(*a: produceYear a:AutoGKT274 "2006"^^xsd:date*)
	a:AutoRKU418	DataPropertyAssertion(*a: produceYear a:AutoRKU418 "2006"^^xsd:date*)
a:travelBags	a:Moto568AE	DataPropertyAssertion(*a:travelBags a:Moto568AE "0"^^xsd:integer*)
	a:Moto968UH	DataPropertyAssertion(*a:travelBags a:Moto968UH "0"^^xsd:integer*)
	a:Moto675SR	DataPropertyAssertion(*a:travelBags a:Moto675SR "2"^^xsd:integer*)
a:withCradle	a:Moto568AE	DataPropertyAssertion(*a:withCradle a:Moto568AE "0"^^xsd:byte*)
	a:Moto968UH	DataPropertyAssertion(*a:withCradle a:Moto968UH "0"^^xsd:byte*)
	a:Moto675SR	DataPropertyAssertion(*a:withCradle a:Moto675SR "1"^^xsd:byte*)