KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Tomas Trainys

Ontologijomis grindžiamas reliacinių duomenų bazių integravimas

Magistro darbas

Darbo vadovas
prof. Rimantas Butleris

Konsultantė
dokt. Virginija Uzdanavičiūtė

Kaunas, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Tomas Trainys

Ontologijomis grindžiamas reliacinių duomenų bazių integravimas

Magistro darbas

Recenzentas
doc. Dalius Makackas
2013-05-

Darbo vadovas
prof. Rimantas Butleris
2013-05-

Konsultantė
dokt. Virginija Uzdanavičiūtė
2013-05-

Darbo autorius
IFM-1/4 gr. stud.
Tomas Trainys
2013-05-

Kaunas, 2013

Turinys

Santrauka

Ontologijomis grindžiamas reliacinių duomenų bazių integravimas

Kuriant naujas ar plečiant esamas informacijos sistemas susiduriama su heterogeninių duomenų bazių jungimo, integravimo bei duomenų pakartotinio panaudojimo problemomis. Yra atlikta tyrimų, eksperimentų, sukurta programinių prototipų, tačiau integravimo uždavinys vis dar iki galo nepritaikytas / neišspręstas.

Šio darbo idėja – integruoti heterogenius duomenų šaltinius, taikant semantinio tinklo metodus ir paruošti toliau naudoti. Tam tikslui, sudaryti duomenų integravimo metodiką ir procesą, taip pat programinį prototipą metodikos efektyvumui patikrinti.

Konceptualaus integruojamų schemų suderinamumo modeliui įgyvendinti taikomos ontologijas, nes jos skatina dalytis žiniomis. OWL modelis pasižymi išraiškingumu, yra standartizuotas (W3C rekomendacija), vienareikšmiškai suprantamas, formalus, jam interpretuoti yra sukurta daug programinės įrangos.

Darbe aprašoma metodika, kaip integruoti duomenų šaltinius taikant ontologijas. Išanalizuotos kalbos ontologijoms sudaryti, aptartos galimos integravimo architektūros, integravimo procesas bei semantiniai konfliktai ir jų sprendimo būdai.

Sudarytos metodikos efektyvumui patikrinti, sudarytas semantinis modelis, taikant globalios schemos metodą, išspręsti semantiniai konfliktai tarp skirtingų duomenų šaltinių schemų. Sukurtas .NET platformos pagrindu programinis prototipas, kuris leidžia įdiegti modelį bei siųsti semantinės užklausas į ontologijoje aprašytus duomenų šaltinius.

Summary

Ontologies based integration of relational databases

The construction of new or expansion of existing information systems encountered heterogeneous database connection, integration, and data re-use problems. There are a number of research, experiment, create software prototypes, however, the integration task is still not completely not adapted / unresolved.

This work was the idea to integrate the heterogeneous data sources using Semantic Web techniques and prepared for later use. For this purpose, compose a data integration methodology and process, as well as software prototype for methodology verification.

Conceptual model integrating the compatibility of schemes to implement ontologies apply because they promote sharing of knowledge. Model has the expressiveness of OWL, is a standardized (W3C Recommendation), clearly understood, formal, a lot of software is designed to interpret it.

The paper describes the techniques for integrating data sources using ontologies. The analysis of language ontologies conclude discuss possible integration of architecture, the process of integration and semantic conflicts and their solutions.

For efficiency verification of methodology created semantic model, the global schemes applying the method to solve semantic conflicts between different sources of data schemes. Created. NET platform software prototype, which enables to install the semantic model, and send queries to the ontology described in the data sources.

Terminų ir santrumpų žodynas

Terminas / santrumpa	Paaiškinimas
OWL	Ontologijų užrašymo kalba.
RDF	Resursų aprašymo karkasas.
RDFS	Resursų aprašymo karkaso schema.
XML	Žymų kalbą.
SOA	Paslaugų teikiamų internetu architektūra.
SWS	Semantinėmis technologijomis grįstas internetu teikiamų paslaugų servisas.
Ontologija	Dalykinės srities konceptualus modelis.
SPARQL	Kalba skirta informacijos išgavimui iš *RDF* duomenų rinkinių.
DBVS	Duomenų bazių valdymo sistema.
SQL	Struktūrizuota užklausų kalba.
Semantinis konfliktas	Konceptų nesutapimas.
Heterogeniškumas	Duomenų modelių, kalbų, techninės įrangos skirtumas.
Konceptas	Realiame pasaulyje egzistuojantis objektas užrašytas išreikštu pavidalu. Objektiniu požiūriu konceptas atitinka klasės sąvoką.
ER	Esybių ryšių modelis.
ETL	Duomenų gavybos operacijos (ang. *extract, transform, load*)
API	Taikomųjų programų programavimo sąsaja.

1. Įvadas

Per pastaruosius kelis dešimtmečius įmonės duomenims saugoti sėkmingai pritaikė įvairias duomenų bazių valdymo sistemas (DBVS). Plačiai paplito įvairių gamintojų *DBVS – MS SQL Server, Oracle, MySQL* ir kt. Istoriškai susiklostė, kad pirmiausia duomenų bazės (DB) buvo diegiamos laikantis decentralizuoto požiūrio, o įmonėms plečiantis, didėjant padalinių skaičiui ir jungiantis į didesnes korporacijas bei sindikatus, atsirado poreikis įmonių duomenis integruoti ir paruošti toliau naudoti. Integravimo klausimas tapo svarbus daugiau nei prieš trisdešimt metų ir iki šiol yra aktualus.

Reikia paminėti, kad pastaruoju metu dideliu tempu besikeičiant technologinei aplinkai bei plečiantis ir bendradarbiaujant įmonėms duomenų integracijos klausimas išlieka aktualus ir turi įtakos galutinei kuriamo produkto kainai. Be to, atsižvelgiant į didėjančią telekomunikacinių tinklų spartą, kuri globaliuose ir vietiniuose kompiuterių tinkluose gali siekti iki kelių GB/s, ir įvertinant šiuo metu eksploatuojamos techninės įrangos našumą (procesorinę galią), taip pat atminčių spartą, galima teigti, kad techninės įrangos sluoksnis yra pakankamai išvystytas didėlės apimties heterogeniniams duomenų šaltiniams integruoti. Todėl yra racionalu geografiškai paskirstytus duomenų šaltinius integruoti, tam pritaikant centralizuotos duomenų bazės (dar vadinamos federaline duomenų baze) metodus. Toks sprendimas leistų sumažinti *DBVS* administravimo kainą ir pagreitinti taikomosios programinės įrangos kūrimo procesą.

Centralizuotą duomenų bazę galima apibūdinti kaip sistemą, suteikiančią naudotojams galimybę, naudojant *SQL / SPARQL* užklausas, manipuliuoti geografiškai paskirstytų duomenų šaltinių duomenimis. Yra galimybių manipuliuoti ir heterogeninių duomenų šaltinių duomenimis, tačiau dažniausiai susiduriama su struktūriniais ir semantiniais konfliktais duomenų šaltinių schemos bei duomenų lygmenyse.

Heterogeniškumą galima apibūdinti keliais aspektais [13]: *struktūrinis* – apima skirtingus duomenų modelius; *sintaksinis* – apima skirtingas kalbas ir duomenų vaizdavimą; *sisteminis* – apima operacines sistemas ir techninę įrangą; *semantinis* – apima semantinį duomenų modelių suderinamumą.

Darbo tikslas yra ištirti galimybes integruoti heterogenius duomenų šaltinius, taikant semantinio tinklo metodus ir paruošti toliau naudoti. Tam tikslui, atlikti egzistuojančių sprendimų, metodų ir priemonių analizę, sudaryti duomenų integravimo metodiką ir procesą, taip pat sukurti programinį prototipą metodikos efektyvumui patikrinti.

Darbas apima tik semantinio suderinamumo klausimus heterogeninių duomenų šaltinių integravimo procese. Aptariami semantinio suderinamumo uždavinių sprendimo būdai,

heterogeniniams duomenų šaltiniams taikant ontologijas, kaip aukštesnio lygmens duomenų šaltinių aprašymo modelį. Plačiai nagrinėjamas ontologijų pagrindu vykdomas integravimo procesas, semantiniai konfliktai ir jų sprendimo būdai taikant *OWL* kalbą, aptariamos architektūros ir metodai, leidžiantys užtikrinti duomenų bazių homogeniškumą bei informacijos integralumą.

Projekto dalyje aprašoma kaip integruoti duomenų šaltinius taikant ontologijas ir įrankiai procesui įgyvendinti, pateikiama sprendimo metodika ir integravimo sistemos architektūros prototipas.

Sudaryta metodika, procesas ir prototipas gali būti taikomi mažų ir vidutinių įmonių heterogeniniams duomenų šaltiniams integruoti.

Darbe išskiriami šie naujumo aspektai: sudaryta metodika Microsoft technologiniams sprendimams; sudarytas mažai žmogiškų išteklių reikalaujantis integravimo procesas; įvertintas integruojamų duomenų šaltinių integracijos efektyvumas, tam palygintas *SQL* ir *SPARQL* užklausų siuntimo laikas.

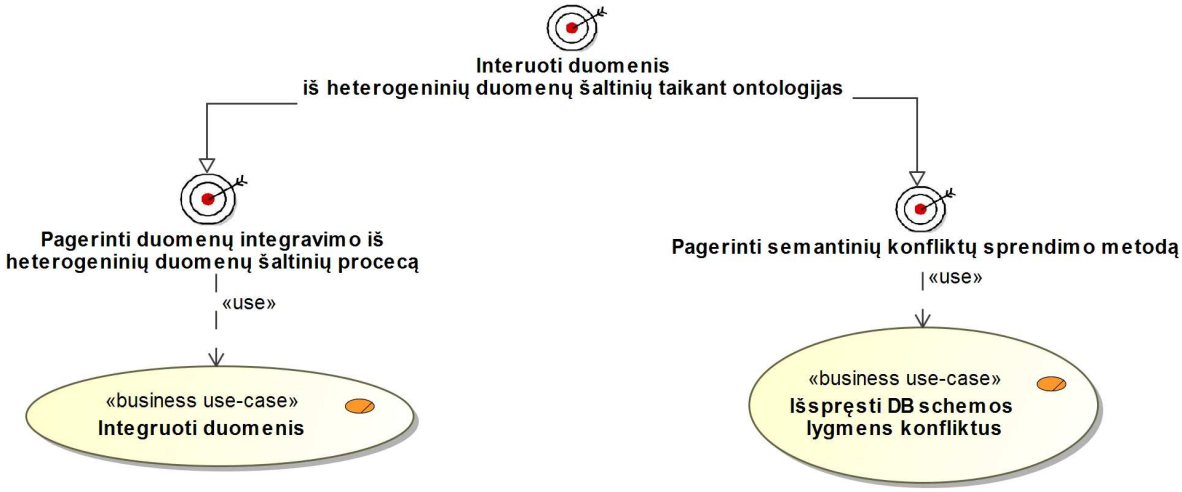
Darbo pabaigoje pateikiamos išvados ir rekomendacijos duomenų šaltinių integracijos ir sudarytos metodikos klausimais. Taip pat pateikiamos nuorodos į naudotos literatūros šaltinius ir priedai. Priedų skyriuje pateikiamas aštuonioliktosios tarpuniversitetinės tarptautinės magistrantų ir doktorantų konferencijoje „Informacinė visuomenė ir universitetinės studijos“ (IVUS 2013) pristatytas straipsnis „Reliacinių duomenų bazių duomenų integravimas taikant ontologijas“.

2. Ontologijomis grindžiamų reliacinių duomenų bazių integravimo analizė

2.1. Analizės tikslas

Dalykinės srities veiklos analizės tikslas – ištirti kompiuterizuojamos srities poreikius ir problemas bei taikant aukštą abstrakcijos lygį ir tam pasirinktą metodą ir priemones juos aprašyti.

Analizės potiksliai – išskirti srities elementus, nustatyti kompiuterizavimo poreikius ir galimybes bei jų tikslingumą, sudaryti dalykinės srities modelį, nustatyti procesus, aprašyti aktorių vaidmenis ir ribas, sudaryti reikalavimus kuriamai sistemai ir duomenų modeliu, nustatyti kuriamos sistemos ribas. Parinkti tinkamus metodus ir priemones analizei atlikti. Tiriamos dalykinės srities tikslų diagrama pateikta 1 pav.



1 pav. Veiklos tikslų modelis

2.1.1. Analizės metodų parinkimas

Programinės įrangos ir informacinių sistemų analizei ir projektavimui taikomi metodai, tokie kaip *UP*, *F3*, *ICONIC* ir kt. Kurie pasirenkami atsižvelgiant į projekto pobūdį, dalykinę sritį bei kitus faktorius.

UP (angl. *Unified Process*) – tai programinės įrangos kūrimo proceso karkasas, tinkantis įvairioms taikymo sritims. Metodas yra komponentais grindžiamas, apima programinės įrangos architektūrą, panaudojimą, funkcionalumą, turi pakartotinio panaudojimo galimybes, yra panaudojimo atvejų valdomas iteracinis procesas.

F3 (angl. *From fuzzy to formal*) – tai metodika, kuri skirta padėti išgauti kuo tikslesnę kuriamos sistemos specifikaciją, sudaryti reikalavimus, išsiaiškinti dalykinės srities poreikius ir problemas.

ICONIC – programinės įrangos kūrimo procesas. Procesas panašus į klasikinį krioklio procesą, tačiau neapima visos kuriamos sistemos gyvavimo ciklo. Pagrindinis akcentas yra tarp panaudojimo atvejų ir programinio kodo. Metodų palyginimas pateiktas 1 lentelėje.

Kūrimo etapas	Metodika		
	UP	F3	ICONIC
Dalykinės srities veiklos analizė	Apima	Apima	Pritaikomas
Sistemos analizė	Apima	Apima	Apima
Projektavimas	Apima	Apima	Apima
Realizavimas	Apima	Galimas prototipų kūrimas	Apima
Testavimas	Apima	Neapima	Neapima

1 lentelė. Informacijos sistemų kūrimo metodų palyginimas

Atlikus metodų palyginimą dalykinės srities analizės vykdymui pasirinktas *UP* metodas, kadangi jis apima visus sistemų kūrimo etapus.

2.2. Tyrimo sritis, objektas ir problema

Tyrimo objektas – ontologijomis grindžiamas reliacinių duomenų bazių integravimo procesas.

Tyrimo sritis – duomenų bazių integravimo metodai, modeliai, kalbos ontologijoms sudaryti, semantiniai konfliktai.

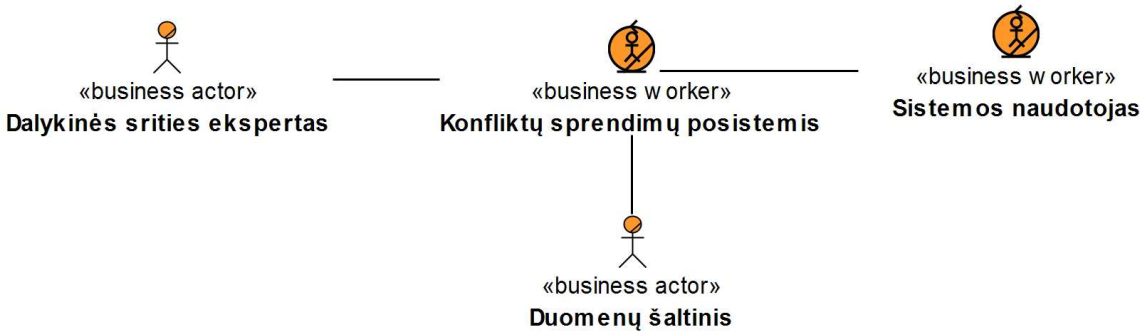
Tyrimo problema – integruojant skirtingų duomenų bazių duomenis, susiduriama su semantiniais konfliktais schemos ir duomenų lygmenyse, todėl iškyla poreikis nustatyti semantinius konfliktus tarp integruojamų duomenų bazių schemų elementų ir išspręsti. Pasiekus schematinį suderinamumą, tarp skirtingų duomenų bazių, sprendimas leis sudaryti bendrą duomenų modelį ir automatizuotai sujungti keletą duomenų bazių į vieną.

2.3. Duomenų integravimo iš reliacinių duomenų bazių sistemos analizė

Analizės metu, naudojant *UP* metodiką buvo sumodeliuotas sistemos kontekstas bei funkcionavimo ribos, išskirti sistemos aktoriai bei aprašyta bendra sistemos elgsena.

2.3.1. Veiklos kontekstas

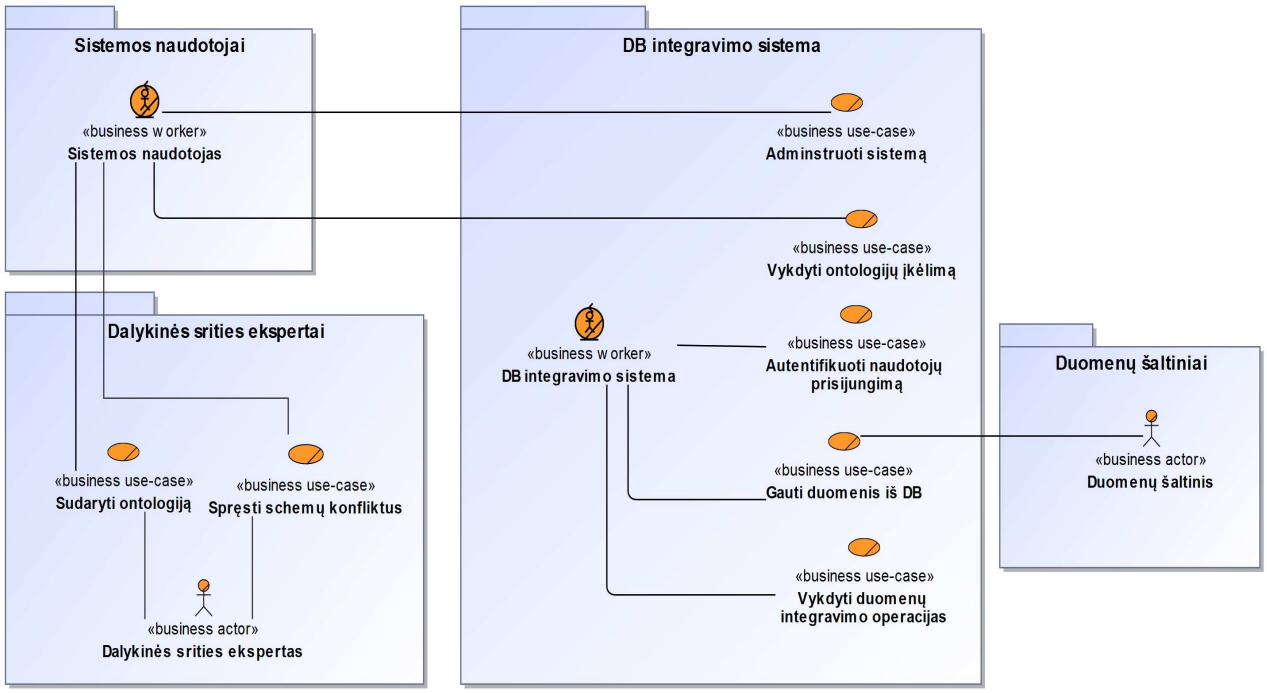
Dalykinės srities veiklos kontekstas pavaizduotas aukščiausio lygio kontekstine diagrama (2 pav.) Dalykinės srityje yra dviejų tipų dalyviai, tai vidiniai ir išoriniai. Išoriniai dalyviai yra dalykinės srities ekspertas ir duomenų šaltiniai, tačiau šie aktoriai, priklausomai nuo sistemos panaudojimo organizacijoje gali būtų ir vidiniais dalyviais. Vidiniai sistemos aktoriai – veiklos darbuotojai t.y. sistemos naudotojas ir *DB* integravimo sistema, kurie tiesiogiai vykdo duomenų integravimo operacijas.



2 pav. Veiklos konteksto diagrama

2.3.2. Veiklos sąveikų modelis

Veiklos sąveikų modelyje vaizduojami sistemos aktorių veiksmai, jų vykdomi panaudojimo atvejai atitinkamose sistemos konteksto ribose (3 pav.). Sistemos naudotojai gali būti kelių tipų, sistemą administruojantis bei jos naudotojai, tai išsprendžiama per sistemos funkcionalumo apribojimus.

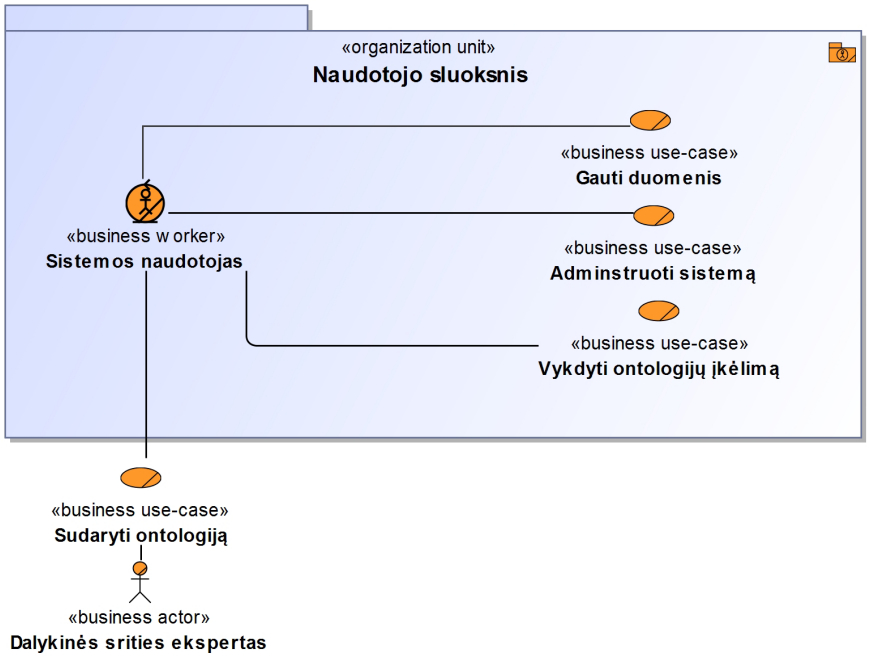


3 pav. „Veiklos sąveikų modelis“

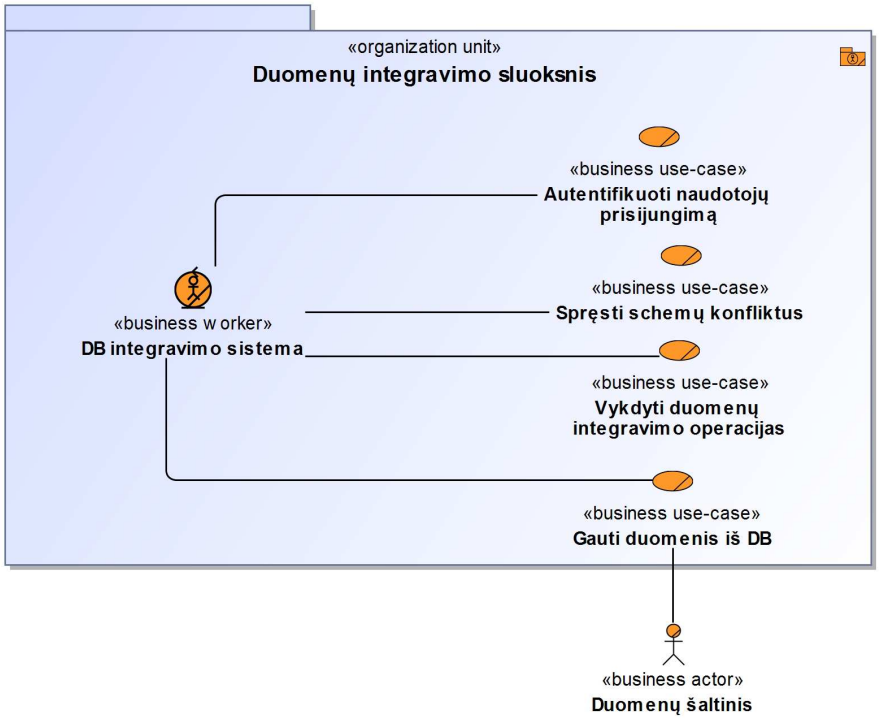
2.3.3. Veiklos panaudojimo atvejai

Sistema suskirstyta į du sluoksnius, kuriuose aprašomi pradiniai sistemos funkcionalumo panaudojimo atvejai, kurie yra išreikšti per sistemos aktorių vykdomas veiklas. Naudotojo sluoksnyje, sistemos naudotojas gali atlikti sistemos nustatymus bei įkelti dalykinės srities eksperto sudarytą dalykinės srities ontologiją (4 pav.). Sluoksnis skirtas sistemos nustatymams atlikti, suderinti duomenų šaltinius ir gauti duomenis iš jų.

Duomenų integravimo sluoksnyje (5 pav.) vykdomos duomenų surinkimo ir integravimo operacijos.



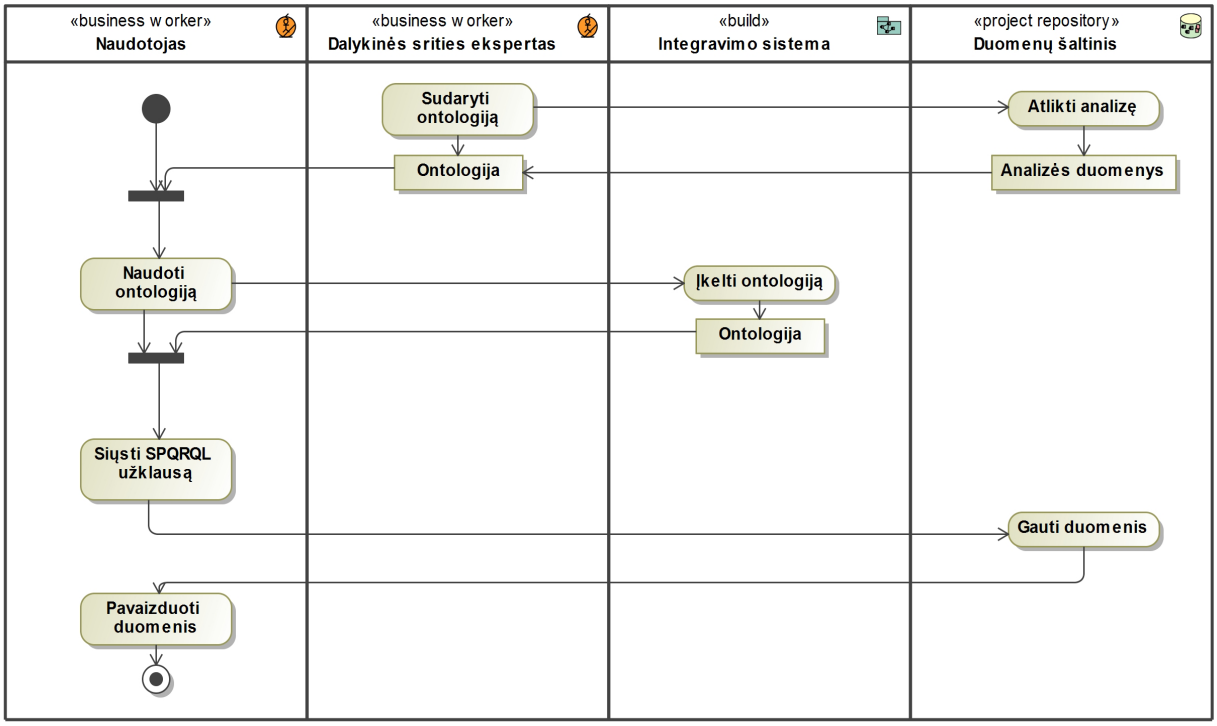
4 pav. Naudotojo sluoksnio veiklos panaudojimo atvejai



5 pav. Duomenų integravimo sluoksnio veiklos panaudojimo atvejai

2.3.4. Veiklos procesų modelis

Tiriamosios srities aktorių ir aprašytų sistemos sluoksnių dinamika atvaizduoti UML veiklos diagrama (6 pav.). Diagramoje vaizduojamos pagrindinės sistemos aktorių vykdomos funkcijos ir sistemos atliekamos operacijos bei jų sužadinimo sąlygos.



6 pav. Veiklos proceso modelis

2.4. Vartotojų analizė

2.4.1. Vartotojų aibė, tipai ir savybės

Orientuojantis į tai, kad duomenys bus integruojami naudojant ontologijų modelius, vienas vartotojų tipas turėtų būti probleminės srities žinovas, kitas vartotojo tipas gali būti pažengęs vartotojas, susipažinęs su duomenų bazių pagrindais (2 lentelė).

Vartotojai	Tipai, savybės
Informacinių sistemų specialistai	programuotojai, duomenų centrų aptarnaujantis personalas, duomenų bazių sistemų architektai, inžinieriai, duomenų bazių administratoriai
Informacinių sistemų naudotojai	inžinieriai, duomenų bazių administratoriai, dalykinės srities ekspertai

2 lentelė. Vartotojų tipai

2.4.2. Vartotojų tikslai ir problemos

Didelėse įmonėse, dažnai skirtingi padaliniai naudoja savo padalinių duomenų bazes, tačiau dėl įvairių priežasčių (statistika, apibendrinimas ir kitų) vienam iš padalinių gali prireikti bendros duomenų bazių aibės informacijos. Taip pat atitinkamos informacijos gali būti aktuali ir skirtingomis laiko perspektyvomis. Didelių informacijos kiekių saugojimas skirtingose duomenų bazėse reikalauja papildomų finansinių ir žmogiškųjų resursų. Vienas iš sprendimų gali būti, duomenų bazių apjungimas į vieną centrinę (federacinę). Apjungimo uždaviniai gali

būti sprendžiami naudojant tarpinį sluoksnį bei metodus reikiamus duomenų integravimui, kurie užtikrintų duomenų bazių vienalytiškumą bei informacijos integralumą.

Problema	Aprašymas
Integruojant duomenų bazes susiduriama su semantiniais konfliktais	Integruojant duomenis iš skirtingų šaltinių reikia išspręsti schemos bei duomenų lygio konfliktus.
Ilgalaikis informacijos saugojimas reikalauja finansinių ir žmoniškųjų resursų	Norint užtikrinti patikimą, ilgalaikį informacijos saugojimas, reikia tam numatyti technologinius procesus bei skirti fizinius resursus, tokių funkcijų vykdymas reikalauja papildomų žmoniškųjų resursų, kas mažom įmonėm nėra naudinga.

3 lentelė. Vartotojų problemos

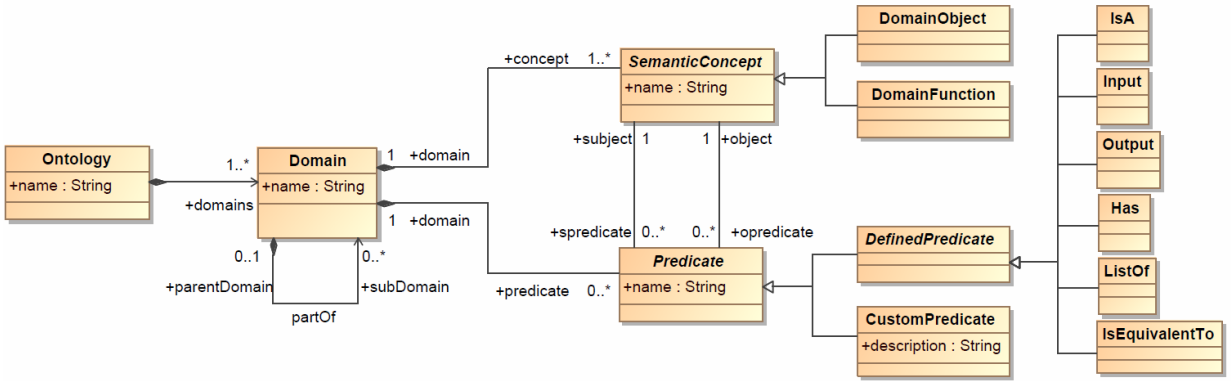
2.5. Problemos sprendimo metodų literatūros šaltiniuose analizė

2.5.1. Heterogeninių duomenų bazių integravimo apžvalga

Heterogeniškumą galima apibūdinti keliais aspektais [13]: *struktūrinis* – apima skirtingus duomenų modelius; *sintaksinis* – apima skirtingas kalbas ir duomenų vaizdavimą; *sisteminis* – apima operacines sistemas ir techninę įrangą; *semantinis* – apima semantinį duomenų modelių suderinamumą.

2.5.2. Ontologijos ir jų aprašymo kalbos

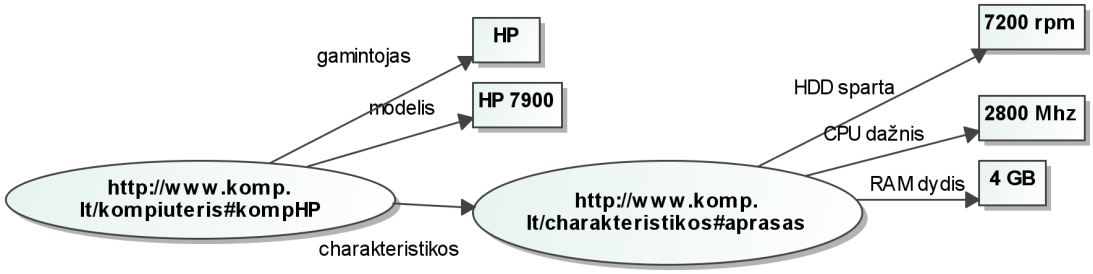
„Ontologijos“ sąvoka informatikos srityje vartojama dalykinės srities konceptų specifikacijai apibūdinti [12]. Ontologija – aukšto semantinio lygmens dalykinės srities aprašas, kuris susideda iš dalykinės srities *konceptų, ryšių tarp konceptų, aksiomų ir apribojimų*. Tai yra duomenų modelis (7 pav.), apimantis klases, objektus, jų savybes ir ryšius [11]. Modelis suteikia galimybę detaliau nei *ER* modelis (angl. *entity relationship*) konceptualizuoti dalykinę sritį, sudaryti klasių hierarchiją bei sukurti ryšius tarp skirtingų dalykinių sričių. Vienas iš reikšmingų ontologijos modelio privalumų – išraiškingumas, žmogui ir kompiuterių sistemai suprantamas, teikiantis galimybę sudaryti dalykinių sričių žodynus.



7 pav. Ontologijos metamodelis

Formaliai ontologiją *O* galima aprašyti grafu [10] *G*, turinčiu *Mo* baigtinę mazgų aibę ir baigtinę briaunų aibę *Bo*. Briaunos gali būti aprašomos trejetu m_1, m_2, p_1, kai m_1, m_2 yra *Mo* konceptai, o p_1 yra briaunos pavadinimas. Formaliai ontologijos koncepcinį duomenų modelį galima užrašyti $G_O = (M_O, G_O)$.

Ontologijoms sudaryti yra naudojami kelių kalbų poaibiai: *RDF*, *RDF shema* (angl. *resource description framework*). Tai *XML* pagrindu sukurtas duomenų modelis [12], [11]. Šią technologiją dar galima apibūdinti kaip modelį išplečiantį *XML* galimybes, kuriuo aprašomi faktai apie resursus. Ši kalba sintaksę paveldėjo iš *XML*, tačiau *XML* nėra būtinas *RDF* modelio komponentas. *RDF* modelis sudarytas iš trijų pagrindinių dalių: *subjekto* – bet kokio daikto, aprašyto *RDF*; *predikato – savybės*, charakteristikos, atributo ar ryšio, skirto ištekliui apibūdinti; *objekto – savybės reikšmės*, tai gali būti kitas išteklius arba duomenų tipo reikšmė, kurie kartu vadinami teiginiu. *RDF* grafo pavyzdys pateiktas 8 pav., jo tekstinis aprašas – 4 lentelėje.



8 pav. RDF grafas

Subjektas	Predikatas	Objektas
http://www.komp.lt/kompiuteris#kompHP	Gamintojas	HP
http://www.komp.lt/kompiuteris#kompHP	Modelis	HP 7900
http://www.komp.lt/kompiuteris#kompHP	Charakteristikos	http://www.komp.lt/kompiuteris#aprasas
http://www.komp.lt/kompiuteris#aprasas	HDD sparta	7200 rpm
http://www.komp.lt/kompiuteris#aprasas	CPU dažnis	2800 Mhz

4 lentelė. RDF grafo aprašas

OWL (angl. *web ontology language*) – *RDF* ir *RDF-S* pagrindu sukurta ontologijų aprašymo kalba. *OWL* ontologija susideda iš konceptų ir ryšių tarp jų, apribojimo ir aksiomų. Toliau detaliai aptarsime *OWL* paketo struktūrą: pagrindinis *OWL* konstrukcinis elementas yra klasė, kalboje ji aprašoma *owl:Class* elementu. *OWL* klasė teikia galimybę grupuoti panašias charakteristikas turinčius išteklius. Savybe *rdfs:subClaccOf* yra aprašomi ryšiai tarp klasių, o savybė *rdfs:subPropertyOf* naudojama klasių hierarchijai sudaryti.

Ekvivalenčios klasės nustatomos naudojant konstrukcinį elementą *owl:equivalentClass*. Tuo pažymima, kad keli ištekliai (klasės, egzemplioriai, savybės) aprašo tą patį objektą. *OWL* klasė yra asocijuojama su klasės egzemplioriais, kurie yra vadinami klasės išplėtimais. Klasių egzemplioriai yra išvardijami naudojant konstrukcinį elementą *owl:oneOf*. Pagrindinės kalbos savybės yra: *owl:DatatypeProperty* ir *owl:ObjectProperty*, šiomis savybėmis yra sudaromi ryšiai tarp klasės egzempliorių bei priskiriami duomenų tipai.

OWL kalba yra aprašomos specifinės savybių charakteristikos, tokios kaip tranzityvumas, simetriškumas, inversija ir funkcinė priklausomybė. Šios charakteristikos aprašomos naudojant: *owl:SymmetricProperty, owl:InverseFunctionalProperty, owl:FunctionalProperty* ir *owl:TransitiveProperty* kalbos elementus.

Ryšių kardinalumas aprašomas naudojant elementus: *owl:cardinality, owl:minCardinality* ir *owl:maxCardinality.*

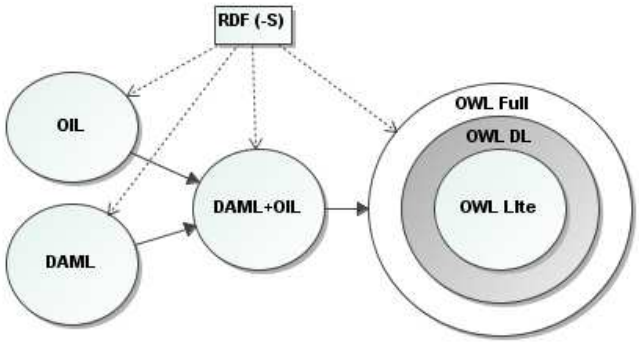
Sudėtinės klasės sudaromos naudojant savybes: *owl:intersectionOf, owl:unionOf* ir *owl:complementOf,* kuriomis vykdant sankirtos, sąjungos ir papildymo operacijas gaunamos sudėtinės klasės (dar vadinamos anoniminėmis klasėmis).

Pagrindinis kalbos teikiamas privalumas – išraiškingumas sudarant ryšius tarp klasių, ypatybių ir egzempliorių. Pagal išraiškingumą *OWL* ontologijos skirstomos į tris poaibius [12] (9 pav.):

OWL Full apima visus *OWL* kalbos elementus. Šio išraiškingumo kalba leidžia apjungti *OWL* su *RDF* ir *RDF-S* bei atlikti OWL dokumento konvertavimą į *RDF* dokumentą. Bet kuris *RDF* dokumentas sintaksiškai ir semantiškai yra ir *OWL* dokumentas.

OWL DL poaibiu užtikrinamas *RDF* ir *OWL* konstrukcinių elementų naudojimo teisingumas. Šio poaibio savybėmis yra vykdomas ontologijos aprašo struktūrizavimą, sudaromi ribojimai (pvz., klasių atskyrimą nuo egzempliorių – klasė negali būti egzemplioriumi ir klase). *OWL DL* yra grindžiama apibūdinimo logika (ang. *description logics*).

OWL Lite – tai skurdesnio ekspresyvumo *OWL* poaibis, teikiantis tik pagrindines *OWL* ypatybes, leidžiančias sudaryti klasių hierarchiją ir pagrindinius ribojimus. Turi mažesnės galios kardinalumo ribojimus (0 arba 1). Šio poaibio ontologijos turi mažesnį formalų sudėtingumą, yra lengviau įgyvendinamos.



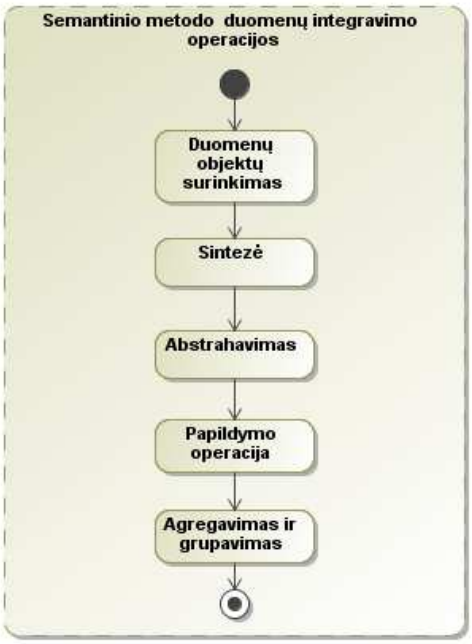
9 pav. OWL prototipai ir poaibiai

SPARQL (angl. *simple protocol and RDF query language*) – tai užklausų formulavimo kalba [22], skirta duomenų gavimui iš *RDF* grafo poaibių bei naujų RDF grafų formavimui. Pagrindiniai kalbos operatoriai: *SELECT* – grąžina rezultatus (pagal užklausoje pateiktus kintamuosius); *ASK* – identifikuoja informacijos egzistavimą; *DESCRIBE* – pateikia išteklius aprašantį *RDF* grafą; *CONSTRUCT* – naudojamas *RDF* grafui konstruoti.

2.5.3. Semantinės duomenų integravimo operacijos

Šiame poskyryje aptarsime duomenų integravimo operacijas. Priklausomai nuo pasirinktam integravimo metodui – pasirenkamas ir integracijos procesas ir operacijos. Kai kuriose situacijose, duomenys iš duomenų šaltinių yra surenkami, jų nekeičiant (nemodifikuojant), o kai kuriose situacijose yra vykdomos aukštesnio lygio semantinis integravimas, pvz., duomenų / schemos sujungimas (angl. *fusion*) be duomenų / schemos. Toliau detaliai aptarsime pagrindines operacijas. [3], (10 pav.). *Surinkimas.* Šios operacijos metu duomenų objektai yra surenkami, bet nekeičiami. Šioje operacijoje nevykdomos, lygiareikšmių duomenų objektų, gaunamų iš skirtingų šaltinių, lyginimo operacijos [3].

Sintezė (angl. *fusion*) šios operacijos metu, vykdomas duomenų objektų išskyrimu (angl. *extraction*), neatliekant jokių specialių skaičiavimų. Priešingai, nei duomenų surinkimo operacija, duomenų objektų sintezė vykdoma kartu su semantiškai lygiaverčiais duomenų objektais, gaunamais iš skirtingų duomenų šaltinių. Be to, sintezės proceso yra nustatomos duomenų reikšmės, pvz., jei iš duomenų šaltinių gaunamos prieštaraujančios tų pačių duomenų elementų vertės, tada duomenų konfliktų šalinimui – gali būti naudojamos įvairius, tokie kaip vaizdavimo (angl. *mapping*) taisyklių, euristiniai, dažnumo įverčio algoritmai ir kiti metodai. Literatūroje minima, kad integravimo procese, duomenų sintezės operacijos yra labai komplikuotas uždavinys, dažnai yra sudėtinga nustatysi / nuspręsti kuri duomenų reikšmė yra teisinga [3].



10 pav. Semantinio duomenų integravimo metodo operacijos

Abstrahavimas (sąvokų išskyrimas). Šios operacijos proceso metu, duomenų šaltiniai gali būti transformuojami, kad atitiktų reikiamą abstrakcijos laipsnį. Abstrahavimas, dažniausiai apima duomenų agregavimo ir esybių perskirstymo funkcijas, tačiau gali apimti ir daug sudėtingesnes operacijas [3].

Papildymo operacija (angl. *supplementation*). Kadangi iš duomenų šaltinių, nėra gaunami vien tik duomenys, bet pridedami ir metaduomenys, kurie apibūdina duomenų objekto turinį arba kontekstą, pvz., semantiniai metaduomenys. [3].

Ši integracijos operacija yra naudojama, numanomų duomenų objektų semantikos valdymui. Operacija yra būtina, kai duomenų šaltiniai nepateikia schemos ir integravimas vykdomas schemos metaduomenų pagrindu.

Agregavimas ir grupavimas. Tai duomenų apdorojimo operacijos, kurių metu vykdomos objektų pergrupavimo ir sujungimo į didesnius masyvus operacijos [3].

2.6. Architektūros ir galimų įgyvendinimo priemonių analizė

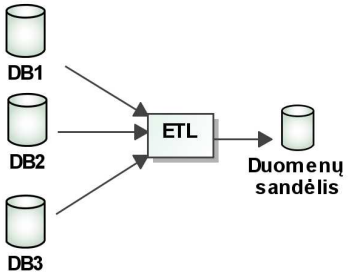
2.6.1. Duomenų šaltinių integravimo architektūros ir metodai

Duomenų, priklausančių skirtingiems duomenų šaltiniams, surinkimo, integravimo bei pateikimo uždaviniai gali būti sprendžiami taikant tradicinius metodus, tokius kaip duomenų sandėlių ir federalinių duomenų bazių [7], [13], [17] bei ontologijomis grindžiamus metodus. Toliau juos aptarsime detaliau.

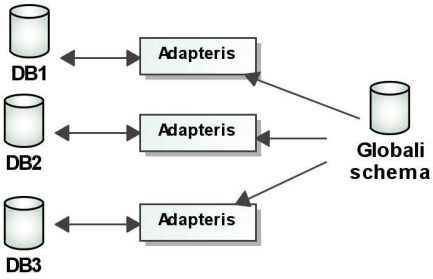
2.6.1.1. Tradiciniai metodai

Duomenų sandėlio (angl. *warehouse*) architektūros (12 pav.) atveju duomenys iš duomenų šaltinių yra surenkami, transformuojami ir perkeliami (*ETL* – angl. *extract*, *transform*,

load) į centralizuotą saugyklą [13]. Šios architektūros privalumas – duomenų gavybos operacijų (informacijos analizė, žinių išgavimas ir kt.) vykdymas nedarant įtakos paskirstytų duomenų šaltiniams. Naudotojai gali siųsti užklausas į didelius duomenų rinkinius. Paprastai tokios sistemos yra labai patikimos, užklausos greitai įvykdomos. Šios architektūros trūkumas – duomenų naujumas, kadangi naujausi duomenys būna paskirstytuose duomenų šaltiniuose ir turi būti sinchronizuojami su duomenų sandėliu.



12 pav. Duomenų sandėlio architektūra



13 pav. Globalios schemos architektūra

Globalios schemos architektūroje, skirtingai nei duomenų sandėlių sistemose, duomenys nėra perkeliami į pagrindinę duomenų saugyklą [13]. Autonominiai duomenų šaltiniai yra integruojami schemos lygmenyje. Tam yra sudaroma globali (13 pav.), integruojamų šaltinių schema (dar vadinama federaline schema arba virtualia duomenų baze). Integracija vykdoma susiejant duomenų šaltinių schemas su globalia schema. Sudarant globalią schemą svarbu suvienodinti šaltinių schemas, susieti schemų elementus ir išspręsti konfliktus tarp jų (pvz., skirtingi schemos elementų pavadinimai ir kt.).

Sistema, kurioje yra įdiegiama globali schema, vadinama globalia ar federaline sistema. Federalinė sistema leidžia siųsti užklausas į visus integruotus duomenų šaltinius. Užklausos į duomenų šaltinius yra siunčiamos per adapterius (angl. *wrapper*), kurie atlieka pateiktų užklausų transformavimo ir rezultatų paketų sudarymo (angl. *encapsulation*) operacijas bei pateikia naudotojams rezultatus. Lyginant duomenų sandėlio ir federalines sistemas paminėtina, kad federalinės sistemos architektūros privalumas – duomenų naujumas. Pagrindinis architektūros trūkumas – ilgesnis užklausų siuntimo laikas.

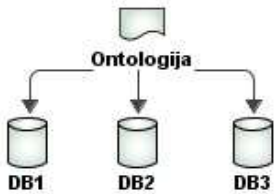
2.6.1.2. Ontologijomis grindžiamas duomenų šaltinių integravimas

Ontologijos teikia galimybę detaliai konceptualizuoti dalykinę sritį, todėl šis semantinių technologijų metodas gali būti taikomas informacijos šaltinių integravimo uždaviniams spręsti. Dalykinės srities konceptualizavimo procese sudaromi ryšiai tarp dalykinės srities konceptų ir nustatomi semantiškai nesuderinami objektai. Paminėtina, kad pats konceptualizavimo procesas padeda semantiškai suderinti skirtingas duomenų šaltinių schemas. Taikant ontologijų metodą galima išspręsti skirtingų schemų terminų daugiaprasmiškumus bei terminų sąryšių konfliktus

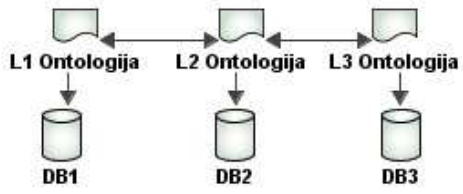
[4], [17], pavyzdžiui, nustatyti ekvivalenčius schemos elementus tarp duomenų bazių ir kt. Tokios problemos apskritai apibūdinamos terminu „semantinis heterogeniškumas“. Ontologijų metodas leidžia naudotojams siųsti užklausas į skirtingus duomenų šaltinius taikant semantinį modelį. Semantinis modelis – globalios schemos atitikmuo (lyginant su tradiciniais metodais). Literatūros šaltiniuose [31], [7], [17] yra pateikiami skirtingi minėtos integracijos įgyvendinimo metodai. Trumpai aptarsime kiekvieną iš jų.

Globalios ontologijos metodas. Taikant šį metodą sudaroma viena ontologija (14 pav.), kurioje aprašomi visi duomenų šaltiniai ir tokiu būdu gaunamas integruotas dalykinės srities vaizdas, kuris dar gali būti apibūdinamas kaip semantinis modelis. Šis metodas leidžia siųsti užklausas per vieną ontologiją. Kai kuriais atvejais gali būti naudojamos kelių specializuotų ontologijų kombinacijos (dar vadinamos modulinėmis arba agreguotomis ontologijomis). Globalios ontologijos metodas gali būti taikomas integruojant konceptualiai panašius informacijos šaltinius [3]. Nors metodo architektūros sudėtis gali atrodyti paprasta, tačiau ontologijai sudaryti reikia dalykinės srities eksperto (dažnai kelių), kuris / kurie žinotų visų duomenų šaltinių semantiką. Pagrindinis metodo trūkumas – semantinio modelio priklausomybė nuo informacijos šaltinių pakeitimų. Pavyzdžiui, vieno informacijos šaltinio schemos pakeitimas gali paveikti visą ontologiją bei kitų informacijos šaltinių ryšius.

Taikant ***sudėtinių ontologijų metodą*** yra sudaroma kiekvieno duomenų šaltinio atskira ontologija (lokali ontologija) bei pavaizduojami ontologijų ryšiai (angl. *mapping*), (15 pav.). Vaizduojant šiuos ryšius svarbu įvertinti integruojamų šaltinių ontologijų terminų semantinį heterogeniškumą. Šioje architektūroje [31], [7] užklausos yra siunčiamos per lokalias ontologijas, o vaizdavimas yra naudojamas lokalių užklausoms siųsti į kitus duomenų šaltinius per jų lokalias ontologijas. Šio metodo privalumas – nepriklausomumas nuo informacijos šaltinių pakeitimų. Pavyzdžiui, pakeitus vieno informacijos šaltinio ontologiją, ji nepaveiktų kitų, pavaizduotų informacijos šaltinių ontologijų. Literatūroje minima [7], kad gana sudėtinga taikyti šį metodą dėl tarpusavio ontologijų semantinio heterogeniškumo aspektų.



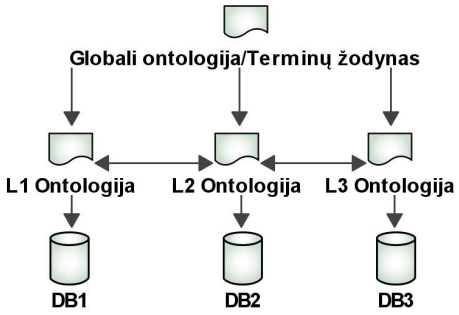
14 pav. Globalios ontologijos architektūra



15 pav. Sudėtinių ontologijų architektūra

Mišrus ontologijų metodas buvo pasiūlytas [7] prieš tai aptartų metodų trūkumams pašalinti. Šio metodo atveju duomenų šaltiniai, kaip ir sudėtinio metodo atveju, yra aprašomi lokalia ontologija (16 pav.), o visų duomenų šaltinių konceptai aprašomi bendra ontologija (dar

vadinam žodynu [7]). Žodynas apima bazinius dalykinės srities terminus. Lokalios ontologijos sudaromos naudojant žodyne aprašytų terminų kompleksinius junginius. Ontologijos konstruojamos vartojant bazinius žodyno terminus, kurie leidžia semantiškai suderinti lokalias ontologijas. Naudotojai užklausas į šaltinius pateikia per bendrą žodyną. Šio metodo privalumai, palyginanti su prieš tai aptartais – paprastesnis papildomų informacijos šaltinių integravimas į esamą struktūrą bei geresnės sąlygos ontologijoms vystyti / plėsti.



16 pav. Mišrių ontologijų architektūra

Apibendrinant aptartus metodus reikėtų pažymėti, kad ontologijos ir duomenų šaltinių schemos yra panašios [9], žiūrint į abu modelius kaip į terminų žodyną, kurie abiejų modelių atveju aprašo dalykinę sritį. Be to, abu modeliai yra suvaržyti dalykinės srities žodyno terminų prasmių / reikšmių.

2.6.2. Semantiniai konfliktai, integravimo proceso problemos ir metodai joms spręsti

Skirtingų duomenų šaltinių integravimo klausimai plačiai analizuojami jau daugiau nei trisdešimt metų. Esami metodai ir algoritmai yra taikomi sistemų diegimo ir palaikymo etapuose. Daugeliu atvejų literatūroje minimų [3], [7], [8], [17], [23] integravimo problemų ir jų sprendimo būdų spręsti duomenų šaltinių heterogeniškumo klausimą yra panašūs. Integravimo procese susiduriama su semantiniais konfliktais, kurie charakterizuojami kaip skirtingos schemų ir duomenų lygmens loginės struktūros. Šie pagal tipą yra skirstomi į: sinonimus, homonimus, duomenų tikslumo bei kt. Minėtiems konfliktams nustatyti galima naudoti jau aptartos *OWL* kalbos savybes, kurios pateiktos 5 lentelėje. Toliau aptarsime pagrindinius konfliktų tipus bei jų nustatymo galimybes naudojant *OWL* savybes [5], [4], [6].

Savybė	Panaudojimo aprašymas
owl:equivalentClass	Nustato ekvivalenčias klases.
owl:euqivalentProperty	Nustato ekvivalenčias ypatybes.
owl:sameIndividualAs	Nustato ekvivalenčius egzempliorius.
owl:disjointWith	Nustato semantiškai ekvivalenčius egzempliorius.
owl:differentFrom	Nustato, kad vienas egzempliorius skiriasi nuo kito.

owl:inverseOf	Pažymi, kad viena ypatybė yra priešinga kitai ypatybei.
owl:unionOf	Pažymi, kad nauja klasė yra gauta iš kitų klasių (sąjungos būdu).
owl:intersectionOf	Pažymi, kad nauja klasė yra gauta iš kitų klasių (sankirtos būdu).
owl:complementOf	Pažymi, kad nauja klasė yra egzistuojančių klasių papildinys.

5 lentelė. OWL / RDF SAVYBĖS KONFLIKTAMS NUSTATYTI

2.6.3. Semantinių konfliktų nustatymas naudojant OWL

Pavadinimų konfliktai. Tokio tipo konfliktai įvyksta, kai du ir daugiau semantiškai ekvivalenčių /neekvivalenčių arba vienodą identifikatorių turinčių konceptų reprezentuoja tą patį objektą. Šie konfliktai įvardijami kaip sinonimai, antonimai ir homonimai:

Sinonimų konfliktai išsprendžiami suvienodinant ekvivalenčių konceptų pavadinimus. Ekvivalentūs konceptai yra nustatomi *OWL* kalbos savybėmis: *owl:equivalentClass, owl:equivalentProperty* ir *owl:sameInvidualAs*. Panagrinėkime pavyzdį (17 pav.), kuriame konceptai *Apskritis* ir *Provincija* yra ekvivalentūs; naudojant *owl:equivalentClass* elementą yra nustatomi tokio tipo konfliktai.

Skirtingų konceptų konfliktai išsprendžiami išskiriant semantiškai neekvivalenčius konceptus. Tokio tipo konfliktai nustatomi *OWL* kalbos elementais *owl:disjointWith* ir *owl:differentFrom*. Pavyzdžiui (18 pav.), konceptai *BendrasĮvertinimas* ir *Pažymys* nėra ekvivalentūs. *OWL* elementu *owl:disjointwiths* yra išskiriami du neekvivalentūs konceptai, ir tokiu būdu išsprendžiamas antonimų konfliktas.

```
<owl:Class rdf:ID="Provincija">
<rdfs:label>Provincija
</rdfs:label>
<owl:equivalentClass
rdf:resource="#Apskritis"/>
</owl:Class>
```

17 pav. Sinonimų konfliktų nustatymo aprašas

```
<owl:Class
rdf:ID="BendrasĮvertinimas">
<rdfs:label>Provincija</rdfs:label>
< owl:disjointwith
rdf:resource="#Pažymys"/>
</owl:Class>
```

18 pav. Skirtingų konceptų konfliktų nustatymo aprašas

Homonimų konfliktai įvyksta, kai konceptai turi skirtingus identifikatorius, tačiau yra žymimi tokiais pačiais ar panašiais pavadinimais. Tokio tipo konfliktai dar yra vadinami identifikavimo konfliktais [5]. Juos reikia spręsti suteikiant atributams skirtingus pavadinimus [5]. Aptarkime pavyzdį (9 pav.): sakykime, turime dvi duomenų šaltinių schemas *A* ir *B*, kurių abiejų atributas yra *tvarkaraščio_Data*, tačiau ontologijose jie yra pavaizduoti skirtingai: *A ontologijoje studijuTvarkaraščioData, o B ontologijoje darboTvarkaraščioData,* tačiau abiejose ontologijose abu atributai žymimi žyme *tvarkaraščioData* (naudojamas elementas *rdfs:label*). Šiame pavyzdyje norint išvengti homonimų konfliktų, *A* ir *B* duomenų šaltinių schemose atributams reikia suteikti skirtingus pavadinimus.

```
<rdf:Property rdf:ID=" studijuTvarkaraščioData ">
<rdfs:label> tvarkaraščio_Data </rdfs:label>
</rdf:Property>
<rdf:Property rdf:ID=" darboTvarkaraščioData ">
<rdfs:label> tvarkaraščio_Data </rdfs:label>
</rdf:Property>
```

19 pav. Homonimų konfliktų nustatymo aprašas

Kompleksinių klasių konfliktai. Tokio tipo konfliktai įvyksta tarp klasių, kurios buvo sudarytos naudojant sankirtos / sąjungos operacijas, kai vieno koncepto elementų priklausomybė nuo kito koncepto elementų yra visiška arba dalinė. Aptarsime du tokių konfliktų atvejus:

Visiškos priklausomybės konfliktų gali įvykti, kai vienas konceptas yra visiškai priklausomas (įeina į kitą konceptą) nuo kito koncepto [15], [5]. Tokio tipo konfliktai yra išsprendžiami klasės atributus išskaidant į dalinius atributus. Šie konfliktai gali būti nustatomi konstrukciniu elementu *owl:unionOf*. Panagrinėkime pavyzdį (20 pav.), kuriame turime du konceptus *Vardas* ir *Pavardė*, ir *owl:unionof* elementas pažymi, kad minėti konceptai yra gauti sąjungos būdu. Kitaip sakant, jei klasė *A* gauta sąjungos būdu iš klasių *B* ir *C*, o *A* klasė nėra ekvivalenti *B* ir *C* klasėms, tada sujungiant *A* ir *B* arba *A* ir *C* klases gali įvykti visiškos priklausomybės konfliktas.

```
<rdf:Property rdf:ID="Pavardė">
< owl:unionOf
rdf:resource="#Vardas">
</rdf:Property>
```

20 pav. Visiškos priklausomybės konfliktų nustatymo aprašas

```
<owl:Class rdf:ID="Asistentas">
<owl: intersectioOf
rdf:resource="#Absolventas"/>
</owl:Class>
```

21 pav. Dalinės priklausomybės konfliktų nustatymo aprašas

Dalinės priklausomybės konfliktų įvyksta tarp dviejų iš dalies sutampančių konceptų. Tokio tipo konfliktai nustatomi per *OWL* ypatybę *owl:intersectioOf*. Šie konfliktai gali būti išsprendžiami išskiriant iš dalies sutampančias konceptų dalis [5], [4]. Pavyzdyje, 21 pav., matyti dviejų schemų integracija, kur vienoje yra *Asistentas*, o kitoje – *Absolventas*. Čia abu konceptai (asistentas ir absolventas) iš dalies sutampa, nes kai kurie asistentai gali būti ir absolventai, bet ne visi asistentai yra absolventai ir ne visi absolventai yra asistentai. Šiuo atveju norint išvengti konfliktų, konceptai turi būti išskaidomi, sudarant atskiras klases: studijų nebaigusius asistentus, studijas baigusius asistentus ir studijuojančius asistentus.

2.6.4. Ontologijų konstravimo metodologija

Konceptualizuojant dalykinę sritį, yra svarbu, kad sudarytą domenų modelį, prireikus galėtumėm vystyti, pakartotinai naudoti ir būtų užtikrintas modelio teisingumas. Dėl minėtų priežasčių, svarbu pasirinkti tinkamą metodologiją, kuri užtikrinti nuoseklų ontologijos kūrimo procesą. Ontologijos kuriamos skirtingoms dalykinėms sritims ir tikslams, todėl konkrečiam uždaviniui turi būti pasirinkta ar sudaryta tam tinkama metodika. Pagal sprendžiamą uždavinį metodikas galima išskirti į dvi grupes: *automatizuotam* modelio sudarymui (generavimui) ir *neautomatizuotam*. *Automatizuotas* ontologijos kūrimo procesas yra kai ontologija yra sugeneruojama iš konceptualizuojamo duomenų resurso, pvz., duomenų bazės, vykdant transformacijas tarp kalbų. *Neautomatizuotas*, kai ontologiją kuriama rankiniu būdu. Iki šiol yra pasiūlyta nemažai metodikų, kurias aptarsime.

TOVE (angl. *Toronto virtual enterprise*), ontologijų inžinerijos metodologija [24] skirta aprašyti sprendžiamą uždavinį scenarijumi, kad būtų galima išgauti kuo išsamesnę informaciją, tolimesniam ontologijos plėtojimui. Išskiriami šie ontologijos sudarymo etapai:

Motyvacijos scenarijus. Šios etapo metu nustatomos problemos, su kuriomis su kuriomis susiduria įmonėje, tam tikroje dalykinėje srityje ir sudaromi reikalavimai jų sprendimui.

Neformalūs kompetencijos klausimai. Remiantis motyvacijos scenarijumi, sudaromi neformalūs kompetencijos klausimai į kuriuos ontologija turi atsakyti. Klausimai turi būti sudaromi taip, kad būtų galima gauti atsakymą iš sudėtinių klausimų. Šiame etape sudaryti klausimai ir jų junginiai turi atsakyti į visas ankstesniame etape nustatytas problemas.

Terminų specifikavimo etape aprašomo objektai, atributai ir ryšiai tarp jų.

Formalūs kompetencijos klausimai. Šiame etape ontologijos reikalavimai yra formalizuojami, tam naudojant pačių pasirinktą terminologiją.

Aksiomų specifikavimas. Aksiomos nurodo sąvokų apibrėžimus ir jų apribojimus, užrašomos naudojant pirmos eilės logiką. Aksiomos turi išreikšti kompetencijos klausimus.

Pilnumo įvertinimo stadijoje yra įvertinimą kaip ontologija atsako į kompetencijos klausimus. Tam yra sudaromi vertinimo kriterijai, pagal kuriuos nustatomas atsakymo tikslumas.

2.6.5. Ontologijų sudarymo įrankių analizė

Ontologijų sudarymui, jų apjungimui bei vystymui svarbu pasirinkti tam tinkamą įrankį. Modelių konstravimo įrankių įvairovė gausi, tačiau svarbu įvertinti konkrečiam uždaviniui tinkamiausią įrankį, tinkamas pasirinkimas sumažina daug rankinio darbo bei klaidų tikimybę. Buvo analizuotas *Protégé, NeOnToolKit ir Altova SemanticWorks* įrankių funkcionalumas ir panaudojimo galimybės integravimo uždaviniams spręsti.

Protege yra atviro kodo principais platinamas, sukurtas Stanfordo universitete, vienas iš plačiausiai naudojamų ontologijų sudarymo įrankis. Įrankis turi grafinę sąsają, leidžia sukurti ontologiją nuo pat hierarchija sudarymo etapo. Ši programinė įranga yra plečiamos, modulinės architektūros, teikia daug papildomų komponentų, kurie išplečia įrankio funkcionalumą, tokių kaip *OWLViz, OntoGraf, Promt Suite* ir kt. Įrankis yra tinkamas ontologijų pakartotinei inžinerijai vykdyti, juo galima vystyti sukurtas ontologijas, tam įvedant papildomas klasės, egzempliorius, sudaryti ryšius ir apribojimus. Taip pat, turi komponentą *DataMaste*r, kuris leidžia automatizuoti duomenų šaltinių integravimo procesą, importuojant reliacinių duomenų bazių schemos struktūra į *Protege*. Turi ontologijų eksportavimo į XML / OWL, XML ir kt., funkcijas. Įrankiu galima vizualizuoti sudarytą ontologiją, pavaizduojant ją grafu. Platforma pritaikyta semantinio tinklo paslaugoms, turi integruotą *Jena* karkasą, kuris leidžia susieti ontologiją su saitynu, tam naudojan *API* (angl. *aplication programing interfase*) metodus. Turi integruotą *SPARQL* procesorių ir *Palet, FaCT++* ir *Hermit* klasifikatorius. *Protege* yra įgyvendintas *Java* programavimo kalba.

NeOnToolKit tai *Eclipse* pagrindu įgyvendintas ontologijų inžinerijai skirtas įrankis, leidžiantis kurti, apjungti ir vystyti ontologijas. Teikia priemones ontologijoms vizualizuoti. Įrankis kaip ir *Protege* yra modulinės, plečiamos architektūros, turi gausų plėtinių pasirinkimą, tokių kaip *KC-Viz*, kuris leidžia vizualizuoti ontologiją, *ODEMapster* procesorių, kuriuo galima vykdyti reliacinių duomenų bazių vaizdavimą su ontologiją ir kt. Turi patogią grafinę sąsają. Platformos funkcionalumas išplečiamas naudojant skirtingas perspektyvas, tokias kaip *R2O,* kuri skirta duomenų šaltinių integravimui, *OWL* ontologijų sudarymui ir kt. Palaiko XML / OWL, XML, RDF ir kt., turi ontologijų eksportavimo / importavimo funkcijas. Sistema turi įgyvendintą *SPARQL* procesorių, teikia *API* (angl. *aplication programing interfase*) metodus programavimui.

Altova Semantic Works – komercinis *RDF / OWL* redaktorius, turintis patogią grafinę sąsają, leidžiančią vizualiai modeliuoti ontologijas, sudaryti *RDFS* žodynus, ontologijos ir juos eksportuoti *RDF, XML, OWL Lite / DL / Full, N-triple* formatus. Įrankiu galima konvertuoti dokumentus tarp RDF / XML ir N – Triples standartų ir generuotų jų kodą. Redaktorius turi automatizuotą sintaksės, formato ir semantikos tikrinimo mechanizmą, kuris užtikriną efektyvų darbą ir dokumento teisingumą Taip pat teikia tekstinę ontologijų aprašymo formą. Įrankis leidžia sudarytų ontologijų grafinę reprezentaciją išsaugoti paveikslėlių formatais. Pagrindinis įrankio trūkumas – komercinis produktas, neturi plėtimo galimybių. Gamintojas teikia pilno funkcionalumo 30 dienų bandomąją versiją.

Apibendrinant aptartus įrankius, paminėtina, kad visi įrankiai suderinti su šiuo metu pagrindinėmis ontologijų kalbomis / standartais. Kiekvienas įrankis turi privalumų ir trūkumų: *Altova Semantic Works* redaktoriaus teikiama grafinė sąsaja leidžia efektyviai kurti ontologijas; *Protégé* teikia gausų plėtinių pasirinkimą, tačiau turi skurdesnę grafinę sąsają; *NeOnToolKit* turi plėtinių ontologijos

vizualizavimui bei duomenų šaltinių ir ontologijos vaizdavimui. Visų paminėtų įrankių gamintojai teikia išsamią naudotojo dokumentaciją.

Šiame darbe nuspręsta naudotis visais aptartais įrankiais, kurie bus naudojami skirtinguose darbo vystymo etapuose, kaip labiausiai tinkantys sprendžiamam uždaviniui įgyvendinti. Pagrindiniai įrankiai pasirinkti *Protégé,* kuris bus naudojamas pradiniame ontologijos sudarymo etape, sudarant klases, egzempliorius ir savybes bei automatizuotam klasių ir egzempliorių klasifikavimui ir *NeOnToolKit,* kuris bus naudojamas metodikos sudarymo etape, semantinių konfliktų tarp duomenų šaltinių schemų sprendimui, o *Altova Semantic Works* kaip pagalbini, kuris bus naudojamas ontologijų diagramų vaizdavimui ir spausdinimui. Įrankių lyginamoji analizė pateikta 6 lentelėje.

Savybės	Įrankio pavadinimas		
	Protégé	*NeOnToolKit*	*Altova Semantic Works*
Ontologijų sudarymo kalbos	RDF, RDFS, OWL, Turtle	RDF, RDFS, OWL, Turtle	RDF, RDFS, OWL, N-Triple
SPARQL užklausų variklis	Taip	Taip	Ne
API programavimo sąsaja	Taip	Taip	Ne
Ontologijos vaizdavimas grafiniu pavidalu	Grafas	Grafas	Schema
Grafinė sąsaja	Skurdi	Patenkinama	Labai gera
Klasifikatoriai	Palet, FaCT++ Hermit	Palet, Hermit	Ne
Plečiamumas	Taip	Taip	Ne
Licencijavimas	GNU	GNU	Komercinė
Dokumentacija	Išsami	Išsami	Išsami
Prieiga prie duomenų bazių	Taip	Taip	Ne
Operacinės sistemos	Windows, Linux	Windows, Linux	Windows

6 lentelė. Ontologijų sudarymo įrankių analizė

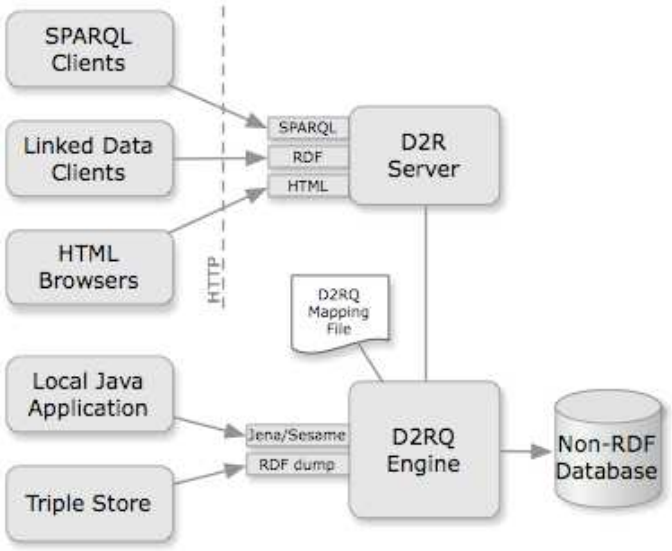
2.6.6. Semantinių paslaugų publikavimo įrankių analizė

Šiame poskyryje aptarsime šiuo metu naudojamus metodus ir priemones skirtus duomenų modelių transformavimui, duomenų susiejimui ir jų publikavimui. Pirmiausia aptarsime iš dalies panašių, *W3C* konsorciumo rekomendacijas ir standartus atitinkančius, ir realizuotus sprendimus, tai *D2R* ir *Openlink Virtuoso Server*. Kiti du sprendimai, tai *Microsoft* sprendimams skirta sistema *Semantics.Framework* ir atviro kodo semantinio suderinimo karkasas *S-Mach.*

DR2 Server yra sėkmingas projektas [18], kuris skirtas reliacinių duomenų bazių turinio publikavimui *RDF* formatu, semantiniame žiniatinklyje. Bazinis *D2R Server* funkcionavimas bazė remiasi pavaizdavimo metodu (angl. *mapping*) reliacinių duomenų bazių SQL užklausų transformavimui į išteklius ir jų savybes. Be to, D2R serveris teikia funkcijas žiniatinklio agentams, siųsti užklausas į *RDF* duomenis, naudojant *SPARQL*, o duomenis pateikia *HTML* ir *RDF* formatais.

D2R sudaro dvi pagrindinės dalys: pirma, *D2RQ* variklio, kuris yra branduolys leidžiantis valdyti duomenų bazes (*RDF* ir *RDB*) bei taikyti pavaizdavimo modelius sąryšių sudarymui su išoriniais (internetas) bei vidiniais duomenų šaltiniai; antroji dalis yra *D2RQ* Variklis. Jis valdo transformavimo iš reliacinių duomenų į *RDF* procesą. Tam naudojama *D2RQ* pavaizdavimo kalba. Transformavimo rezultatas – sugeneruotas pavaizdavimo dokumentas. Antroji – *D2RQ* variklis. Jis valdo transliavimo procesą tarp reliacinių duomenų bazių ir *RDF*. Tam naudojamas D2RQ vaizdavimo dokumentas (modelis), kuriame yra aprašyti duomenų šaltinių ir ontologijos ryšiai.

Platforma teikia tris skirtingas sąsajos duomenų mainams, naudojant *HTTP* protokolą: *SPARQL* semantinėms užklausoms, *RDF* susietiems duomenų šaltiniams ir *HTML*.



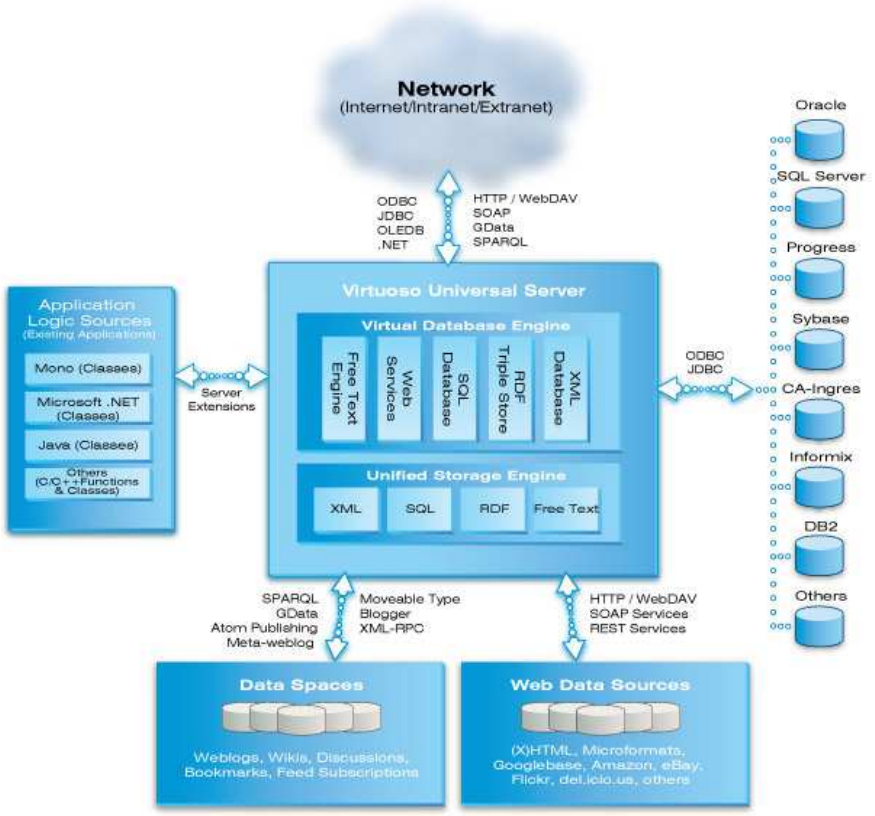
22 pav. D2R Server architektūra

Pav. šaltinis: http://d2rq.org/d2r-server

Openlink Virtuoso yra universali platforma [25] teikianti semantinių technologijų paslaugas. Jis veikia kaip virtuali duomenų bazės. Platforma suderinta su daugeliu duomenų bazių sistemų, tokių kaip *MS SQL, Oracle, MySQL* ir kt. Ši sistema taip pat teikia *XML, WSDL, SOAP* paslaugas, duomenys pasiekiami per *ODBC* tvarkykles. Sistemos architektūra pateikta? pav.

Sistema teikia susietų su reliacinėmis duomenų bazėmis vaizdus. Duomenis galima gauti SPARQL užklausomis. Sistemoje naudojama deklaratyvi meta schemos kalba SQL duomenų / schemai susieti su ontologija.

Virtoso platforma yra modulinio tipo, gamintojas tekia plėtinius įvairiems uždaviniams spręsti, tokiems kaip XML duomenims valdyti, dokumentų *Web* paslaugoms įgyvendinti, *Web* taikomųjų programų paslaugoms teikti ir kt. Turi tvarkykles ir bibliotekas, darbui Microsoft .Net sprendimais.



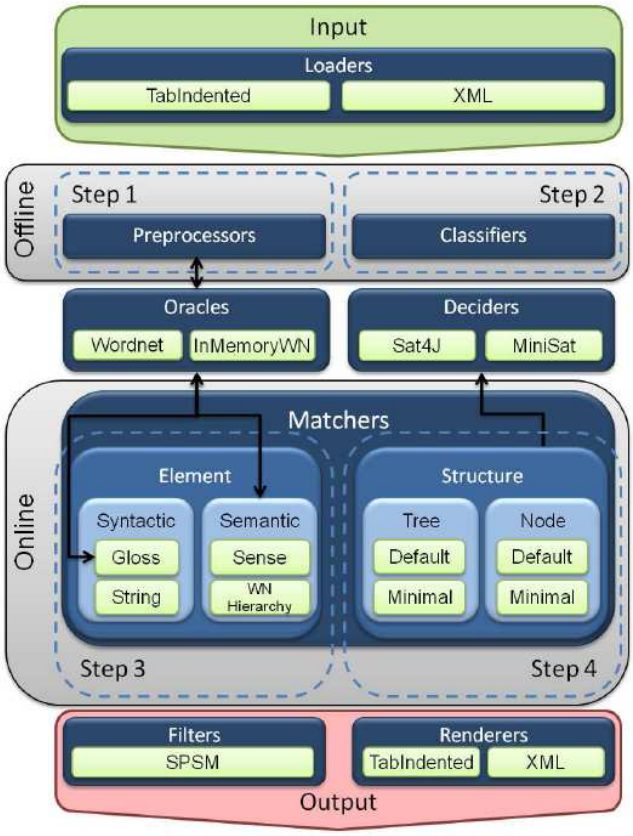
23 pav. Virtuoso Server architektūra

Pav. šaltinis: http://virtuoso.openlinksw.com/

S-Match – semantines paslaugas teikianti sistema [26], kuri teikia semantines suderinimo paslaugas ir algoritmus joms įgyvendinti. Sistema apima metodus ir priemones resursams apjungti (ontologijas, duomenų šaltinių schemas), automatizuotai atvaizduoti duomenų šaltinius ir failų sistemų direktorijas, sudaryti ontologijas ir kt. Be to, atvaizduojamas grafas įvertinamas, taikant panašumų įvertinimą koeficientus.

Sistema architektūra yra komponentų tipo, ją sudaro: *įkėlimo komponentai (angl. loaders),* kurie skirti rinkmenoms, turinčius medžio struktūrą ir blokinį suskirstymą įkelti į sistemą. Sistema suderinama su *XML, RDF* ir *OWL* standartais; P*irminio apdorojimo procesorius* vykdo metaduomenų transliavimą (žymių, ontologijų klasių pavadinimų ir kt.); *Klasifikatoriai* skirti

mazgų konceptams sudaryti; *prognozavimo komponentai (angl. oracles)* vykdo semantinio suderinamumo funkcijas; *Sprendimo komponentas (angl. deciders)* vykdo semantinių santykių skaičiavimą tarp koncepto mazgų; *Derinimo komponentas (angl. matchers)* susideda iš dviejų paketų, pirmas – elementų lygmens derinimo komponento ir struktūrinio derinimo komponentų. Elementų lygio derinimo komponentas skaičiuoja santykius tarp mazgo žymių, o struktūrinio derinimo komponentas vykdo semantinių santykių skaičiavimą tarp mazgų konceptų. *Filtravimo komponentas (angl. filters)* teikia atvaizdavimo detalumą / aktualumą. *Atvaizdavimo komponentas (angl. renderers)* formuoja pavaizduotų elementų išvedimą.



24 pav. S-Match sistemos architektūra

Pav. šaltinis: http://semanticmatching.org/s-match.html/

Karkasas yra pristatomas kaip semantinio suderinamumo projektas, teikią sąsają plėtimui, Pakete yra realizuoti keli naudotojų sąsajos tipai: grafinė, komandinė aplinka bei programos plėtimo. Turi Java biblioteką, kuri leidžia plėsti sistemos galimybes. Platinama *GNU* (angl. *general public license*) licencijos sąlygomis. Sistemų lyginamoji analizė pateikta 7 lentelėje.

Savybės	Įrankio pavadinimas		
	D2RQ	*Virtuoso*	*S-Match*
Ontologijų saugykla	RDF, RDFS, OWL, Turtle	RDF, RDFS, OWL, Turtle	RDF, RDFS, OWL Lite
SPARQL užklausų variklis	Taip	Taip	Taip
API programavimo sąsaja	Taip	Taip	Taip
Ontologijos vaizdavimas grafiniu pavidalu	Grafas	Grafas	Schema
Grafinė sąsaja	Skurdi	Gera	Patenkinama
Reliacinių duomenų bazių integravimo galimybė	Taip	Taip	Taip
Vaizdavimo kalba	D2RQ	RDB2RDF (angl. Relational Databases to RDF)	Tiesioginis
Plečiamumas	Taip	Taip	Taip
Licencijavimas	GNU	GNU ir komercinė	Komercinė
Dokumentacija	Išsami	Išsami	Išsami
Prieiga prie duomenų bazių	Taip	Taip	Ne
Operacinės sistemos	Windows, Linux	Windows, Linux	Windows, Linux

7 lentelė. Karkasų / sistemų palyginimo lentelė

2.6.7. Vaizdavimo kalbų analizė

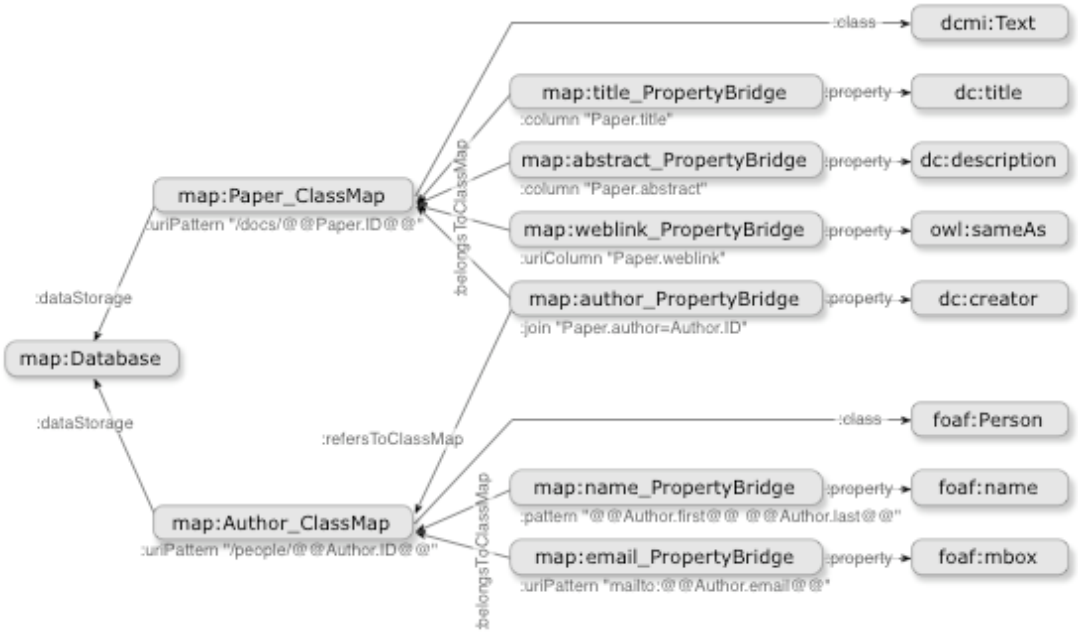
Duomenų šaltinių vaizdavimas, tai duomenų šaltinių ir ontologijos susiejimas, kuris gali būti aprašomas viena iš standartizuotų kalbų, tokių kaip *D2RQ*, *Tiesioginio RDB* į *RDF vaizdavimo* kalbos. Šis procesas pradedamas nuo duomenų bazės schema lentelių ir sąryšių tarp jų nustatymo. Remiantis nustatytais atvejais kiekvienas duomenų bazės komponentas (lentelės, stulpelis, apribojimas) yra konvertuota į atitinkamą ontologijos komponentą (klasė, savybės, ryšys). Pats susiejimas gali būti atliekamas dalykinės srities eksperto (rankiniu būdu) arba generuojamas naudojant transformavimo metodus. Transformuotas ar sudarytas pavaizdavimo rezultatas išsaugomas, kaip pavaizdavimo dokumentas, kuris vėliau gali būti papildomas / koreguojamas.

D2RQ yra deklaratyvi vaizdavimo kalba, skirta aprašyti santykiams tarp reliacinės duomenų bazės schemos ir RDFS žodyno arba OWL ontologijos. Vykdant transformacijų operacijas, ji apibrėžti kaip RDF trejetas (*subjektas, savybė, reikšmė*) turi būti sukurtas iš reliacinių duomenų šaltinių. Šią kalbą yra naudojama *D2RQ* ir *NeonToolKit* ir kt. platformų

aprašant integruojamus duomenų šaltinius. Kalba yra daugiau skirta aprašyti duomenų šaltinių išdėstymą, o ne visas galimas klasių ypatybes.

Pagrindinis kalbos elementas, sudarantis D2RQ vaizdavimą yra *d2rq:ClassMap*. Juo yra nustatomi ryšiai tarp elementų, kurie yra išvedami iš tam tikrų išteklių. Šiuo elementu kiek yra nustatyta atskirų klasės atvejų. Elementas *d2rq:ClassMap* yra susiejamas su *d2rq:Database* elementu.

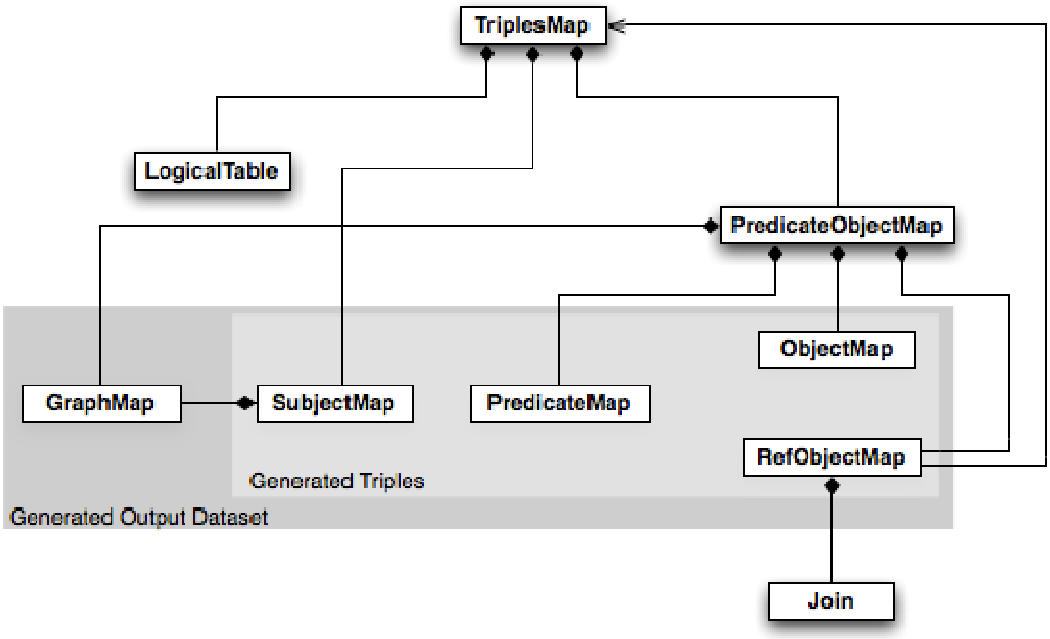
D2R vaizdavimo kalba turi mechanizmus, leidžiančius nustatyti atskirus klasės atvejus, kalboje kiekvienas *classMap* elementas yra *d2rq:propertyBridges* sudėtyje. Šie elementai suteikia susiejimo savybes *classMaps* elemento sugeneruotiems resursams. Tai reiškia, kad *d2rq:propertyBrigde* suteikia predikatą ir objekto dalis generuojamam *RDF* trejetui. Elementas *d2rq:propertyBridges* teikia tokį patį, kaip ir atvaizduojant klases *URI* generavimo mechanizmą. Kalbos pavyzdys pateiktas



25 pav. D2RQ vaizdavimo struktūra

Pav. šaltinis: http://d2rq.org/d2rq-language

R2RML tai reliacinių aprašų ir RDF vaizdavimo kalba (angl. *Relational to RDF mapping*). Kalba patvirtinta *W3* konsorciumo. Kalba leidžia reliacines duomenų bazes atvaizduoti RDF duomenų modeliu. Kalba turi vaizdavimo taisykles, kurios aprašo kaip turi būti transliuojami reliacinių duomenų bazių duomenys. Tokios taisyklės vadinamos *RDF* trejetų vaizdavimais. Kalbos vaizdavimo schema pateikta 26 pav.



26 pav. R2RML vaizdavimo kalbos schema

Pav. šaltinis: http://www.w3.org/TR/r2rml/

2.7. Siekiamo uždavinio sprendimo metodikos apibrėžimas

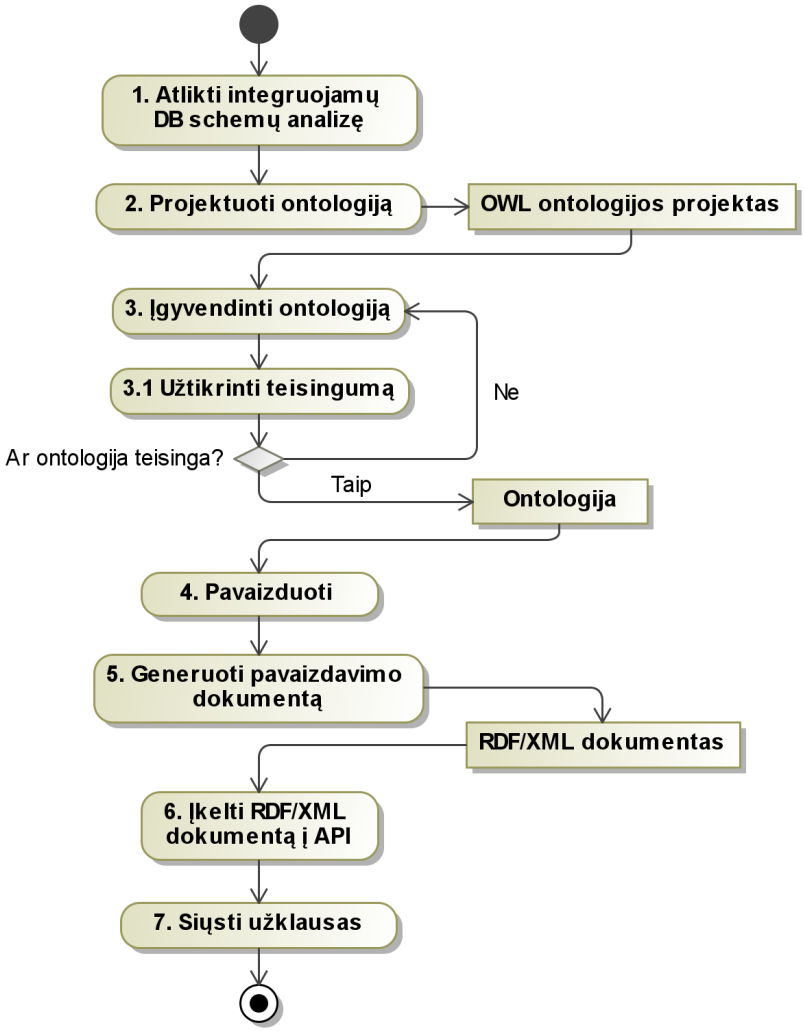
Siekiamas sistemos sprendimas - duomenų surinktų iš heterogeninių duomenų šaltinių konfliktų, kylančių duomenų integravimo metu sprendimas. Duomenų integravimui naudojant globalią ontologiją, o konfliktų sprendimui konfliktų sprendimų ontologiją. Veiklos kontekstas aprašytas kontekstine diagrama (20 pav.).

2.7.1. Uždavinio sprendimo struktūra

Šioje dalyje aptarsime siūlomą schematinio integravimo procesą, taikant globalios ontologijos metodą. Heterogeninių duomenų šaltinių integravimo uždavinių procesas pavaizduotas 24 pav. Proceso diagramoje kiekvienas uždavinys pažymėtas skaitmenimis, kuriuos detaliai aptarsime.

2.7.1.1. Uždavinio sprendimo metodika

Integravimo procesas (27 pav.) pradedamas nuo integruojamų duomenų šaltinių schemų analizės, kurios metu nagrinėjami schemų konceptai, jų egzemplioriai ir ryšiai tarp jų.



27 pav. Duomenų šaltinių integravimo proceso diagrama

Antrame etape, remiantis integruojamų duomenų šaltinių analizės duomenimis, sudaromas ontologijos projektas. Tam naudojami programiniai įrankiai, tokie kaip *MagicDraw*, *Altova Semantic Works, NeOnToolKit* ir kiti bei ontologijų sudarymo metodologijos, pvz., *TOVE, UPON* ir kt. Metodologijos aprašo, kokio proceso reikėtų laikytis sudarant, jungiant ar pakartotinai naudojant ontologijas, taip pat aptariami ontologijų evoliucionavimo aspektai ir jų pritaikymas taikomajai programinei įrangai.

Trečiame etape įgyvendinama ontologija. Tam taip pat rekomenduojama naudotis ontologijų sudarymo metodologijomis bei programiniais įrankiais, pvz.: *Protege, NeOnToolKit, Altova Semantic Works* ir kt. Ontologijų konstravimas yra iteracinis procesas, kuris pradedamas nuo dalykinės srities nustatymo, po to išvardijami terminai, sukuriamos klasės ir jų egzemplioriai, sudaromos ypatybės (ryšiai) tarp klasių ir jų egzempliorių, klasių egzemplioriams priskiriamos reikšmės ir galiausiai sudaromos aksiomos. Ontologijos teisingumui nustatyti, yra naudojami loginių išvadų varikliai, tokie kaip *Palet, F++, HermiT* ir kt., kurie iš sudarytų faktų leidžia atlikti logines išvadas ir automatizuotai sudaro klasių hierarchiją. Kai kurie įrankiai (*Protege, NeOnToolKit*) turi integruotus išvadų variklius.

Ketvirtame etape pavaizduojamos ontologijos ir duomenų šaltinių schemos (angl. *mapping*). Tai daroma šiais būdais: pirmas – naudojant įrankį *NeONToolKit su ODE Mapster* plėtiniu. Įrankis leidžia grafiniu režimu sukurti duomenų šaltinių ir ontologijos vaizdavimo dokumentą; antras – naudojant *2DR* programinį paketą [18], kuriuo galima sugeneruoti vaizdavimo dokumentą. *2DR* programinis paketas naudoja *D2RQ* vaizdavimo kalbą, kurios aprašas pateiktas [19] šaltinyje; trečias – naudojant *W3C* [20], [21] tiesioginio vaizdavimo standartą, kuris leidžia vaizduoti duomenų šaltinius ir globalią ontologiją. Sudarant atvaizdą, gali būti naudojamas vienas iš antrame etape paminėtų ontologijų konstravimo įrankių.

Penktame etape iš sudaryto vaizdavimo modelio sugeneruojamas *RDF*, XML ar kito specifinio formato dokumentas. Paminėtina, kad sugeneruotas vaizdavimo dokumentas (jo standartas) turi būti suderintas su integravimo platforma, kurioje jis bus naudojamas.

Šeštame etape sugeneruotas modelis įdiegiamas taikomajai programinei įrangai, turinčiai *OWL / RDF* bibliotekas, arba gali būti naudojamas integravimui skirtuose technologiniuose sprendimuose, pvz., *2DR*, *Virtuoso* ir kt.

Septintame etape suderinama sistema ir siunčiamos semantinės užklausos į ontologijoje aprašytus duomenų šaltinius.

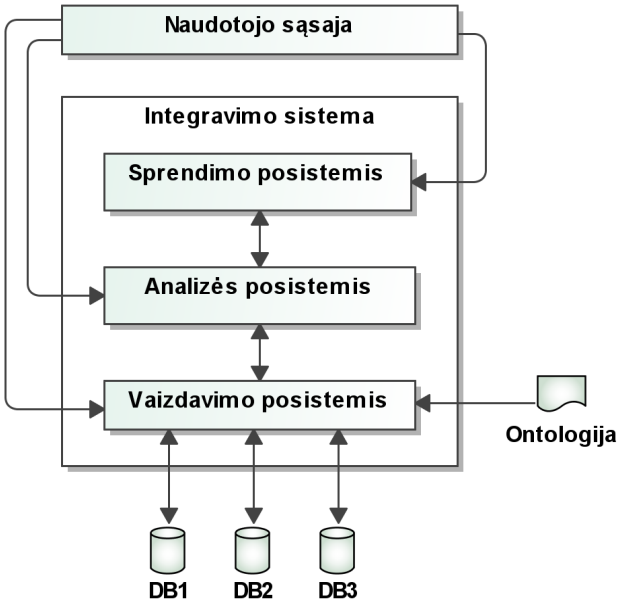
2.7.1.2. Integravimo sistemos prototipo komponentai uždaviniui spręsti

Aprašytų ontologijomis grindžiamų duomenų šaltinių integravimo metodams bei jų procesui įgyvendinti literatūros šaltiniuose [2], [16], [18], [19] pateikiamos įvairios galimos integravimo sistemų komponentų kompozicijos bei galimi, semantinio duomenų šaltinių integravimo sprendimai. Atsižvelgiant į technologinius pasiekimus ir prieinamas eksperimento priemones, šiame darbe aprašoma architektūra, kurią galima apibūdinti kaip susietų komponentų sistemą, kurios pagrindinės dalys nurodytos 28 pav. Toliau detaliai aptarsime kiekvieną iš jų.

Vaizdavimo posistemis vykdo atvaizdavimą tarp duomenų šaltinių ir globalios ontologijos bei leidžia vykdyti semantines užklausas. Vaizdavimui gali būti naudojami tokie įrankiai: *NeonToolKit, 2RD, Protege* (naudojant *DataMaster* išplėtimą), *2DRQ* ir *W3C* kalbos [20] ir *Jena* ar *Sesam* adapteriai.

Analizės posistemis vykdo semantinių konfliktų nustatymo operacijas, o aptiktus konfliktus perduoda *sprendimo posistemiui*, kuris atlieka objektų transformavimo operacijas. Literatūroje [11] aptariami įvairūs galimi šio komponento įgyvendinimo ir diegimo atvejai. Tai gali būti *SOA* (angl. *service oriented architecture*). Vienas iš *SOA* atvejų *XML* technologijos pagrindu veikiantis servisas, kuriuo yra vykdomos objektų transformavimo operacijos. Kitas *RDF / OWL* pagrindu veikiantis servisas *SWS* (angl. *semantic web service*) [11]. Konfliktai paprastai nėra sprendžiami visiškai automatizuotai, dažniausiai – pusiau automatizuotai, leidžiant naudotojui priimti konkrečiam atvejui tinkamą sprendimą.

Naudotojo sąsaja skirta sistemai konfigūruoti, ontologijai įkelti, konfliktams valdyti ir užklausoms į duomenų šaltinius siųsti.

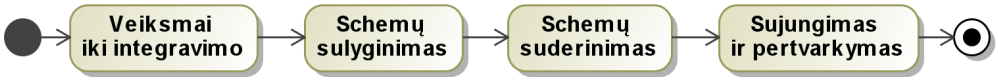


28 pav. Integravimo sistemos architektūra

Komponentai gali būti sujungiami naudojant programavimo technologijas (pvz., *JAVA, .Net ir kt.*), kurios turi bibliotekas dirbti su *RDF / OWL*, pvz., *Java.API, DotNet.RDF* ir kt.

2.7.2.Principinis integravimo algoritmas

Heterogeninių duomenų šaltinių integravimo ontologijų pagrindu metodai yra įgyvendinami pagal procesą, pateiktą 29 pav. Išskiriami šie proceso etapai: pasiruošimas arba veiksmai iki integravimo. Šiame etape yra atliekama integruojamų schemų analizė, sudaromas integravimo algoritmas ir nustatomas integravimo veiksmų eiliškumas; schemų sulyginimo etape suderinami schemų konceptai ir nustatomi jų konfliktai; schemų suderinimo etape sprendžiami schemų konfliktai; jungimo ir pertvarkymo etape schemos sujungiamos ir pertvarkomos pagal iš anksto apibrėžtus kriterijus.



29 pav. Principinis integravimo algoritmas

2.8. Darbo tikslas ir siekiami privalumai

Darbe siekiama ištirti egzistuojančias duomenų integravimo sistemas, skirtas heterogeninių duomenų šaltinių integravimui, ištirti jų veikimo principus, nustatyti ontologijų pritaikymo galimybes duomenims integruoti. Atskleisti galimybe pritaikyti ontologijų metodo adaptavimą vienai iš egzistuojančių sistemų. Ištirti algoritmus semantinių konfliktų sprendimui, nustatyti jų trūkumus bei patobulinti, atlikti jų efektyvumo testavimą.

2.9. Nefunkciniai reikalavimai ir apribojimai

Duomenų bazių integravimo sistemai keliami nefunkciniai reikalavimai:

1. Reikalavimai suderinamumui

Sistema turi būti suderinama duomenų integravimui iš reliacinių duomenų bazių.

2. Reikalavimai sistemos saugumui

Sistema privalo užtikrinti sistemos naudotojų informacijos slaptumą. Slaptažodžiai turi būti užšifruojami ir saugomi duomenų bazėje.

3. Reikalavimai programinei įrangai

Sistemos turi būti realizuota naudojant JAVA arba Microsoft.NET programavimo kalbas.

2.10. Rizikos faktorių analizė

Rizikos faktorius	**Tikimybinis įvertinimas**
Reikalavimų specifikacijos pasikeitimai realizavimo fazėje	5/10
Realizavimo fazėje žymūs sistemos pasikeitimas, dėl kurių sukurta dalis netiks	4/10
Nesklandumai semantinių konfliktų sprendimų metodo pritaikyme	6/10

8 lentelė. Rizikos faktoriai

Sprendimo būdas

Rizikos faktoriai ir numatomas planas problemoms spręsti

Rizikos faktorius	**Problemos sprendimas**
Reikalavimų specifikacijos pasikeitimai realizavimo fazėje	Užtikrinti sistemos reikalavimų stabilumą, numatyti papildomo laiko pakeitimų vykdymui
Realizavimo fazėje žymūs sistemos pasikeitimai, dėl kurių sukurta dalis netiks	Atsisakyti kai kurių savybių, kurios galėtų sukelti neatitikimą ir likus laikui jas perkurti naujai pritaikant sistemos pokyčiams

9 lentelė. Rizikos faktorių sprendimo būdai

2.11. Analizės išvados

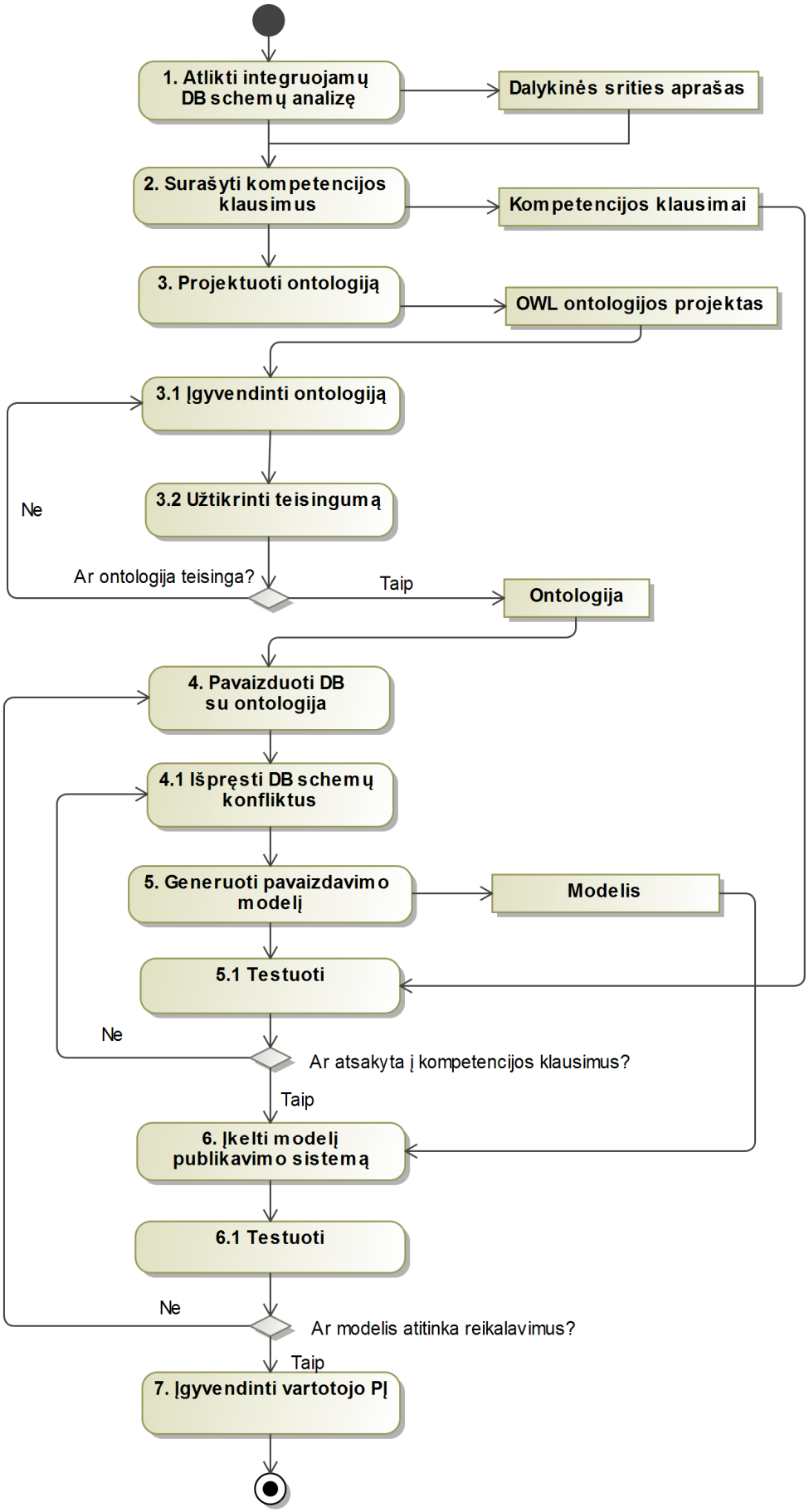
Sistemos analizės metu buvo apibrėžtos sistemos ribos, nustatyti kompiuterizuojami panaudojimo atvejai, ištirta sistemos elgsena bei nustatyti sistemos naudotojai.

1. Ontologijų taikymas duomenų šaltinių integravimui reikalauja analitinio požiūrio, reikia atlikti dalykinės srities analizę, apibrėžti sistemos ribas, sudaryti procesą teisingam ontologijų įgyvendinimui ir vystymui.
2. Atlikus technologinių sprendimų analizę, paaiškėjo, kad egzistuoja technologiniai sprendimai leidžiantys automatizuotai atlikti integruojamų duomenų šaltinių transformacijas – tai pagreitina integravimo procesą. Pastebėta, kad transformuoti dariniai iš karto po transformacijos neatitinka poreikių ir turi būti koreguojami rankiniu būdu.

3. Ontologijomis grindžiama reliacinių duomenų bazių integravimo metodika

3.1. Metodikos procesas

Remiantis atliktos analizės duomenimis – sudaryta *RDB* integravimo metodika, tam parinkti įrankiai, maksimaliai leidžia automatizuoti kūrimo procese vykdomas operacijas. Procesas apima ontologijos generavimo, vystymo, pritaikymo konkrečiai dalykinei sričiai bei jos teisingumo užtikrinimą. Kitas svarbus proceso operacija – duomenų šaltinių vaizdavimas su dalykinės srities ontologija, vaizdavimo teisingumo testavimas bei modelio publikavimas. Kai paskelbiamas modelis, galima siųsti semantines užklausas į duomenų šaltinius. Duomenų šaltinių integravimo proceso diagrama pateikta 30 pav.



30 pav. Metodikos proceso diagrama

3.1.1. Dalykinės srities analizė

Prieš atliekant duomenų šaltinių integravimą atliekama dalykinės srities analizė, kurios metu iš nustatoma srities ribos, jos aktoriai ir konceptai, kurie atstovauja dalykinę sritį. Sudaromas dalykinės srities aprašas, kontekstinė diagrama. Dalykinės srities analizė pateikta darbo 4.2 punkte.

3.1.2. Reikalavimų sudarymas

Vienas iš būdų nustatyti, ontologijai reikalavimus – sudaryti kompetencijos klausimus. Klausimai turi spręsti dalykinės srities problemas / uždavinius, naudotojams pateikti aktualią informaciją. Sudaryti klausimai ontologijos įvertinimo metu virsta kokybiniai įverčiai, pagal kuriuos gali įvertinti sistemos ar modelio kokybę.

Kompetencijos klausimus galime apibūdinti kaip neformalius klausimus, pvz., „Kokie įmonėje yra darbuotojų?“, „Koks prekių likutis sandėlyje?“ ir pan. Sudaryti klausimai, ontologijos projektavimo ir vystymo stadijose gali būti tikslinami. Sudarant svarbu įvertinti galimybę, kad iš atskirų klausimų būtų galima sudaryti kompleksinis klausimų junginiu, t. y. būtų galima taikyti sankirtos / sąjungos operacijas.

Ontologijų inžinerijoje, kompetencijos klausimų sudarymo etapą galima sulyginti su programinės įrangos reikalavimų specifikavimo etapu. Klausimai, pasirinktai dalykinei sričiai pateikiami darbo 4.2 punkte.

3.1.3. Ontologijos projektas

Ontologijos projektavimo stadijoje, remiantis dalykinės srities aprašu ir surašytais kompetencijos klausimais parengiamas projektas. Projektą yra patogu išreikšti klasių diagrama, kurioje išvardijami konceptai, jų egzemplioriai ir sudaromi ryšiai tarp. Taip pat, numatomi apribojimai, aksiomos. Projektas leidžia dar detaliau ištirti dalykinę sritį. Ontologijos projektas pateiktas 4.4. ir 4.4.1. punktuose.

3.1.4. Ontologijos įgyvendinimas

Šiame etape remiantis sudarytu klasių modeliu sudaroma ontologija. Pirmiausiai yra išvardijamos klases ir sudaroma jų hierarchija. Klases dar galime suskirstyti į pirmines klases ir išvedamos iš kitų klasių, tam naudojant ontologijų aksiomas bei taisykles. Kitu žingsniu sukuriami klasių egzemplioriai. Klasių egzemplioriams yra taikomos ontologijų taisyklių išvedimo mechanizmai. Sudarant ontologiją svarbu nuolat tikrinti jos neprieštaringumą / teisingumą. Tam naudojami įrankiai, tokie kaip *Protege ir* kiti. Kiekviena klasė turi turėti bent po vieną egzempliorių.

Pateikto proceso 30 pav. 4 punkto aprašymas pateikiamas darbo 5.2. punktu. Proceso 4.1. punkto aprašymas pateiktas 5.3. darbo punktu. Proceso 5 ir 5.1. punktai pateikti darbo 6.3.1., 6.3.2. ir 6.3.3. punktais.

4. Reliacinių duomenų bazių duomenų integravimas taikant ontologijas reikalavimų specifikacija ir projektas

4.1. Reikalavimai duomenų šaltinių integravimo sistemai

Šiame poskyryje sudaryti reikalavimai integruojamų duomenų šaltinių paruošimui integracijai, sudaromos ontologijos, sprendžiami schemų semantiniai konfliktai. Duomenų integravimo sistema nebus kuriama nuo pradžių, sistemos architektūrai yra parinktos sistemos ir komponentai, kurie turi leisti susieti ontologijos konceptus su atitinkamais reliacinės duomenų bazės konceptais, vykdyti semantines užklausas, automatizuotai generuoti duomenų bazių ir ontologijos modelį. Modelis turi būti plečiamas pagal poreikį. Naudotojui turi būti sukurta taikomoji programinė įranga, kuri leistų gauti duomenis iš modelyje aprašytų duomenų šaltinių. Sistemos panaudojimo atvejų diagrama pateikta pav. Nr. 31.

Pagrindiniai sistemos kūrimo ir vystymo aktoriai yra analitikas, dalykinės srities žinovas, ontologijų inžinierius, projektuotojas ir vartotojas.

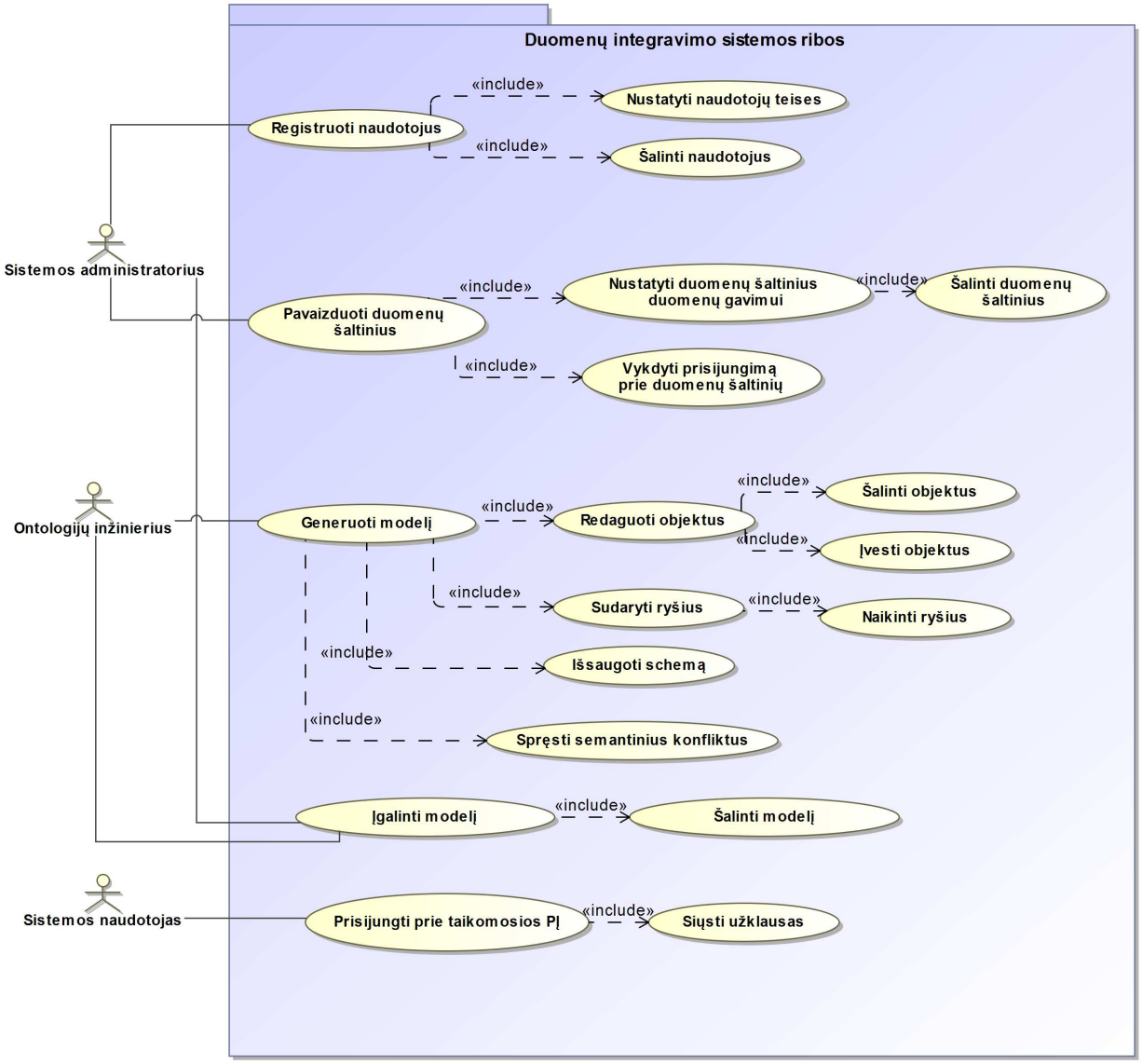
Analitikas – atlieka dalykinės srities analizę, sudaro reikalavimų specifikaciją.

Dalykinės srities ekspertas – padeda ontologijos projektuotojui sudarant ontologiją, tai gali būti nebūtinai informacinių sistemų srities specialistas.

Ontologijų inžinierius – projektuoja ir įgyvendina ontologija, dalykinės srities klausimais konsultuojasi su dalykinės srities žinovu.

Projektuotojas – parengia projekto dokumentaciją.

Vartotojas – naudojasi integruotų duomenų šaltinių resursais, siunčia užklausas į duomenų šaltinius.



31 pav. Sistemos panaudojimo atvejų diagrama

4.1.1. Panaudojimo atvejų diagrama

Panaudojimo atvejų diagramoje pavaizduotos sistemos aktorių vykdomos funkcijos bei ryšiai tarp sistemos aktorių.

4.1.2. Panaudojimo atvejų specifikacijos

Lentelė 10. PA Registruoti naudotojus“ specifikacija

Reikalavimas#: 1	Panaudojimo atvejis *„Registruoti naudotojus“*
Aprašymas	**Sistema turi kurti naudotojų sąskaitas**
Pagrindimas	Sistema turi kurti sąskaitas įmonės naudotojams, prisijungimui prie taikomosios PĮ, kad prie sistemos galėtų prisijungti tik autentifikuoti naudotojai ir tokiu būdu būtų užtikrintas informacijos konfidencialumas.
Aktorius	Sistemos administratorius

Sistema	Užklausų pateikimo į duomenų šaltinius įrankis		
Tikimo kriterijus	Kiekviename darbui su programa žingsnyje turi būti numatyta galimybė grįžti į prieš tai buvusią sistemos būseną bei ištaisius padarytas klaidas ir atlikus neteisingą veiksmą užtikrinti, kad nebūtų sugadinti duomenys ar sutriktų pats sistemos funkcionavimas. Darbiniame programos lange turi būti rodoma tik ta informacija, kuri reikalinga tam tikros funkcijos atlikimui. Išvedami pranešimai ar pateikiami rezultatai turi būti naudotojui suprantami be papildomų paaiškinimų. Vartotojo atliekami veiksmai, turi būti vizualiai patvirtinti, tam, kad vartotojas žinotų, kad jo pasirinkta komanda buvo įvykdyta/neįvykdyta.		
Prieš sąlygą	Administratorius atidaro naudotojų registravimo formą		
Po sąlygos	Suvedami naudotojo duomenys, sukuriama prisijungimo sąskaita		
Alternatyvos			
Neteisingai suvedus duomenis	Sistema pateikia sistemos pranešimas apie klaidos tipą ir jos kodą.		
Užsakovo tenkinimas: 5	Užsakovo netenkinimas: 4		
Priklausomybės	Nėra	Konfliktai	Nėra
Papildoma medžiaga	Nėra		
Istorija	Užregistruotas 2012 m. kovo 1 d.		

Lentelė 11. PA Pavaizduoti duomenų šaltinius“ specifikacija

Reikalavimas#: 2	Panaudojimo atvejis *„Pavaizduoti duomenų šaltinius“*
Aprašymas	**Sistema turi automatizuotai pavaizduoti duomenų šaltinius, sugeneruoti modelį**
Pagrindimas	Sistema turi teikti galimybę generuoti duomenų pavaizdavimo modelį, pagal reliacinių duomenų bazių schemas.
Aktorius	Analitikas, administratorius
Sistema	Semantinių technologijų sistema
Tikimo kriterijus	Kiekviename darbui su programa žingsnyje turi būti numatyta galimybė grįžti į prieš tai buvusią sistemos būseną bei ištaisius padarytas klaidas ir atlikus neteisingą veiksmą užtikrinti, kad nebūtų sugadinti duomenys ar sutriktų pats sistemos funkcionavimas. Darbiniame programos lange turi būti rodoma tik ta informacija, kuri reikalinga tam tikros funkcijos

	atlikimui. Išvedami pranešimai ar pateikiami rezultatai turi būti naudotojui suprantami be papildomų paaiškinimų. Vartotojo atliekami veiksmai, turi būti vizualiai patvirtinti, tam, kad vartotojas žinotų, kad jo pasirinkta komanda buvo įvykdyta/neįvykdyta.		
Prieš sąlygą	Naudotojas inicijuoja duomenų bazių susiejimo posistemę.		
Po sąlygos	Sistema pateikia naudotojui interaktyvų vedlį, kurio pagalba atliekamas duomenų šaltinių pasirinkimas bei vaizdavimo funkcijos.		
Alternatyvos:			
Nepavyko prisijungti prie duomenų šaltinių	Sistema pateikia sistemos pranešimas apie klaidos tipą ir jos kodą.		
Užsakovo tenkinimas: 5	Užsakovo netenkinimas: 4		
Priklausomybės	Nėra	Konfliktai	Nėra
Papildoma medžiaga	Nėra		
Istorija	Užregistruotas 2012 m. kovo 1 d.		

Lentelė 12. PA Prisijungti prie naudotojo taikomosios PĮ specifikacija

Reikalavimas#: 3	Panaudojimo atvejis *„Prisijungti prie naudotojo taikomosios PĮ“*
Aprašymas	**Sistema turi leisti prisijungti prie taikomosios PĮ, užklausoms siųsti**
Pagrindimas	Sistema turi leisti naudotojams, prisijungimui prie taikomosios PĮ. Prisijungimo metu naudotojai turi būti autentifikuojami.
Aktorius	Analitikas, administratorius, naudotojas
Sistema	Užklausų pateikimo į duomenų šaltinius įrankis
Tikimo kriterijus	Kiekviename darbui su programa žingsnyje turi būti numatyta galimybė grįžti į prieš tai buvusią sistemos būseną bei ištaisius padarytas klaidas ir atlikus neteisingą veiksmą užtikrinti, kad nebūtų sugadinti duomenys ar sutriktų pats sistemos funkcionavimas. Darbiniame programos lange turi būti rodoma tik ta informacija, kuri reikalinga tam tikros funkcijos atlikimui. Išvedami pranešimai ar pateikiami rezultatai turi būti naudotojui suprantami be papildomų paaiškinimų. Vartotojo atliekami veiksmai, turi būti vizualiai patvirtinti, tam, kad vartotojas žinotų, kad jo pasirinkta komanda buvo įvykdyta/neįvykdyta.
Prieš sąlygą	Atidaryta naudotojo prisijungimo forma
Po sąlygos	Naudotojas suveda naudotojo vardą ir slaptažodį, sistema autentifikuoja

	naudotoją ir leidžia naudotis resursais.		
Alternatyvos:			
Nepavyko prisijungti	Sistema pateikia sistemos pranešimas apie klaidos tipą ir jos kodą.		
Užsakovo tenkinimas: 5	Užsakovo netenkinimas: 4		
Priklausomybės	Nėra	Konfliktai	Nėra
Papildoma medžiaga	Nėra		
Istorija	Užregistruotas 2012 m. kovo 1 d.		

4.2. Dalykinės srities analizė

Prieš dalykinės srities konceptualizavimo procesą atliekama integruojamų duomenų šaltinių semantikos analizė. Identifikuojami duomenų šaltinių schemų elementų skirtumai, kurie įvardijami kaip konfliktais. Pav. Nr. 32, 33, 34. pateikti nustatyti schemų elementų konfliktai. Detalios schemos pateiktos prieduose.

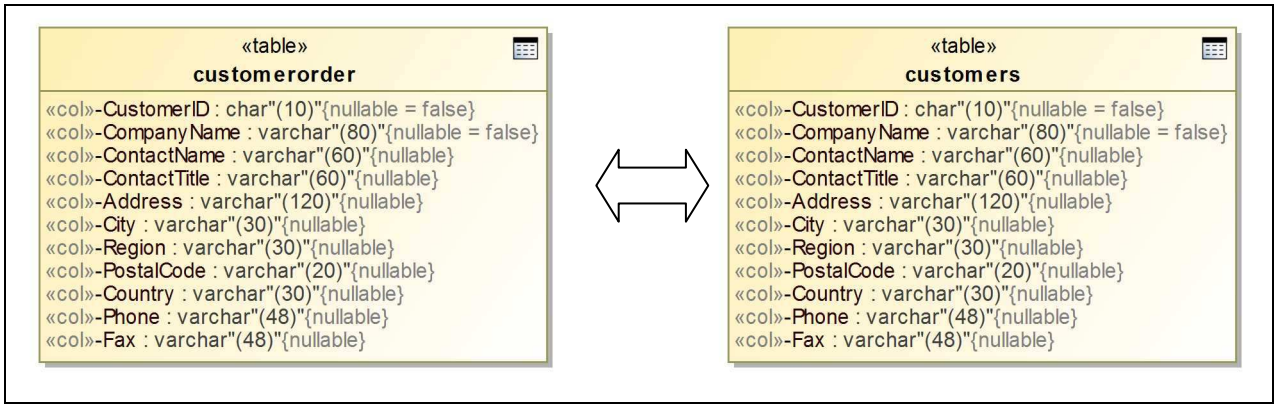
Konfliktų detalizavimas:

32 pav., lentelių pavadinimai nesutampa: vienoje pavadintas *„customerorder“*, o kitoje *„customers“*;

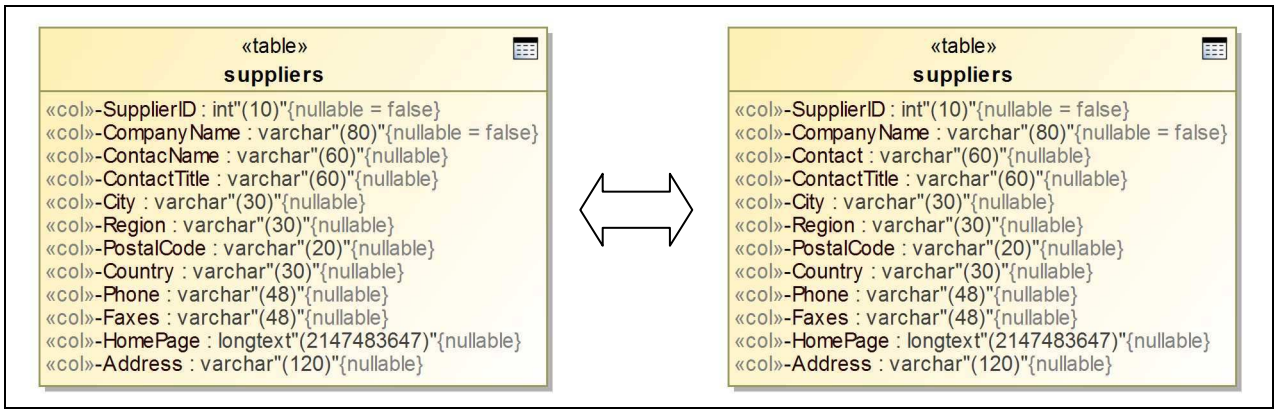
33 pav. atributų pavadinimai nesutampa: vienoje *„ContactName“*, o kitoje *„Contact“*;

34 pav., atributų pavadinimai nesutampa: vienoje *„LastName“*, o kitoje *„SurName“*.

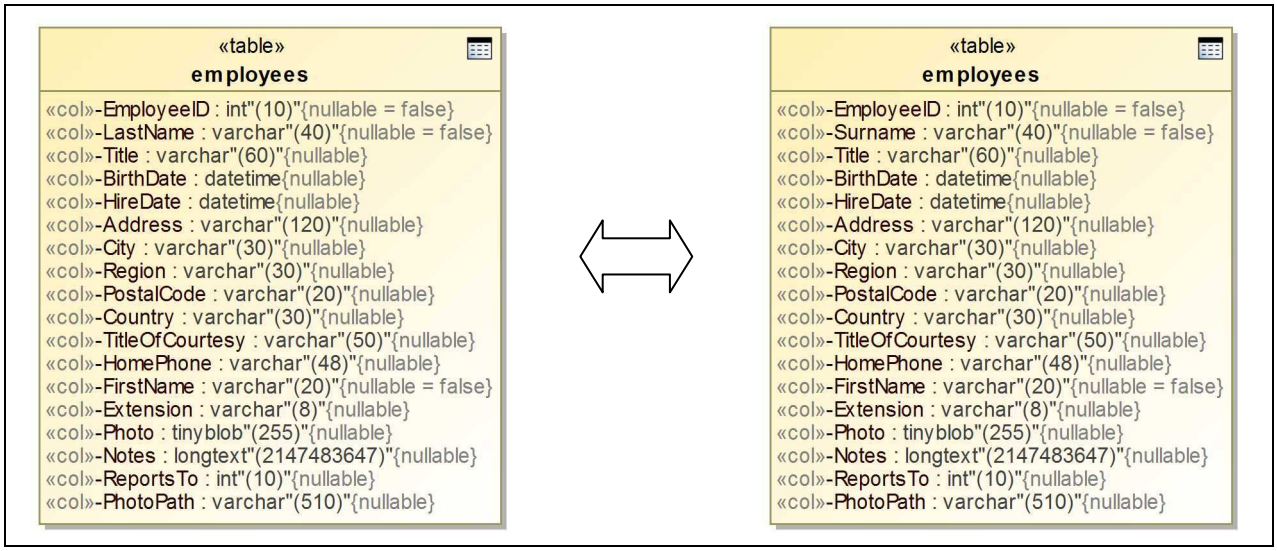
Visais atvejais konfliktai turi būti sprendžiami, tam, kad išvengtume konfliktų modelio naudojimo metu. Sprendimo variantai bus aptarti kituose skyriuose.



32 pav. Schemos konfliktai



33 pav. Schemos konfliktai



34 pav. Schemos konfliktai

4.3. Reikalavimai modeliui

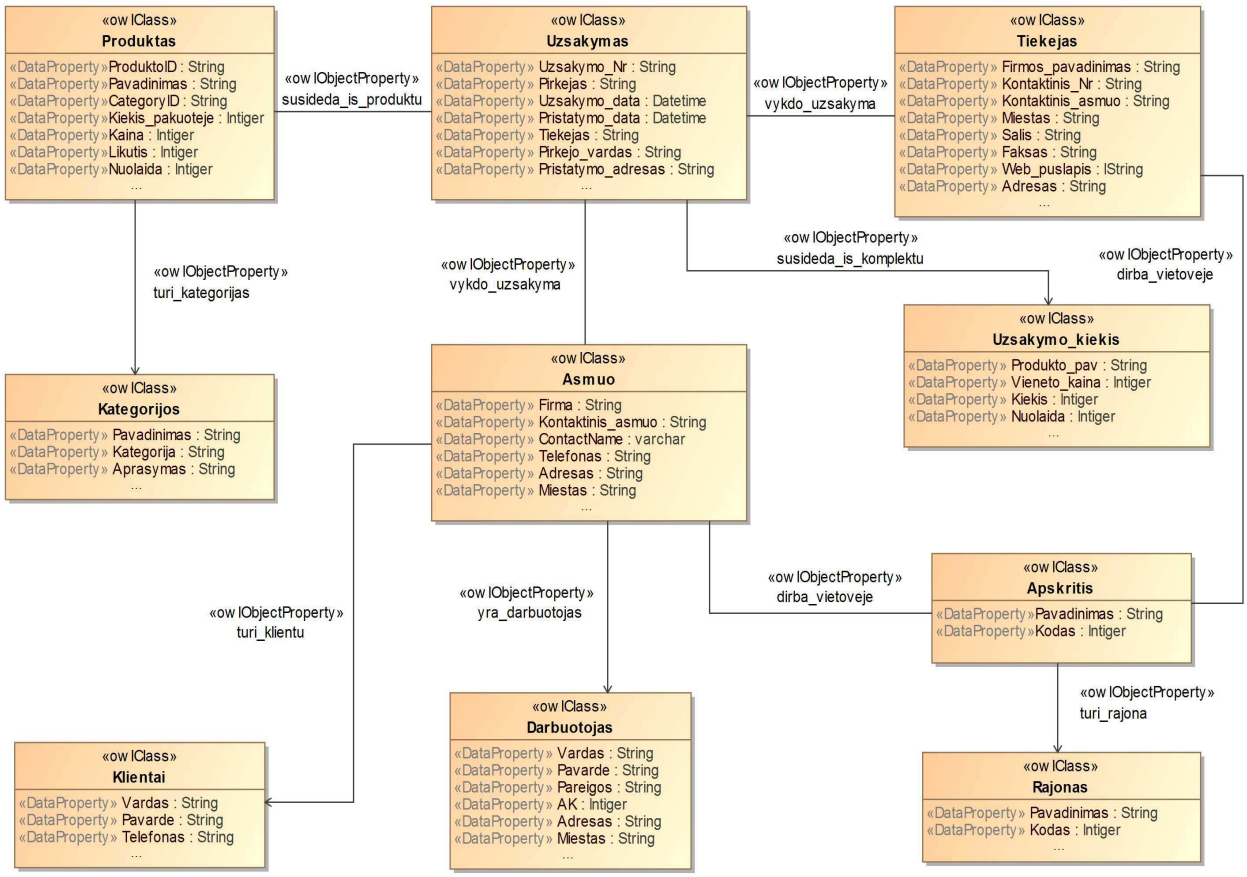
Šiame etape būtina sudaryto reikalavimus modeliui, tam surašomo kompetencijos klausimai, į kuriuos sistema turės atsakyti. Atlikę dalykinės srities analizės – galime sudaryti kompetencijos klausimus.

Kompetencijos klausimai:

1. Kas yra įmonės darbuotojai ir pateikti jų kontaktus.
2. Koks yra prekių likutis sandėlyje?
3. Kas yra įmonės tiekėjai?
4. Su kokiais logistikos centrais įmonė yra sudariusi kontraktą?
5. Kas yra įmonės klientai ir pateikti jų kontaktus.

4.4. Dalykinės srities ontologijos projektas

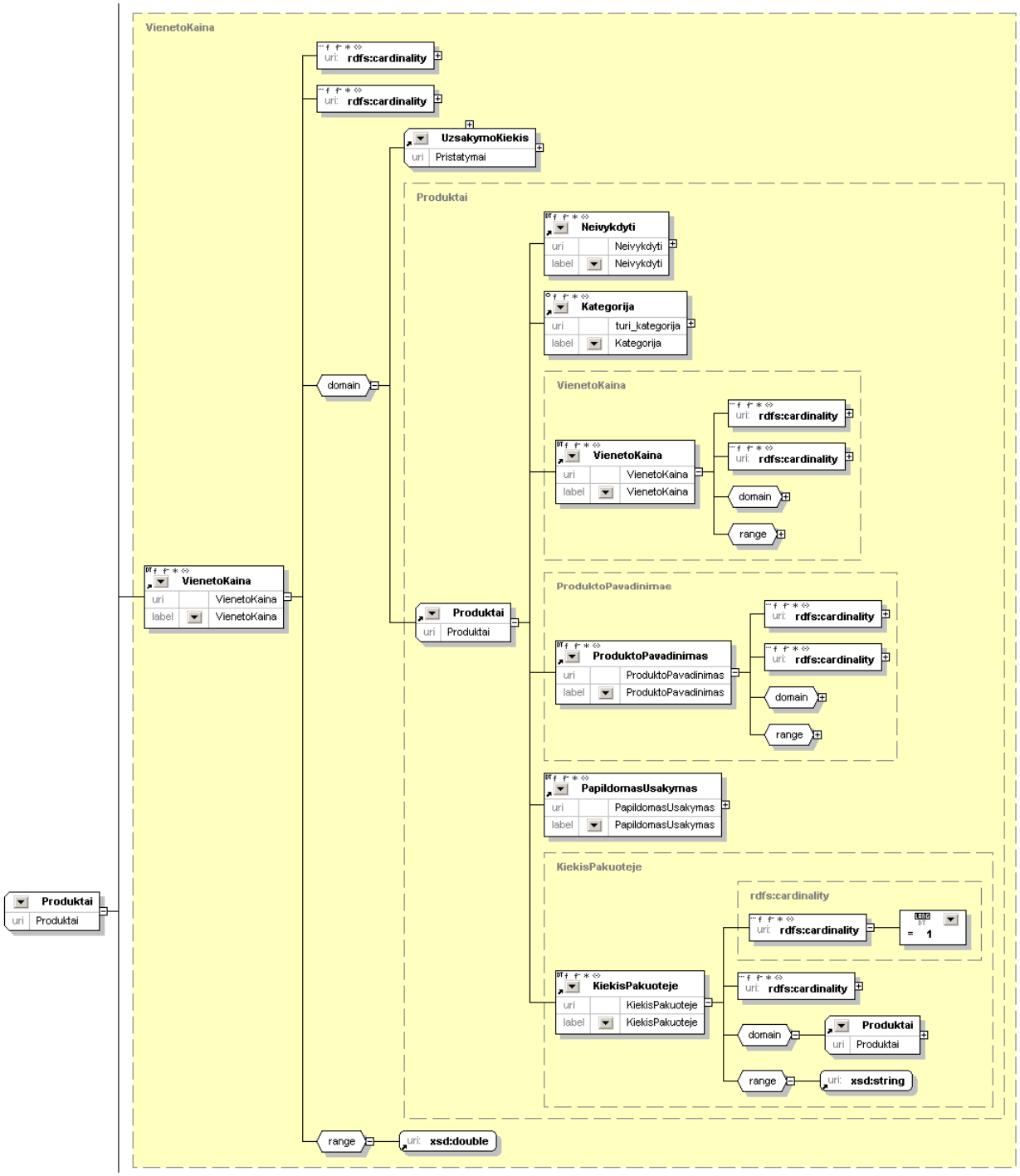
Šiame proceso etape sudaroma globali ontologija pav., 35, tam atliekama integruojamų šaltinių analizė, nustatomas jų schemų semantika. Procesas pradedamas nuo reliacinių duomenų bazių schemos automatizuoto transformavimo į ontologiją. Transformuoti elementai perkeliami į ontologijų redaktorių, kuriuo perskirstomos klasės, sudaroma hierarchija.



35 pav. Ontologijos klasių modelis

4.4.1. Dalykinės srities ontologijos modelis

Dalykinės srities klasių diagrama (36 pav.) pateikiamos analizės metu ištirtos esybės, kurios panaudotos sudarant dalykinės srities modelį. Šis modelis vėlesnėje projektavimo stadijoje bus transformuojamas į duomenų bazės schemą.



36 pav. Ontologijos diagrama

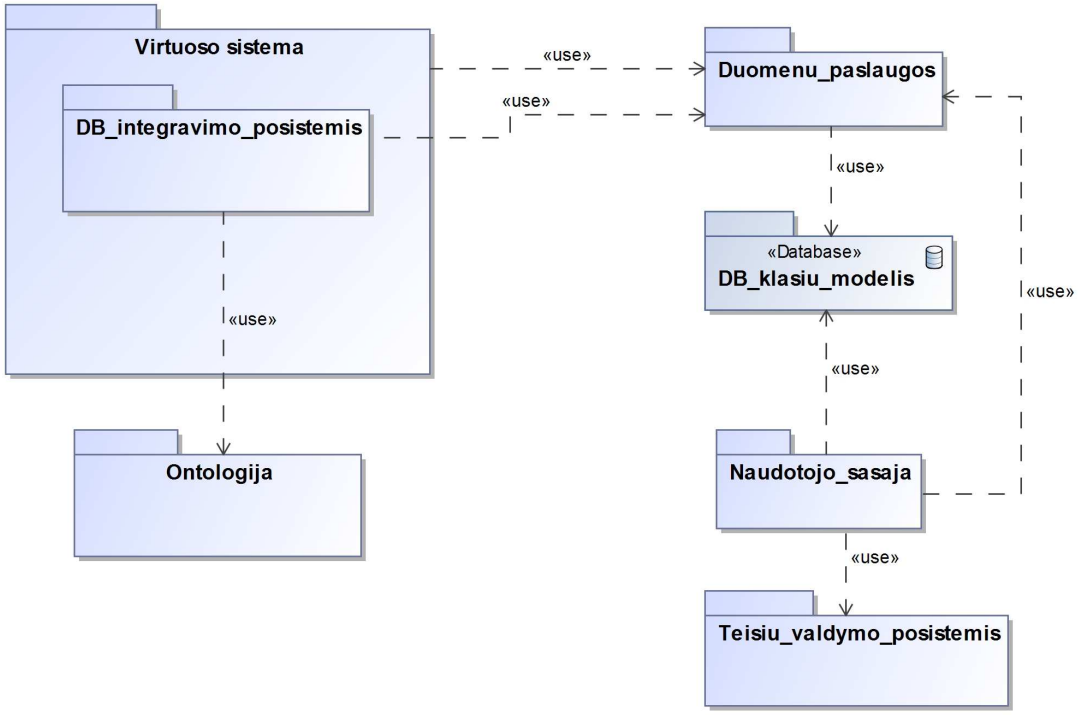
4.5. Sistemos architektūra

Virtuoso sistema vykdo atvaizdavimą tarp duomenų šaltinių ir globalios ontologijos bei leidžia vykdyti semantines užklausas. Sistemos architektūra pateikta 37 pav. *Ši* sistema leidžia pasirinkti ontologijas, duomenų bazių tarnybines stotis, derinti prisijungimus bei atlikti kitas derinimo funkcijas. *Virtuoso* modelių valdymo modulis, *Views* ir *R2RML* vykdo duomenų šaltinių, reliacinių duomenų bazių schemą transformaciją į ontologijų objektus bei sukuria modelį, kuriame aprašomi visi susieti duomenų šaltiniai. Taip pat apdoroja SPARQL užklausas, kurias transformuoja į atitinkamas SQL užklausas ir siunčia į duomenų modelyje pavaizduotus

duomenų šaltinius. Schemoje šie moduliai nedetalizuojami, jie yra *Virtuoso* sistemos sudedamosios dalys.

Virtuoso sistema turi integruotą grafinę sąsają, kuri pasiekiama per interneto naršyklę, per šią sąsają galima konfigūruoti sistemą, įkelti ontologijas, valdyti konfliktus ir siųsti užklausoms į duomenų šaltinius. Ši sąsaja skirta tik administratoriams.

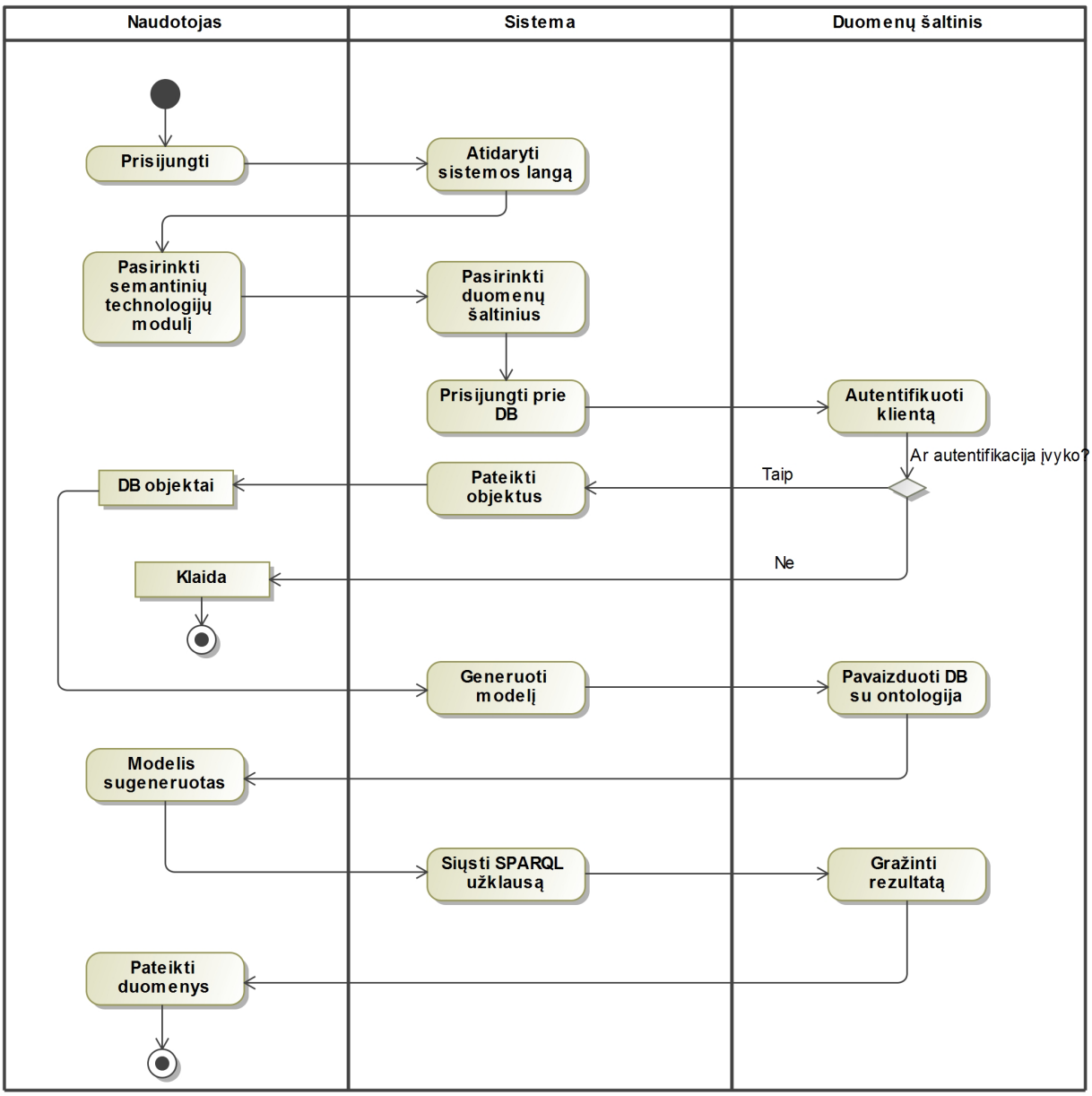
Atskirai pavaizduotas naudotojo sąsajos modulis skirtas duomenims gauti, jis leidžia siųsti užklausas į duomenų šaltinius.



37 pav. Sistemos architektūros diagrama

4.6. Sistemos elgsenos modelis

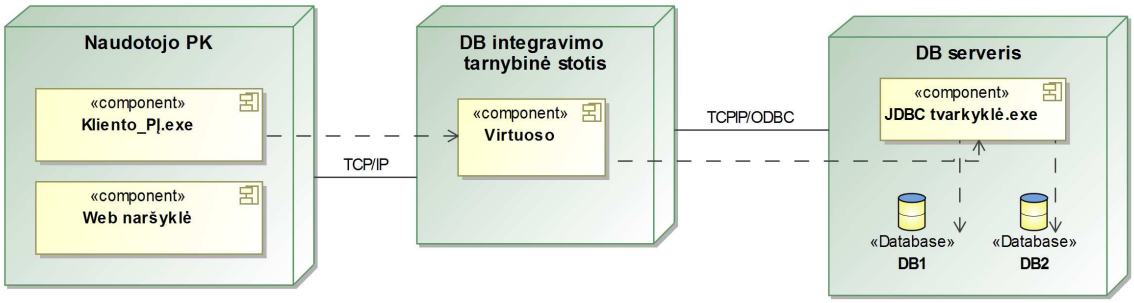
Sistemos naudotojo veiksmų seka naudojant integruotus duomenų šaltinius pavaizduota veiklos diagrama paveiksle 38 pav. Naudojimasis sistema susideda iš paruošiamojo etapo ir naudojimosi. Pirmiausiai sugeneruojamas modelis, tam naudotojas prisijungia prie sistemos, nurodo duomenų šaltinius, kuriems sukuriamas pavaizdavimo (modelis) dokumentas. Šios operacijos metu duomenų šaltiniai yra susiejami su ontologija. Po šio etapo naudotojas gali pateikti SPARQL užklausas į ontologijoje aprašytus duomenų šaltinius.



38 pav. Sistemos elgsenos diagrama

4.7. Diegimo diagrama

Sudaryta diegimo diagrama, diagramoje 39 pav. pateikiami komponentai ir jų pasiskirstymas po fizinius įrenginius.



39 pav. Sistemos diegimo diagrama

5. Duomenų integravimo sistemos realizacija

Iš atliktos probleminės srities analizės: apžvelgtų egzistuojančių sprendimų, ontologijų metodų, algoritmų bei karkasų – sudarytas konfliktų sprendimą realizuojančios programinės įrangos projektas, kuris leis automatizuotai integruoti skirtingas duomenų bazes, tam taikant bendrą modelį. Modelis skirtas vykdyti schematinį integravimo metodą.

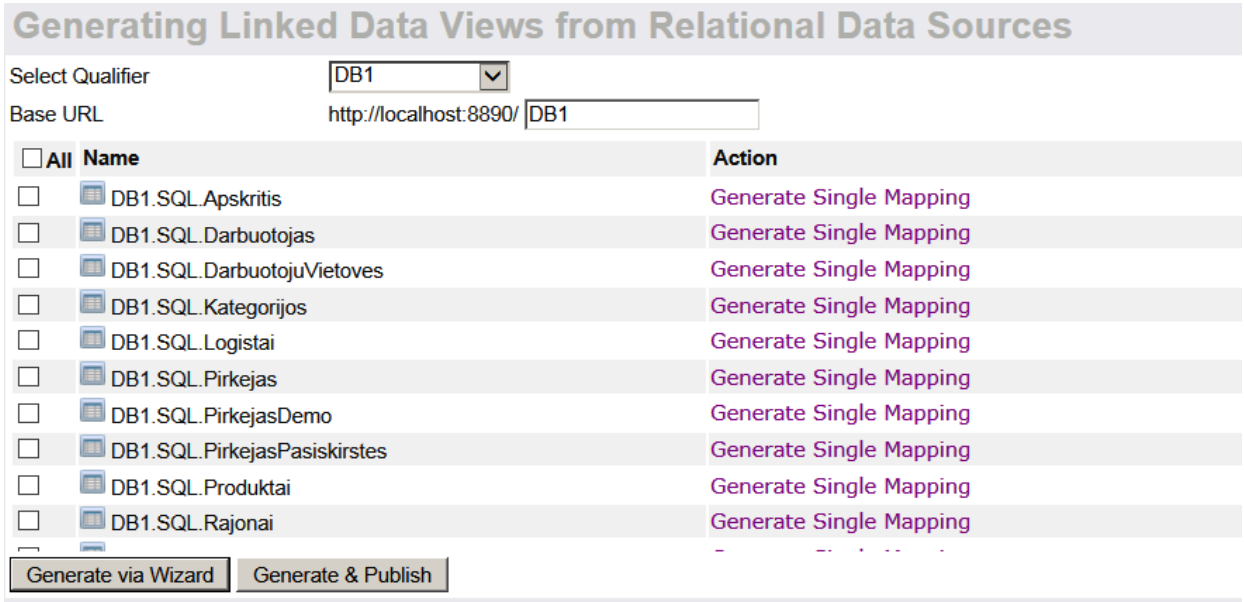
5.1. Realizacijos technologinės priemonė

Sistemos realizavimui pasirinkta *Virtuoso* platforma. Eksperimentas atliktas su 07.00.3202 šios sistemos versija. Pagrindinės priežastys lėmusios šį pasirinkimą – suderinamumas su *.Net* produktais, *Microsoft Visual Studio, Silver Lite* technologijomis.
Duomenų bazių sistemos pasirinktos Microsoft SQL Server 2008 ir MySQL 5.6 v. Ontologijų redagavimo įrankis *Protege* ir *Altova SemanticWorks.*

5.2. Ontologijos realizacija

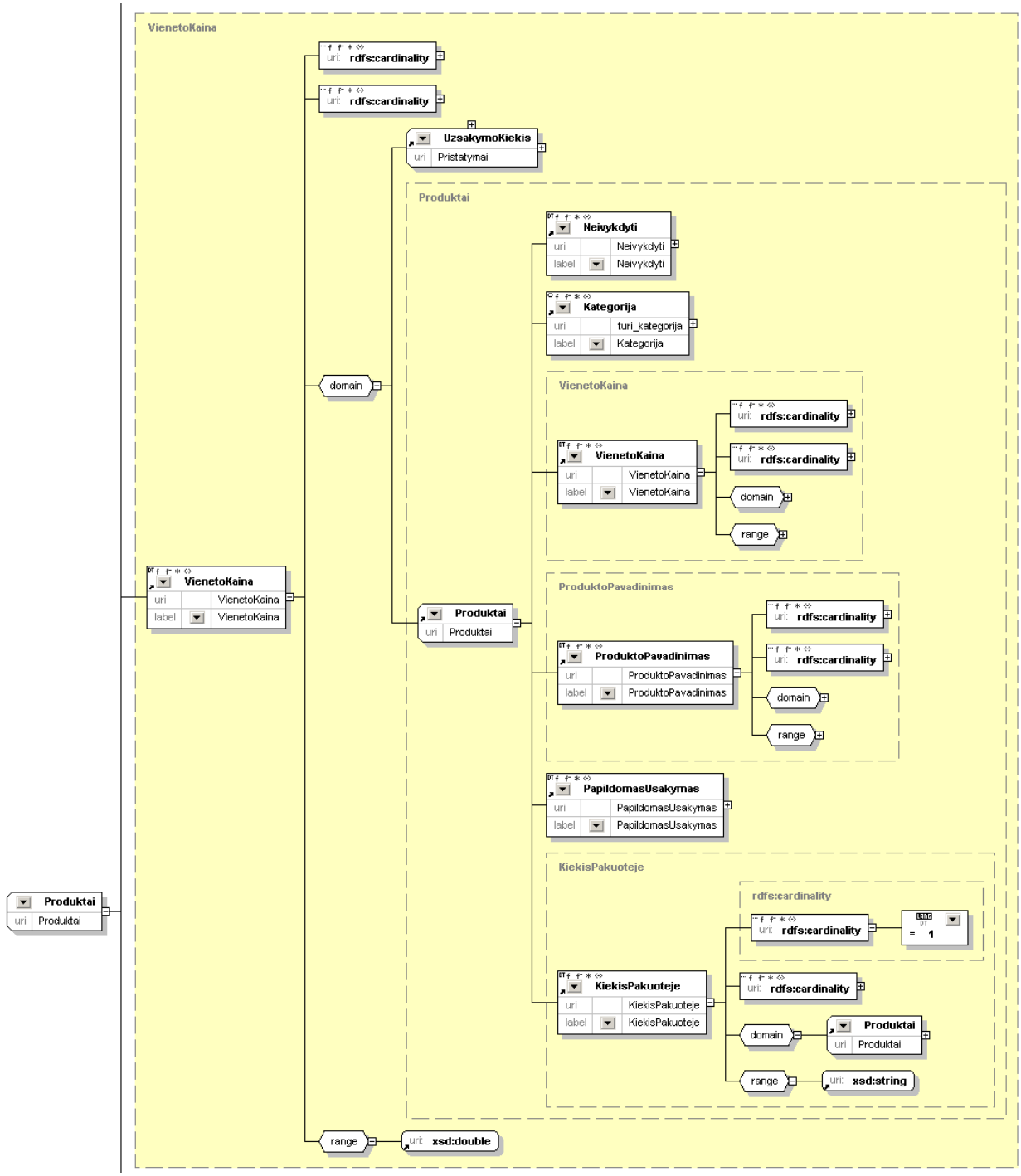
Realizacijai pasirinkta Microsoft platinama *Northwind* duomenų bazė. Projektinėje dalyje buvo ištirta dalykinės sritis ir pagal ją sudarytos klasės, objektų duomenų ypatybės. Ontologijos projektavimo metu naudoti išvadų – teisingumo užtikrinimo varikliai Palete ir FaCT++.

Ontologijos klasės, objektų duomenų ypatybės sugeneruoto naudojant *Virtuoso Views* komponentą. Sugeneruoti konceptai perkelti į *Protege* sistemą ir sutvarkyti pagal dalykinės srities tyrimo rezultatus. *Virtuoso* generatoriaus langas pateiktas 40 pav.



40 pav. Virtuoso langas

Sugeneruotos ir pataisytos ontologijos fragmentas pateiktas 41 pav.

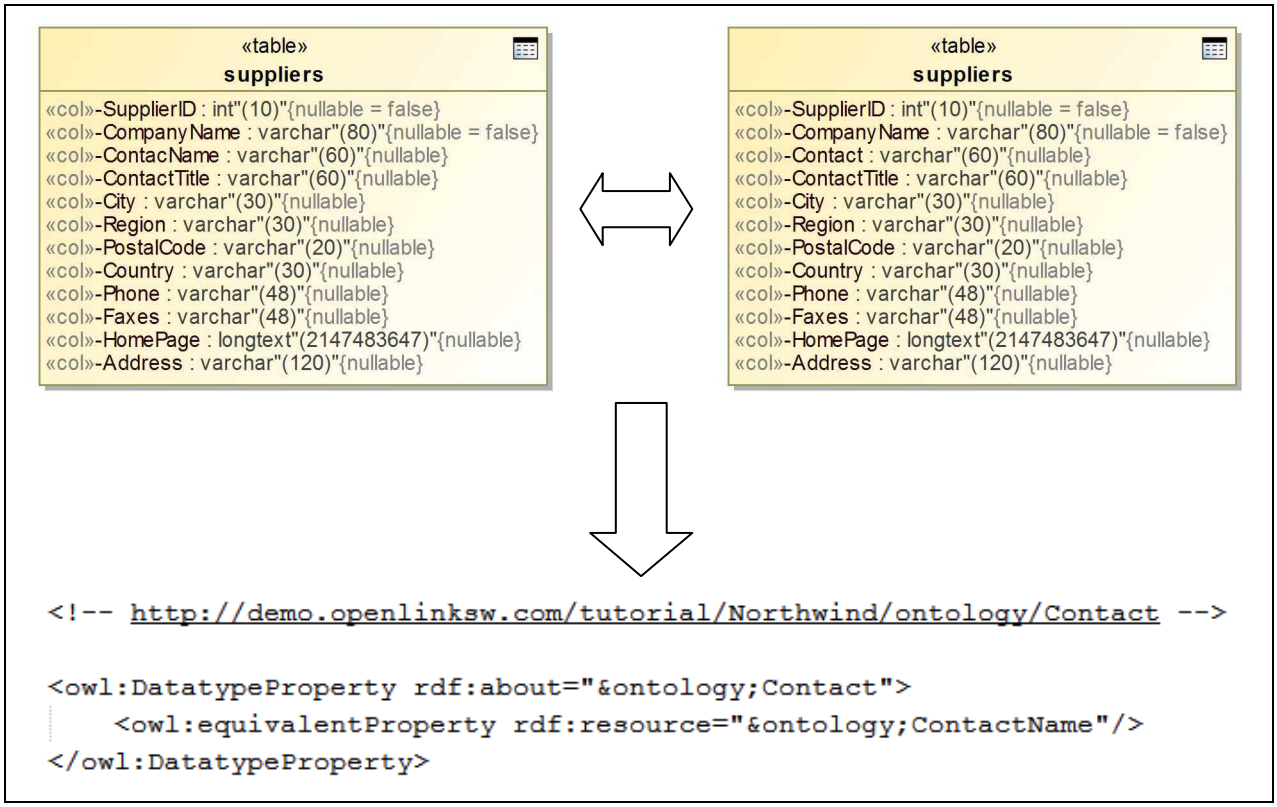


41 pav. Ontologijos schema

5.3. Konfliktų šalinimas

Tyrimo dalyje nustatyta, kad integruojamų duomenų šaltinių DB1 ir DB2 schemos nėra tolygios, todėl buvo pritaikyti metodai jų pašalinimui. Sprendimai pateikti 42, 43 paveikslėliuose.

42 pav. Schemos konfliktai



43 pav. Schemos konfliktai

6. Eksperimentinis integravimo metodikos įvertinimas

6.1. Eksperimento planas

Sudarytos proceso ir metodikos efektyvumui įvertinti buvo pasirinkta globalios schemos ontologijų architektūra, integruojamiems duomenų šaltiniams apjungti įgyvendinta globali ontologija. Globalia ontologija pavaizduoti dviejų eksperimentui pasirinktų duomenų bazių DB1 ir DB2 schemos. Integruojamos duomenų bazės yra panašios, tačiau yra esminių skirtumų, dėl kurių reikėjo taikyti *OWL* savybes, schematiniams konfliktams spręsti. Sudarytos ontologijos publikavimui pasirinkta *Virtuoso* semantinių paslaugų sistema. Pavaizdavimas tarp ontologijos ir reliacinių duomenų bazių atliktas *D2RDB* vaizdavimo kalba. Ontologijos teisingumui užtikrinti, taikyti analizės dalyje sudaryti kompetencijos klausimai. *SPARQL* užklausoms siųsti naudotas *Virtuoso SPARQL* komponentas ir realizavimo etape sukurta taikomoji programinė įranga.

6.2. Eksperimento priemonės

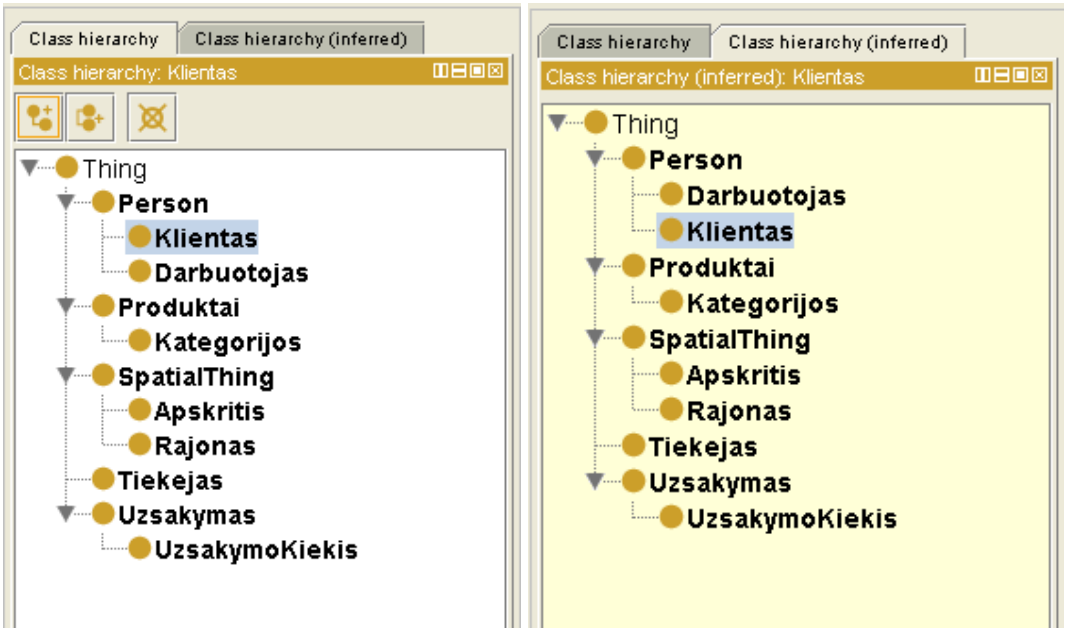
Eksperimentui naudotas MS SQL Server 2008 R2, įdiegtas operacinėje sistemoje Windows 7 64 bit., JDBC tvarkyklė; Techinės įrangos parametrai: mikroprocesorius Intel Core 2 3 GHz; operatyvioji atminti - 8 Gb; standusis diskas – SSD 128 Gb.

6.3. Eksperimentas

Eksperimento uždavinys – patikrinti sudarytą duomenų modelį. Tam naudojamos *Virtuoso* sistemos *SPARQL* komponentas. Kuris leidžia siųsti užklausas į duomenų modelyje aprašytus šaltinius.

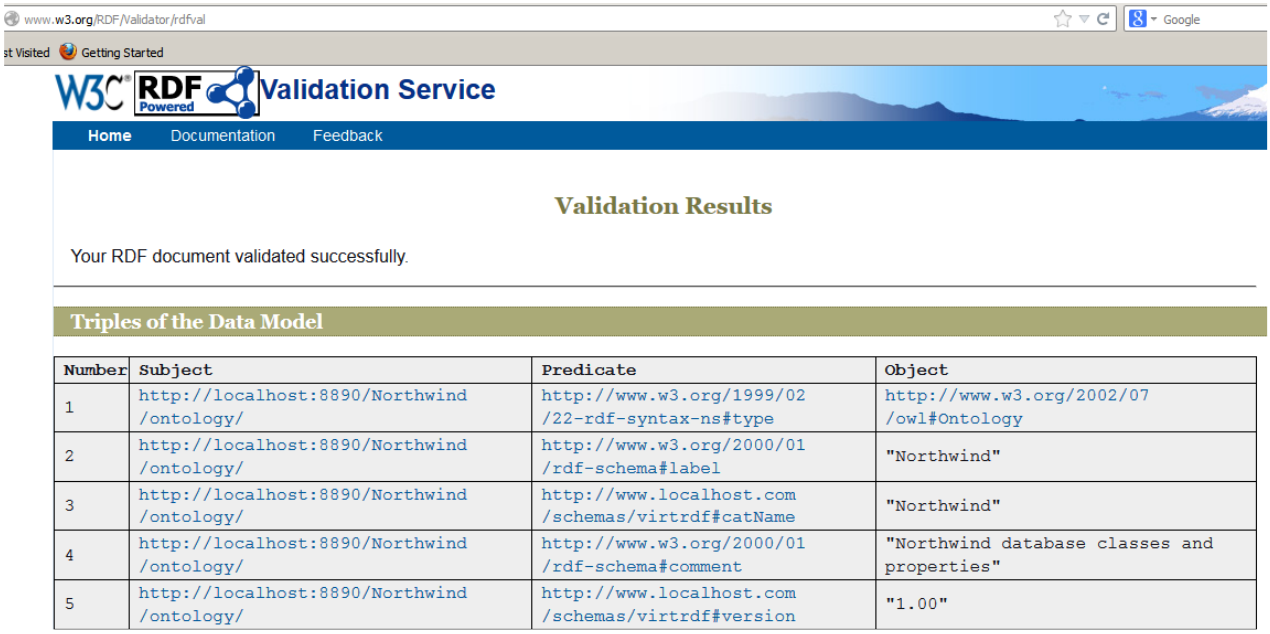
6.3.1. Bazinės ontologijos testavimas

Bazinė dalykinės srities ontologija turi atsakyti į sudarytus kompetencijos klausimus. Remiantis sudarytais kompetencijos klausimais, sukurta bazinė ontologija. Ontologijos konstravimui naudoti *Protege* ir *Altova Semantic Works* įrankiai. Ontologijai keliamų reikalavimų užtikrinimui, jos sudarymo ir vystymo metu buvo naudoti ontologijų nepriekaištingumo tikrinimo mechanizmai *FaCT++, Hermit.* Buvo palygintos minėtų mechanizmų išvedamos klasės ir jų hierarchija (44 pav.). Ontologijos pirminės klasės sutampa su išvedamomis, tuo pažymima, kad klasių hierarchija yra teisinga.



44 pav. Išvedamų klasių hierarchija

Ontologijos aprašo teisingumas patikrintas *W3C RDF* teisingumo patvirtinimo servisu (*angl. Validation Results*). Patikrinimo rezultatai parodė, kad įgyvendinta ontologija atitinka *W3* konsorciumo rekomendacijas. Rezultatai pateikti 45 pav.

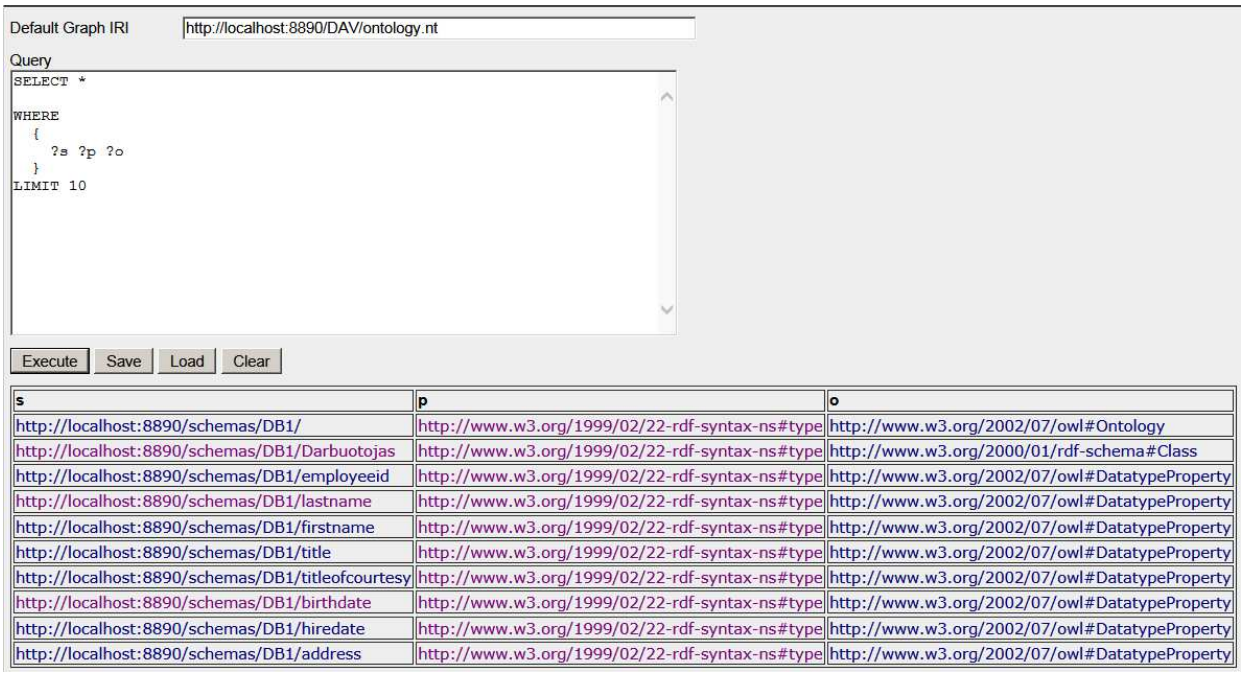


Number	Subject	Predicate	Object
1	http://localhost:8890/Northwind /ontology/	http://www.w3.org/1999/02 /22-rdf-syntax-ns#type	http://www.w3.org/2002/07 /owl#Ontology
2	http://localhost:8890/Northwind /ontology/	http://www.w3.org/2000/01 /rdf-schema#label	"Northwind"
3	http://localhost:8890/Northwind /ontology/	http://www.localhost.com /schemas/virtrdf#catName	"Northwind"
4	http://localhost:8890/Northwind /ontology/	http://www.w3.org/2000/01 /rdf-schema#comment	"Northwind database classes and properties"
5	http://localhost:8890/Northwind /ontology/	http://www.localhost.com /schemas/virtrdf#version	"1.00"

45 pav. Išvedamų klasių hierarchija

6.3.2. Duomenų modelio teisingumo užtikrinimas

Duomenų modelio teisingumui tikrinti buvo sudaryto SPARQL užklausos, atsakančios į kompetencijos klausimus. Užklausos į visus duomenų rinkinius pateiktos 46 pav.



s	p	o
http://localhost:8890/schemas/DB1/	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Ontology
http://localhost:8890/schemas/DB1/Darbuotojas	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2000/01/rdf-schema#Class
http://localhost:8890/schemas/DB1/employeeid	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#DatatypeProperty
http://localhost:8890/schemas/DB1/lastname	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#DatatypeProperty
http://localhost:8890/schemas/DB1/firstname	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#DatatypeProperty
http://localhost:8890/schemas/DB1/title	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#DatatypeProperty
http://localhost:8890/schemas/DB1/titleofcourtesy	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#DatatypeProperty
http://localhost:8890/schemas/DB1/birthdate	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#DatatypeProperty
http://localhost:8890/schemas/DB1/hiredate	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#DatatypeProperty
http://localhost:8890/schemas/DB1/address	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#DatatypeProperty

46 pav. SPARQL resursų IRI vaizdavimas

6.3.3. Kompetencijos klausimai

Sudarytas duomenų modelis turi atsakyti į ontologijos projektavimo etape sudarytus kompetencijos klausimą. Šiame etape atliktas atsakymų į sudarytus kompetencijos klausimus įvertinimas. Kiekvienam kompetencijos klausimui buvo sudarytos užklauso, duomenims gauti. Duomenų rinkinių pavyzdžiai pateikiami kiekvienam klausimui atskirai.

1. Kas yra įmonės darbuotojai ir pateikti jų kontaktus.
2. Koks yra prekių likutis sandėlyje?
3. Kas yra įmonės tiekėjai?
4. Su kokiais logistikos centrais įmonė yra sudariusi kontraktą?
5. Kas yra įmonės klientai ir pateikti jų kontaktus.

Sudarytas duomenų modelis turi atsakyti į klausimą, kas yra įmonės klientai ir pateikti jų kontaktus. Atsakymas į klausimą pateiktas 47 pav. Šis kompetencijos klausimas yra tenkinamas, kadangi duomenų modelis užtikrina reikiamų duomenų pateikimą naudotojui.

Attributes	Values
type	DB1.SQL.Pirkejas
Address	Obere Str. 57
City	Berlin
PostalCode	12209
Country	Germany
CompanyName	Alfreds Futterkiste
Phone	030-0074321
CustomerID	ALFKI
ContactName	Maria Anders
ContactTitle	Sales Representative
Fax	030-0076545
Relation to DB1.SQL.PirkejasDemo	http://localhost:8890/DB1/uzsakymai/OrderID/10643#this http://localhost:8890/DB1/uzsakymai/OrderID/10692#this http://localhost:8890/DB1/uzsakymai/OrderID/10702#this http://localhost:8890/DB1/uzsakymai/OrderID/10835#this http://localhost:8890/DB1/uzsakymai/OrderID/10952#this »more»
is Relation to DB1.SQL.Pirkejas of	http://localhost:8890/DB1/uzsakymai/OrderID/10643#this http://localhost:8890/DB1/uzsakymai/OrderID/10692#this http://localhost:8890/DB1/uzsakymai/OrderID/10702#this http://localhost:8890/DB1/uzsakymai/OrderID/10835#this http://localhost:8890/DB1/uzsakymai/OrderID/10952#this »more»

47 pav. Įmonės klientų duomenys

Sudarytas duomenų modelis turi atsakyti į klausimą, kas yra įmonės darbuotojai ir pateikti jų kontaktus. Atsakymas į klausimą pateiktas 48 pav. Šis kompetencijos klausimas yra tenkinamas, kadangi duomenų modelis užtikrina reikiamų duomenų pateikimą naudotojui.

Attributes	Values
type	DB1.SQL.Darbuotojas
EmployeeID	3(xsd:integer)
LastName	Leverling
FirstName	Janet
Title	Sales Representative
TitleOfCourtesy	Ms.
BirthDate	1963-08-30 00:00:00(xsd:date)
HireDate	1992-04-01 00:00:00(xsd:date)
Address	722 Moss Bay Blvd.
City	Kirkland
Region	WA
PostalCode	98033
Country	USA
HomePhone	(206) 555-3412
Extension	3355
Photo	http://localhost:8890/DB1/objects/darbuotojas/EmployeeID/3/Photo.bin

48 pav. Įmonės darbuotojų duomenys

Sudarytas duomenų modelis turi atsakyti į klausimą: „Koks yra prekių likutis sandėlyje?“. Atsakymo į klausimą pavyzdinis atvejis pateiktas 49 pav.

Attributes	Values
type	DB1.SQL.Produktai
CategoryID	1(xsd:integer)
ProductID	1(xsd:integer)
ProductName	Chai
SupplierID	1(xsd:integer)
QuantityPerUnit	10 boxes x 20 bags
UnitPrice	18(xsd:double)
UnitsInStock	39(xsd:integer)
UnitsOnOrder	0(xsd:integer)
ReorderLevel	10(xsd:integer)
Discontinued	0(xsd:integer)
Relation to DB1.SQL.Kategorijos	http://localhost:8890/DB1/kategorijos/CategoryID/1#this
Relation to DB1.SQL.Tiekejai	http://localhost:8890/DB1/tiekejai/SupplierID/1#this

49 pav. Produktai sandėlyje

Sudarytas duomenų modelis turi atsakyti į klausimą: „Kas yra įmonės tiekėjai?“. Atsakymo į klausimą pavyzdinis atvejis pateiktas 50 pav.

Attributes	Values
type	DB1.SQL.Tiekejai
Address	Order Processing Dept. 2100 Paul Revere Blvd.
City	Boston
Region	MA
PostalCode	02134
Country	USA
CompanyName	New England Seafood Cannery
Phone	(617) 555-3267
ContactName	Robb Merchant
ContactTitle	Wholesale Account Agent
SupplierID	19(xsd:integer)
Faxes	(617) 555-3389
Relation to DB1.SQL.Produktai	http://localhost:8890/DB1/produktai/ProductID/40#this http://localhost:8890/DB1/produktai/ProductID/41#this
is Relation to DB1.SQL.Tiekejai of	http://localhost:8890/DB1/produktai/ProductID/40#this http://localhost:8890/DB1/produktai/ProductID/41#this

50 pav. Įmonės tiekėjai

Sudarytas duomenų modelis turi atsakyti į klausimą: „Su kokiais logistikos centrais įmonė yra sudariusi kontraktą?“. Atsakymo į klausimą pavyzdinis atvejis pateiktas 51 pav.

Attributes	Values
type	DB1.SQL.Logistai
ShipperID	1(xsd:integer)
CompanyName	Speedy Express
Phone	(503) 555-9831
Relation to DB1.SQL.Uzsakymai	http://localhost:8890/DB1/uzsakymai/OrderID/10249#this http://localhost:8890/DB1/uzsakymai/OrderID/10251#this http://localhost:8890/DB1/uzsakymai/OrderID/10258#this http://localhost:8890/DB1/uzsakymai/OrderID/10260#this http://localhost:8890/DB1/uzsakymai/OrderID/10265#this »more»

51 pav. Logistikos įmonės

6.3.4. Konfliktų statistika

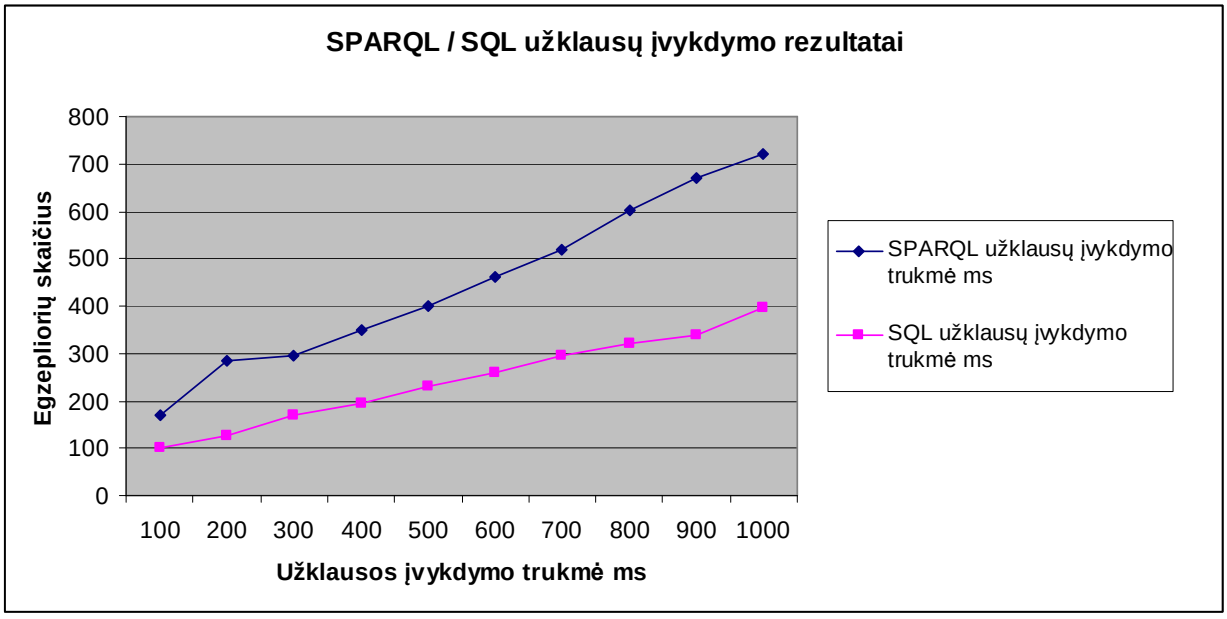
Integruojamų duomenų šaltinių schemų analizės metu buvo identifikuota vienuolika schematinio tipo konfliktų (13 lentelė), aštuoni konfliktai – sinonimai. Konfliktai buvo išspręsti naudojant *OWL* savybes. Duomenų tikslumo konfliktai nustatyti tarp datos formato, vienoje DB1 schemoje datos formatas dd.mm.yyyy, kitoje schemoje yyyy.mm.dd, šie konfliktai nebuvo išspręsti.

Konflikto tipas	Identifikuota konfliktų	Išspręsti konfliktai
Sinonimai	8	8
Antonimai	1	1
Homonimai	2	2
Duomenų tikslumo	3	0

Lentelė 13. Nustatyti ir išspręsti konfliktai

6.3.5. Funkcionavimo našumo analizė ir rezultatai

Integruotų duomenų šaltinių našumui įvertinti naudotas *Microsoft SQL Server Profiler* įrankis, kuriuo buvo įvertinta užklausų vykdymo trukmė. Atliktas *SPARQL* užklausų įvykdymo trukmės įvertinimas, naudojant sudarytą globalų modelį, kuris susietas su dviem duomenų šaltiniais *DB1* ir *DB2*. Vertinant *SQL* užklausų vykdymo trukmę, užklausos buvo siunčiamos į tuos pačius duomenų rinkinius, kaip ir *SPARQL* atveju. Užklausų vykdymo trukmės rezultatai pateikti grafike 52 pav. Nustatyta, kad *SQL* užklausos apdorojamos apie 30 % ilgiau už *SPARQL*.



52 pav. SPARQL / SQL vykdymo trukmė

6.4. Eksperimento išvados

Eksperimento metu, naudojantis sudaryta metodiką buvo įgyvendintas modelis apjungiantis dvi reliacines duomenų bazes, kuris atsako į iškeltus kompetencijos klausimus. Sudaryta metodika reikalauja nuoseklaus modelio kūrimo proceso, siekiant, kad duomenų šaltinių integracija atitiktų visus keliamus reikalavimus.

Sudarytas duomenų modelis gali būti keičiamas, pridedant naujus duomenų šaltinius, tačiau tam reikia modifikuoti bazinę ontologiją ir išspręsti naujų, integruojamų duomenų šaltinių schematinius konfliktus.

7. Išvados

Šio darbo etape buvo apžvelgtos ontologijų panaudojimų galimybės duomenų bazių integravimo srityje, buvo analizuojamos ontologijų rūšys bei algoritmai, įrankiai ir karkasai sprendžiantys integruotą konfliktų aptikimo bei sprendimo uždavinius. Buvo iškirtos pagrindinės problemos, susijusios su pasirinkta dalykine sritimi. Apibendrinus darbą, buvo padarytos tokios išvados:

1. Schematiniai konfliktų gali būti išsprendžiami taikant *OWL* kalbą, nes kalba turi savybes tinkančias integruojamų duomenų šaltinių shematiniams konfliktams spręsti.
2. Automatizuotas reliacinių duomenų šaltinių integravimas yra komplikuotas uždavinys, nes integruojant schemas reikalinga nustatyti neatitikimus (sinonimai, antonimai, homonimai) tarp schemos elementų ir duomenų. Tokie neatitikimai įvardijami konfliktais ir yra sprendžiami dalykinės srities eksperto, dalykinės srities konceptualizavimo metu.
3. Taikant tik schematinio integravimo metodą, be duomenų suderinimo, galima suderinti skirtingas schemas, kurios gali būti semantiškai skirtingos, bet struktūriškai panašios, tačiau daugelyje taikomųjų uždavinių svarbu pasiekti abiejų lygių suderinamumą.
4. Duomenims suderinti reikia įvykdyti duomenų transformacijas, tam gali būti naudojami XML arba SWS servisai.
5. Taikant sudarytą metodika galima įgyvendinti reliacinių duomenų šaltinių integracija, pasiekti shematinį duomenų šaltinių suderinamumą. Sudaryta metodika ir procesas yra efektyvus sprendžiant heterogeninių duomenų šaltinių integracijos uždavinius.
6. Eksperimento metu gauti sudaryto modelio kokybės įverčiai yra tinkami taikyti modelį praktiniams taikymas.

8. Literatūra

[1] S. Ram, J. Park, „Semantic Conflict Resolution Ontology (SCROL): An Ontology for Detecting and Resolving Data and Schema-Level Semantic Conflict“, IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 2, pp. 189-202, 2004.

[2] F. Giunchiglia, A. Autayeu, J. Pane, „S-MATCH: „An Open Source Framework for Maching Lightweight Ontologines“, 2010, DISI - Via Sommarive 14 - 38123 Povo - Trento (Italy).

[3] M. Gagnon, „Ontology-Based Integration of Data Sources“, Information Fusion, 10th International Conference, ISBN: 978-0-662-45804-3, pp. 1-8, 2007.

[4] J. Wen, S. Zhang, Z. Yan, „SLCO and DLCO: Two Ontologies for Detecting and Resolving Schema and Data-Level Semantic Conflicts“, Information and Automation, 2009. ICIA '09. International Conference, pp. 637-642, 2009.

[5] C. Li, T. Wang Ling, „OWL-Based Semantic Conflicts Detection and Resolution for Data Interoperability“, Conceptual Modelling for Advanced Application Domains Lecture Notes in Computer Science, vol. 3289, pp. 266-277, 2004.

[6] Y. Zhao, S. Zhang, „Ontology-based Model for Resolving the Data-Level and Semantic-Level Conflicts“, Information and Automation, 2009. ICIA '09. International Conference, pp. 455-459, 2009; M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[7] H. Wache, T. Vogele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, S. Hubner, „Ontology-Based Integration of Information – A Survey of Existing Approaches“, Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing. Seattle, WA, pp. 108-117, 2001.

[8] J. Wang, J. Lu, Y. Zhang, et al., „Integrating Heterogeneous Data Source using Ontology “, Journal of Software, vol. 4, no. 8, pp. 843-850, 2009.

[9] P. Shvaiko, J. Euzenat, „A Survey of Schema-Based Matching Approaches“, Journal on Data Semantics, vol. IV, pp.146-171, 2005.

[10] X. Liu, S. Akram, A. Bouguettaya, Change Management for Semantic Web Services. Springer Science+Business Media. LLC, pp. 14-30, 2011.

[11] D. Gaševič, D. Djurič, V. Devedžič, Model Driven Engineering and Ontology Development. 2nd Edition, Springer, pp. 110-138; 133-141, 2009.

[12] D. Allemanng, J. Hendler, Semantic Web for Working Ontologist, Efective Modeling in RDFS and OWL. Second Edition, Elsevier Inc., pp. 13-22, 125-279, 2011.

[13] S. Busse, R. D. Kutsche, U. Leser, H. Weber, Federated Information Systems: Concepts, Terminology and Architectures. Technische Universität Berlin, Fachbereich 13-Informatik, 1999.

[14] W3C, OWL 2 Web Ontology Language Document Overview (Second Edition). Prieiga per internetą: http://www.w3.org/TR/owl2-overview/.

[15] W3C, RDF Vocabulary Description Language 1.0: RDF Schema. Prieiga per internetą: http://www.w3.org/TR/2003/PR-rdf-schema-20031215/.

[16] V. Uzdanavičiūtė, R. Butleris, „Oracle DBVS galimybės ontologijomis grindžiamo duomenų integravimo procese“. Proceedings of the 15th Master and PhD conference „Information Society and University studies“ (IVUS 2010), 13th May 2010, Kaunas, ISSN-4824, pp. 169-174, 2011.

[17] V. Uzdanavičiūtė, R. Butleris, „Ontology-Based Foundations for Data Integration“. BUSTECH' 2011: Proceedings of The First International Conference on Business Intelligence and Technology, 2011.

[18] D2R Server: Accessing Databases with SPARQL and as Linked Data. Prieiga per internetą: http://d2rq.org/d2r-server.

[19] The D2RQ Mapping Language. Prieiga per internetą: http://d2rq.org/d2rq-language.

[20] OWL 2 Web Ontology Language Mapping to RDF Graphs. Prieiga per internetą: http://www.w3.org/TR/owl2-mapping-to-rdf/.

[21] J. F. Sequeda, M. Arenas, D. P. Miranker, „On Directly Mapping Relational Databases to RDF and OWL“, Proceedings of the 21st International Conference on World Wide Web, April 16-20, Lyon, France, pp. 649-658, 2012.

[22] SPARQL Query Language for RDF. Prieiga per internetą: http://www.w3.org/TR/rdf-sparql-query/.

[23] C. E. Naiman, A. M. Ouksel, „A Classification of Semantic Conflicts in Heterogeneous Database Systems“, Journal of Organizational Computing, vol. 5, no. 2, pp. 167-193, 1995.

[24] M. Gruninger, „Methodology for the Design and Evaluation of Ontologies“, Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95, Montreal, 1995.

[25] *Virtuoso* sistemos aprašymas. Prieiga per internetą: http://virtuoso.openlinksw.com/.

[26] *S-Match* sistemos aprašymas. Prieiga per internetą: http://semanticmatching.org/s-match.html.

9. Priedai

9.1. Straipsnis

Aštuonioliktosios tarpuniversitetinės tarptautinės magistrantų ir doktorantų konferencijoje „Informacinė visuomenė ir universitetinės studijos“ (IVUS 2013) pristatytas straipsnis „Reliacinių duomenų bazių duomenų integravimas taikant ontologijas“.

Reliacinių duomenų bazių duomenų integravimas taikant ontologijas

Tomas Trainys
Informacijos sistemų katedra
Kauno technologijos universitetas
Kaunas, Lietuva
tomas.trainys@stud.ktu.lt

Virginija Uzdanavičiūtė
Informacijos sistemų katedra
Kauno technologijos universitetas
Kaunas, Lietuva
virginija.uzdanaviciute@stud.ktu.lt

prof. Rimantas Butleris
Informacijos sistemų katedra
Kauno technologijos universitetas
Kaunas, Lietuva
rimantas.butleris@ktu.lt

Santrauka. **Kuriant naujas ar plečiant esamas informacijos sistemas (IS) susiduriama su heterogeninių duomenų bazių jungimo, integravimo bei duomenų pakartotinio panaudojimo problemomis. Yra atlikta tyrimų, eksperimentų, sukurta programinių prototipų, tačiau integravimo uždavinys vis dar iki galo nepritaikytas / neišspręstas. Konceptualaus integruojamų schemų suderinamumo modeliui įgyvendinti siūloma taikyti ontologijas, nes jos skatina dalytis žiniomis. OWL modelis pasižymi išraiškingumu, yra standartizuotas (W3C rekomendacija), vienareikšmiškai suprantamas, formalus, jam interpretuoti yra sukurta daug programinės įrangos. Straipsnyje aprašoma metodika, kaip integruoti duomenų šaltinius taikant ontologijas. Išanalizuotos kalbos ontologijoms sudaryti, aptartos galimos integravimo architektūros, integravimo procesas bei semantiniai konfliktai ir jų sprendimo būdai.**

Reikšminiai žodžiai: ontologija, OWL, duomenų šaltinių integravimo metodai, duomenų modelis, semantiniai konfliktai, heterogeniniai duomenų šaltiniai.

I. ĮVADAS

Per pastaruosius kelis dešimtmečius įmonės duomenims saugoti sėkmingai pritaikė įvairias duomenų bazių valdymo sistemas (DBVS). Plačiai paplito įvairių gamintojų DBVS – *MS SQL Server, Oracle, MySQL* ir kt. Istoriškai susiklostė, kad pirmiausia duomenų bazės (DB) buvo diegiamos laikantis decentralizuoto požiūrio, o įmonėms plečiantis, didėjant padalinių skaičiui ir jungiantis į didesnes korporacijas bei sindikatus, atsirado poreikis įmonių duomenis integruoti ir paruošti toliau naudoti. Integravimo klausimas tapo svarbus daugiau nei prieš trisdešimt metų ir iki šiol yra aktualus.

Reikia paminėti, kad pastaruoju metu dideliu tempu besikeičiant technologinei aplinkai bei plečiantis ir bendradarbiaujant įmonėms duomenų integracijos klausimas išlieka aktualus ir turi įtakos galutinei kuriamo produkto kainai. Be to, atsižvelgiant į didėjančią telekomunikacinių tinklų spartą, kuri globaliuose ir vietiniuose kompiuterių tinkluose gali siekti iki kelių GB/s, ir įvertinant šiuo metu eksploatuojamos techninės įrangos našumą (procesorinę galią), taip pat atminčių spartą, galima teigti, kad techninės įrangos sluoksnis yra pakankamai išvystytas didėlės apimties heterogeniniams duomenų šaltiniams integruoti. Todėl yra racionalu geografiškai paskirstytus duomenų šaltinius integruoti, tam pritaikant centralizuotos duomenų bazės (dar vadinamos federaline duomenų baze) metodus. Toks sprendimas leistų sumažinti DBVS administravimo kainą ir pagreitinti taikomosios programinės įrangos kūrimo procesą.

Centralizuotą duomenų bazę galima apibūdinti kaip sistemą, suteikiančią naudotojams galimybę, naudojant *SQL / SPARQL* užklausas, manipuliuoti geografiškai paskirstytų duomenų šaltinių duomenimis. Yra galimybių manipuliuoti ir heterogeninių duomenų šaltinių duomenimis, tačiau dažniausiai susiduriama su struktūriniais ir semantiniais konfliktais duomenų šaltinių schemos bei duomenų lygmenyse.

Heterogeniškumą galima apibūdinti keliais aspektais [13]: *struktūrinis* – apima skirtingus duomenų modelius; *sintaksinis* – apima skirtingas kalbas ir duomenų vaizdavimą; *sisteminis* – apima operacines sistemas ir techninę įrangą; *semantinis* – apima semantinį duomenų modelių suderinamumą.

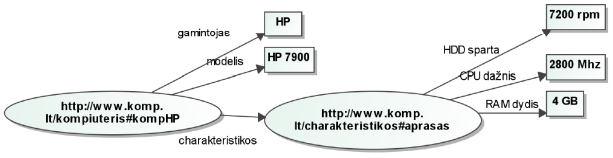
Darbas yra tiriamojo pobūdžio, apima tik semantinio suderinamumo klausimus heterogeninių duomenų šaltinių integravimo procese. Straipsnyje aptariami semantinio suderinamumo uždavinių sprendimo būdai, heterogeniniams duomenų šaltiniams taikant ontologijas, kaip aukštesnio lygmens duomenų šaltinių aprašymo modelį. Plačiai nagrinėjamas ontologijų pagrindu vykdomas integravimo procesas, aptariamos architektūros ir metodai, leidžiantys užtikrinti duomenų bazių homogeniškumą bei informacijos integralumą. Taip pat aprašomi įrankiai procesui įgyvendinti, pateikiama sprendimo metodika ir integravimo sistemos architektūros prototipas.

II. ONTOLOGIJOS IR JŲ APRAŠYMO KALBOS

„Ontologijos“ sąvoka informatikos srityje vartojama dalykinės srities konceptų specifikacijai apibūdinti [12]. Ontologija – aukšto semantinio lygmens dalykinės srities aprašas, kuris susideda iš dalykinės srities *konceptų*, *ryšių tarp konceptų, aksiomų ir apribojimų*. Tai yra duomenų modelis, apimantis klases, objektus, jų savybes ir ryšius [11]. Modelis suteikia galimybę detaliau nei *ER* modelis (angl. *entity relationship*) konceptualizuoti dalykinę sritį, sudaryti klasių hierarchiją bei sukurti ryšius tarp skirtingų dalykinių sričių. Vienas iš reikšmingų ontologijos modelio privalumų – išraiškingumas, žmogui ir kompiuterių sistemai suprantamas, teikiantis galimybę sudaryti dalykinių sričių žodynus. Formaliai ontologiją *O* galima aprašyti grafu [10] *G*,

turinčiu *Mo* baigtinę mazgų aibę ir baigtinę briaunų aibę *Bo*. Briaunos gali būti aprašomos trejetu m_1, m_2, p_1, kai m_1, m_2 yra *Mo* konceptai, o p_1 yra briaunos pavadinimas. Formaliai ontologijos koncepcinį duomenų modelį galima užrašyti $G_O = (M_O, G_O)$.

Ontologijoms sudaryti yra naudojami kelių kalbų poaibiai: *RDF, RDF shema* (angl. *resource description framework*). *RDF* modelis sudarytas iš trijų pagrindinių dalių: *subjekto* – bet kokio daikto, aprašyto *RDF*; *predikato – savybės*, charakteristikos, atributo ar ryšio, skirto ištekliui apibūdinti; *objekto – savybės reikšmės*, tai gali būti kitas išteklius arba duomenų tipo reikšmė, kurie kartu vadinami teiginiu. *RDF* grafo pavyzdys pateiktas 1 pav., jo tekstinis aprašas – 1 lentelėje.



1 pav. RDF grafas

1 lentelė. RDF GRAFO APRAŠAS

Subjektas	Predikatas	Objektas
http://www.komp.lt/kompiuteris#kompHP	Gamintojas	HP
http://www.komp.lt/kompiuteris#kompHP	Modelis	HP 7900
http://www.komp.lt/kompiuteris#kompHP	Charakteristikos	http://www.komp.lt/kompiuteris#aprasas
http://www.komp.lt/kompiuteris#aprasas	HDD sparta	7200 rpm
http://www.komp.lt/kompiuteris#aprasas	CPU dažnis	2800 Mhz

OWL (angl. *web ontology language*) – *RDF* ir *RDF-S* pagrindu sukurta ontologijų aprašymo kalba. *OWL* ontologija susideda iš konceptų ir ryšių tarp jų, apribojimo ir aksiomų. Toliau detaliai aptarsime *OWL* paketo struktūrą: pagrindinis *OWL* konstrukcinis elementas yra klasė, kalboje ji aprašoma *owl:Class* elementu. *OWL* klasė teikia galimybę grupuoti panašias charakteristikas turinčius išteklius. Savybe *rdfs:subClaccOf* yra aprašomi ryšiai tarp klasių, o savybė *rdfs:subPropertyOf* naudojama klasių hierarchijai sudaryti.

Ekvivalenčios klasės nustatomos naudojant konstrukcinį elementą *owl:equivalentClass*. Tuo pažymima, kad keli ištekliai (klasės, egzemplioriai, savybės) aprašo tą patį objektą. *OWL* klasė yra asocijuojama su klasės egzemplioriais, kurie yra vadinami klasės išplėtimais. Klasių egzemplioriai yra išvardijami naudojant konstrukcinį elementą *owl:oneOf*. Pagrindinės kalbos savybės yra: *owl:DatatypeProperty* ir *owl:ObjectProperty,* šiomis savybėmis yra sudaromi ryšiai tarp klasės egzempliorių bei priskiriami duomenų tipai.

OWL kalba yra aprašomos specifinės savybių charakteristikos, tokios kaip tranzityvumas, simetriškumas, inversija ir funkcinė priklausomybė. Šios charakteristikos aprašomos naudojant: *owl:SymmetricProperty, owl:InverseFunctionalProperty, owl:FunctionalProperty* ir *owl:TransitiveProperty* kalbos elementus.

Ryšių kardinalumas aprašomas naudojant elementus: *owl:cardinality, owl:minCardinality* ir *owl:maxCardinality.*

Sudėtinės klasės sudaromos naudojant savybes: *owl:intersectionOf, owl:unionOf* ir *owl:complementOf,* kuriomis vykdant sankirtos, sąjungos ir papildymo operacijas gaunamos sudėtinės klasės (dar vadinamos anoniminėmis klasėmis).

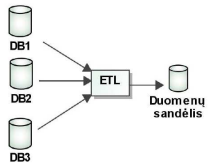
SPARQL (angl. *simple protocol and RDF query language*) – tai užklausų formulavimo kalba [22], skirta duomenų gavimui iš *RDF* grafo poaibių bei naujų RDF grafų formavimui. Pagrindiniai kalbos operatoriai: *SELECT* – grąžina rezultatus (pagal užklausoje pateiktus kintamuosius); *ASK* – identifikuoja informacijos egzistavimą; *DESCRIBE* – pateikia išteklius aprašantį *RDF* grafą; *CONSTRUCT* – naudojamas *RDF* grafui konstruoti.

III. DUOMENŲ ŠALTINIŲ INTEGRAVIMO ARCHITEKTŪROS IR METODAI

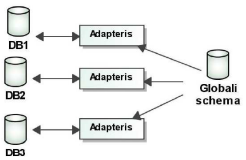
Duomenų, priklausančių skirtingiems duomenų šaltiniams, surinkimo, integravimo bei pateikimo uždaviniai gali būti sprendžiami taikant tradicinius metodus, tokius kaip duomenų sandėlių ir federalinių duomenų bazių [7], [13], [17] bei ontologijomis grindžiamus metodus. Toliau juos aptarsime detaliau.

A. Tradiciniai metodai

Duomenų sandėlio (angl. *warehouse*) architektūros (2 pav.) atveju duomenys iš duomenų šaltinių yra surenkami, transformuojami ir perkeliami (*ETL* – angl. *extract, transform, load*) į centralizuotą saugyklą [13]. Šios architektūros privalumas – duomenų gavybos operacijų (informacijos analizė, žinių išgavimas ir kt.) vykdymas nedarant įtakos paskirstytų duomenų šaltiniams. Naudotojai gali siųsti užklausas į didelius duomenų rinkinius. Paprastai tokios sistemos yra labai patikimos, užklausos greitai įvykdomos. Šios architektūros trūkumas – duomenų naujumas, kadangi naujausi duomenys būna paskirstytuose duomenų šaltiniuose ir turi būti sinchronizuojami su duomenų sandėliu.



2 pav. Duomenų sandėlio architektūra



3 pav. Globalios schemos architektūra

Globalios schemos architektūroje, skirtingai nei duomenų sandėlių sistemose, duomenys nėra perkeliami į pagrindinę duomenų saugyklą [13]. Autonominiai duomenų šaltiniai yra integruojami schemos lygmenyje. Tam yra sudaroma globali (3 pav.), integruojamų šaltinių schema (dar vadinama federaline schema arba virtualia duomenų baze). Integracija vykdoma susiejant duomenų šaltinių schemas su globalia schema. Sudarant globalią schemą svarbu suvienodinti šaltinių schemas, susieti schemų elementus ir išspręsti konfliktus tarp jų (pvz., skirtingi schemos elementų pavadinimai ir kt.).

Sistema, kurioje yra įdiegiama globali schema, vadinama globalia ar federaline sistema.

Federalinė sistema leidžia siųsti užklausas į visus integruotus duomenų šaltinius. Užklausos į duomenų šaltinius yra siunčiamos per adapterius (angl. *wrapper*), kurie atlieka pateiktų užklausų transformavimo ir rezultatų paketų sudarymo (angl. *encapsulation*) operacijas bei pateikia naudotojams rezultatus. Lyginant duomenų sandėlio ir federalines sistemas paminėtina, kad federalinės sistemos architektūros privalumas – duomenų naujumas. Pagrindinis architektūros trūkumas – ilgesnis užklausų siuntimo laikas.

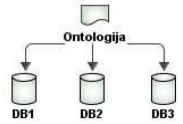
B. *Ontologijomis grindžiamas duomenų šaltinių integravimas*

Ontologijos teikia galimybę detaliai konceptualizuoti dalykinę sritį, todėl šis semantinių technologijų metodas gali būti taikomas informacijos šaltinių integravimo uždaviniams spręsti. Dalykinės srities konceptualizavimo procese sudaromi ryšiai tarp dalykinės srities konceptų ir nustatomi semantiškai nesuderinami objektai. Paminėtina, kad pats konceptualizavimo procesas padeda semantiškai suderinti skirtingas duomenų šaltinių schemas. Taikant ontologijų metodą galima išspręsti skirtingų schemų terminų daugiaprasmiškumus bei terminų sąryšių konfliktus [4], [17], pavyzdžiui, nustatyti ekvivalenčius schemos elementus tarp duomenų bazių ir kt. Tokios problemos apskritai apibūdinamos terminu „semantinis heterogeniškumas“. Ontologijų metodas leidžia naudotojams siųsti užklausas į skirtingus duomenų šaltinius taikant semantinį modelį. Semantinis modelis – globalios schemos atitikmuo (lyginant su tradiciniais metodais). Literatūros šaltiniuose [31], [7], [17] yra pateikiami skirtingi minėtos integracijos įgyvendinimo metodai. Trumpai aptarsime kiekvieną iš jų.

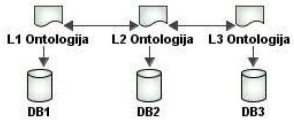
Globalios ontologijos metodas. Taikant šį metodą sudaroma viena ontologija (4 pav.), kurioje aprašomi visi duomenų šaltiniai ir tokiu būdu gaunamas integruotas dalykinės srities vaizdas, kuris dar gali būti apibūdinamas kaip semantinis modelis. Šis metodas leidžia siųsti užklausas per vieną ontologiją. Kai kuriais atvejais gali būti naudojamos kelių specializuotų ontologijų kombinacijos (dar vadinamos modulinėmis arba agreguotomis ontologijomis). Globalios ontologijos metodas gali būti taikomas integruojant konceptualiai panašius informacijos šaltinius [3]. Nors metodo architektūros sudėtis gali atrodyti paprasta, tačiau ontologijai sudaryti reikia dalykinės srities eksperto (dažnai kelių), kuris / kurie žinotų visų duomenų šaltinių semantiką. Pagrindinis metodo trūkumas – semantinio modelio priklausomybė nuo informacijos šaltinių pakeitimų. Pavyzdžiui, vieno informacijos šaltinio schemos pakeitimas gali paveikti visą ontologiją bei kitų informacijos šaltinių ryšius.

Taikant *sudėtinių ontologijų metodą* yra sudaroma kiekvieno duomenų šaltinio atskira ontologija (lokali ontologija) bei pavaizduojami ontologijų ryšiai (angl. *mapping*), (5 pav.). Vaizduojant šiuos ryšius svarbu įvertinti integruojamų šaltinių ontologijų terminų semantinį heterogeniškumą. Šioje architektūroje [31] [7] užklausos yra siunčiamos per lokalias ontologijas, o vaizdavimas yra naudojamas lokalių užklausoms siųsti į kitus duomenų šaltinius per jų lokalias ontologijas. Šio metodo privalumas – nepriklausomumas nuo informacijos šaltinių pakeitimų.

Pavyzdžiui, pakeitus vieno informacijos šaltinio ontologiją, ji nepaveiktų kitų, pavaizduotų informacijos šaltinių ontologijų. Literatūroje minima [7], kad gana sudėtinga taikyti šį metodą dėl tarpusavio ontologijų semantinio heterogeniškumo aspektų.

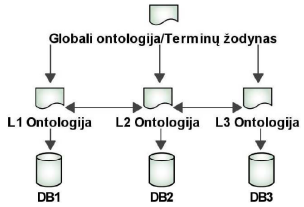


4 pav. Globalios ontologijos architektūra



5 pav. Sudėtinių ontologijų architektūra

Mišrus ontologijų metodas buvo pasiūlytas [7] prieš tai aptartų metodų trūkumams pašalinti. Šio metodo atveju duomenų šaltiniai, kaip ir sudėtinio metodo atveju, yra aprašomi lokalia ontologija (6 pav.), o visų duomenų šaltinių konceptai aprašomi bendra ontologija (dar vadinama žodynu [7]). Žodynas apima bazinius dalykinės srities terminus. Lokalios ontologijos sudaromos naudojant žodyne aprašytų terminų kompleksinius junginius. Ontologijos konstruojamos vartojant bazinius žodyno terminus, kurie leidžia semantiškai suderinti lokalias ontologijas. Naudotojai užklausas į šaltinius pateikia per bendrą žodyną. Šio metodo privalumai, palyginanti su prieš tai aptartais – paprastesnis papildomų informacijos šaltinių integravimas į esamą struktūrą bei geresnės sąlygos ontologijoms vystyti / plėsti.



6 pav. Mišrių ontologijų architektūra

Apibendrinant aptartus metodus reikėtų pažymėti, kad ontologijos ir duomenų šaltinių schemos yra panašios [9], žiūrint į abu modelius kaip į terminų žodyną, kurie abiejų modelių atveju aprašo dalykinę sritį. Be to, abu modeliai yra suvaržyti dalykinės srities žodyno terminų prasmių / reikšmių.

IV. SEMANTINIAI KONFLIKTAI, INTEGRAVIMO PROCESO PROBLEMOS IR METODAI JOMS SPRĘSTI

Skirtingų duomenų šaltinių integravimo klausimai plačiai analizuojami jau daugiau nei trisdešimt metų. Esami metodai ir algoritmai yra taikomi sistemų diegimo ir palaikymo etapuose. Daugeliu atvejų literatūroje minimų [3], [7], [8], [17], [23] integravimo problemų ir jų sprendimo būdų spręsti duomenų šaltinių heterogeniškumo klausimą yra panašūs. Integravimo procese susiduriama su semantiniais konfliktais, kurie charakterizuojami kaip skirtingos schemų ir duomenų lygmens loginės struktūros. Šie pagal tipą yra skirstomi į: sinonimus, homonimus, duomenų tikslumo bei kt. Minėtiems konfliktams nustatyti galima naudoti jau aptartos *OWL* kalbos savybes, kurios pateiktos 1 lentelėje. Toliau aptarsime pagrindinius konfliktų tipus bei jų nustatymo galimybes naudojant *OWL* savybes [5], [4], [6].

2 lentelė. OWL / RDF SAVYBĖS KONFLIKTAMS NUSTATYTI

Savybė	Panaudojimo aprašymas
owl:equivalentClass	Nustato ekvivalenčias klases.
owl:euqivalentProperty	Nustato ekvivalenčias ypatybes.
owl:sameIndividualAs	Nustato ekvivalenčius egzempliorius.
owl:disjointWith	Nustato semantiškai ekvivalenčius egzempliorius.
owl:differentFrom	Nustato, kad vienas egzempliorius skiriasi nuo kito.
owl:inverseOf	Pažymi, kad viena ypatybė yra priešinga kitai ypatybei.
owl:unionOf	Pažymi, kad nauja klasė yra gauta iš kitų klasių (sąjungos būdu).
owl:intersectionOf	Pažymi, kad nauja klasė yra gauta iš kitų klasių (sankirtos būdu).
owl:complementOf	Pažymi, kad nauja klasė yra egzistuojančių klasių papildinys.

A. *Semantinių konfliktų nustatymas naudojant OWL*

1) Pavadinimų konfliktai. Tokio tipo konfliktai įvyksta, kai du ir daugiau semantiškai ekvivalenčių /neekvivalenčių arba vienodą identifikatorių turinčių konceptų reprezentuoja tą patį objektą. Šie konfliktai įvardijami kaip sinonimai, antonimai ir homonimai:

a) Sinonimų konfliktai išsprendžiami suvienodinant ekvivalenčių konceptų pavadinimus. Ekvivalentūs konceptai yra nustatomi *OWL* kalbos savybėmis: *owl:equivalentClass, owl:equivalentProperty* ir *owl:sameInvidualAs*. Panagrinėkime pavyzdį (7 pav.), kuriame konceptai *Apskritis* ir *Provincija* yra ekvivalentūs; naudojant *owl:equivalentClass* elementą yra nustatomi tokio tipo konfliktai.

b) Antonimų konfliktai išsprendžiami išskiriant semantiškai neekvivalenčius konceptus. Tokio tipo konfliktai nustatomi *OWL* kalbos elementais *owl:disjointWith ir owl:differentFrom*. Pavyzdžiui (8 pav.), konceptai *BendrasĮvertinimas* ir *Pažymys* nėra ekvivalentūs. OWL elementu *owl:disjointwiths* yra išskiriami du neekvivalentūs konceptai, ir tokiu būdu išsprendžiamas antonimų konfliktas.

```
<owl:Class
rdf:ID="Provincija">
<rdfs:label>Provincija
</rdfs:label>
<owl:equivalentClass
rdf:resource="#Apskritis"/>
</owl:Class>
```

7 pav. Sinonimų konfliktų nustatymo aprašas

```
<owl:Class
rdf:ID="BendrasĮvertinimas">
<rdfs:label>Provincija</rdfs:label>
< owl:disjointwith
rdf:resource="#Pažymys"/>
</owl:Class>
```

8 pav. Antonimų konfliktų nustatymo aprašas

c) Homonimų konfliktai įvyksta, kai konceptai turi skirtingus identifikatorius, tačiau yra žymimi tokiais pačiais ar panašiais pavadinimais. Tokio tipo konfliktai dar yra vadinami identifikavimo konfliktais [5]. Juos reikia spręsti suteikiant atributams skirtingus pavadinimus [5]. Aptarkime pavyzdį (9 pav.): sakykime, turime dvi duomenų šaltinių schemas *A* ir *B*, kurių abiejų atributas yra *tvarkaraščio_Data*, tačiau ontologijose jie yra pavaizduoti skirtingai: *A* ontologijoje *studijuTvarkaraščioData*, o *B* ontologijoje *darboTvarkaraščioData*, tačiau abiejose ontologijose abu atributai žymimi žyme *tvarkaraščioData* (naudojamas elementas *rdfs:label*). Šiame pavyzdyje norint išvengti homonimų konfliktų, *A* ir *B* duomenų šaltinių schemose atributams reikia suteikti skirtingus pavadinimus.

```
<rdf:Property rdf:ID=" studijuTvarkaraščioData ">
<rdfs:label> tvarkaraščio_Data </rdfs:label>
</rdf:Property>
<rdf:Property rdf:ID=" darboTvarkaraščioData ">
<rdfs:label> tvarkaraščio_Data </rdfs:label>
</rdf:Property>
```

9 pav. Homonimų konfliktų nustatymo aprašas

2) Kompleksinių klasių konfliktai. Tokio tipo konfliktai įvyksta tarp klasių, kurios buvo sudarytos naudojant sankirtos / sąjungos operacijas, kai vieno koncepto elementų priklausomybė nuo kito koncepto elementų yra visiška arba dalinė. Aptarsime du tokių konfliktų atvejus:

a) Visiškos priklausomybės konfliktų gali įvykti, kai vienas konceptas yra visiškai priklausomas (įeina į kitą konceptą) nuo kito koncepto [15], [5]. Tokio tipo konfliktai yra išsprendžiami klasės atributus išskaidant į dalinius atributus. Šie konfliktai gali būti nustatomi konstrukciniu elementu *owl:unionOf.* Panagrinėkime pavyzdį (10 pav.), kuriame turime du konceptus *Vardas* ir *Pavardė*, ir *owl:unionof* elementas pažymi, kad minėti konceptai yra gauti sąjungos būdu. Kitaip sakant, jei klasė *A* gauta sąjungos būdu iš klasių *B* ir *C*, o *A* klasė nėra ekvivalenti *B* ir *C* klasėms, tada sujungiant *A* ir *B* arba *A* ir *C* klases gali įvykti visiškos priklausomybės konfliktas.

```
<rdf:Property rdf:ID="Pavardė">
< owl:unionOf
rdf:resource="#Vardas">
</rdf:Property>
```

10 pav. Visiškos priklausomybės konfliktų nustatymo aprašas

```
<owl:Class
rdf:ID="Asistentas">
<owl: intersectioOf
rdf:resource="#Absolventas"/>
</owl:Class>
```

11 pav. Dalinės priklausomybės konfliktų nustatymo aprašas

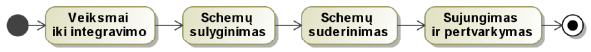
b) Dalinės priklausomybės konfliktų įvyksta tarp dviejų iš dalies sutampančių konceptų. Tokio tipo konfliktai nustatomi per *OWL* ypatybę owl:*intersectioOf*. Šie konfliktai gali būti išsprendžiami išskiriant iš dalies sutampančias konceptų dalis [5], [4]. Pavyzdyje, 11 pav., matyti dviejų schemų integracija, kur vienoje yra *Asistentas*, o kitoje – *Absolventas*. Čia abu konceptai (asistentas ir absolventas) iš dalies sutampa, nes kai kurie asistentai gali būti ir absolventai, bet ne visi asistentai yra absolventai ir ne visi absolventai yra asistentai. Šiuo atveju norint išvengti konfliktų, konceptai turi būti išskaidomi, sudarant atskiras klases: studijų nebaigusius asistentus, studijas baigusius asistentus ir studijuojančius asistentus.

B. *Integravimo procesas*

Heterogeninių duomenų šaltinių integravimo ontologijų pagrindu metodai yra įgyvendinami pagal procesą, pateiktą 12 pav.

Išskiriami šie proceso etapai: *pasiruošimas arba veiksmai iki integravimo.* Šiame etape yra atliekama integruojamų schemų analizė, sudaromas integravimo algoritmas ir nustatomas integravimo veiksmų eiliškumas; *schemų sulyginimo* etape suderinami schemų konceptai ir nustatomi

jų konfliktai; *schemų suderinimo* etape sprendžiami schemų konfliktai; *jungimo ir pertvarkymo* etape schemos sujungiamos ir pertvarkomos pagal iš anksto apibrėžtus kriterijus.



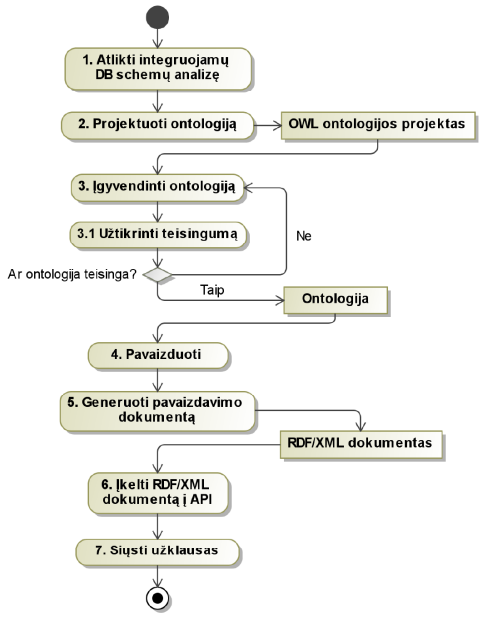
12 pav. Integravimo proceso diagrama

C. *Duomenų šaltinių integravimo ontologijų pagrindu metodika*

Šioje dalyje aptarsime siūlomą schematinio integravimo procesą, taikant globalios ontologijos metodą.

Heterogeninių duomenų šaltinių integravimo uždavinių procesas pavaizduotas 13 pav. Proceso diagramoje kiekvienas uždavinys pažymėtas skaitmenimis, kuriuos detaliai aptarsime.

Integravimo procesas pradedamas nuo integruojamų duomenų šaltinių schemų analizės, kurios metu nagrinėjami schemų konceptai, jų egzemplioriai ir ryšiai tarp jų.



13 pav. Duomenų šaltinių integravimo proceso diagrama

Antrame etape, remiantis integruojamų duomenų šaltinių analizės duomenimis, sudaromas ontologijos projektas. Tam naudojami programiniai įrankiai, tokie kaip *MagicDraw*, *Altova Semantic Works, NeOnToolKit* ir kiti bei ontologijų sudarymo metodologijos, pvz., *TOVE, UPON* ir kt. Metodologijos aprašo, kokio proceso reikėtų laikytis sudarant, jungiant ar pakartotinai naudojant ontologijas, taip pat aptariami ontologijų evoliucionavimo aspektai ir jų pritaikymas taikomajai programinei įrangai.

Trečiame etape įgyvendinama ontologija. Tam taip pat rekomenduojama naudotis ontologijų sudarymo metodologijomis bei programiniais įrankiais, pvz.: *Protege, NeOnToolKit*, *Altova Semantic Works* ir kt. Ontologijų konstravimas yra iteracinis procesas, kuris pradedamas nuo dalykinės srities nustatymo, po to išvardijami terminai, sukuriamos klasės ir jų egzemplioriai, sudaromos ypatybės (ryšiai) tarp klasių ir jų egzempliorių, klasių egzemplioriams priskiriamos reikšmės ir galiausiai sudaromos aksiomos. Ontologijos teisingumui nustatyti, yra naudojami loginių išvadų varikliai, tokie kaip *Palet*, *F++, HermiT* ir kt., kurie iš sudarytų faktų leidžia atlikti logines išvadas ir automatizuotai sudaro klasių hierarchiją. Kai kurie įrankiai (*Protege, NeOnToolKit*) turi integruotus išvadų variklius.

Ketvirtame etape pavaizduojamos ontologijos ir duomenų šaltinių schemos (angl. *mapping*). Tai daroma šiais būdais: pirmas – naudojant įrankį *NeONToolKit su ODE Mapster* plėtiniu. Įrankis leidžia grafiniu režimu sukurti duomenų šaltinių ir ontologijos vaizdavimo dokumentą; antras – naudojant *2DR* programinį paketą [18], kuriuo galima sugeneruoti vaizdavimo dokumentą. *2DR* programinis paketas naudoja *D2RQ* vaizdavimo kalbą, kurios aprašas pateiktas [19] šaltinyje; trečias – naudojant *W3C* [20], [21] tiesioginio vaizdavimo standartą, kuris leidžia vaizduoti duomenų šaltinius ir globalią ontologiją. Sudarant atvaizdą, gali būti naudojamas vienas iš antrame etape paminėtų ontologijų konstravimo įrankių.

Penktame etape iš sudaryto vaizdavimo modelio sugeneruojamas *RDF*, XML ar kito specifinio formato dokumentas. Paminėtina, kad sugeneruotas vaizdavimo dokumentas (jo standartas) turi būti suderintas su integravimo platforma, kurioje jis bus naudojamas.

Šeštame etape sugeneruotas modelis įdiegiamas taikomajai programinei įrangai, turinčiai *OWL / RDF* bibliotekas, arba gali būti naudojamas integravimui skirtuose technologiniuose sprendimuose, pvz., *2DR*, *Virtuoso* ir kt.

Septintame etape suderinama sistema ir siunčiamos semantinės užklausos į ontologijoje aprašytus duomenų šaltinius.

V. INTEGRAVIMO SISTEMOS PROTOTIPO KOMPONENTAI

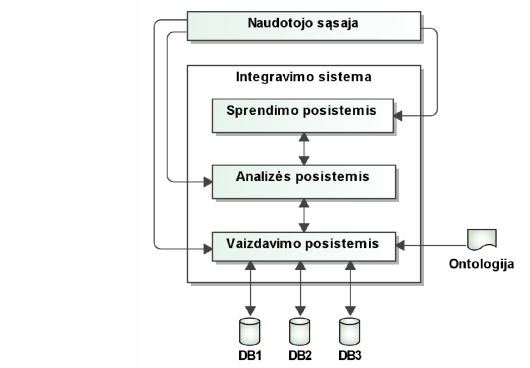
Aprašytų ontologijomis grindžiamų duomenų šaltinių integravimo metodams bei jų procesui įgyvendinti literatūros šaltiniuose [2], [16], [18], [19] pateikiamos įvairios galimos integravimo sistemų komponentų kompozicijos bei galimi, semantinio duomenų šaltinių integravimo sprendimai. Atsižvelgiant į technologinius pasiekimus ir prieinamas eksperimento priemones, šiame darbe aprašoma architektūra, kurią galima apibūdinti kaip susietų komponentų sistemą, kurios pagrindinės dalys nurodytos 14 pav. Toliau detaliai aptarsime kiekvieną iš jų.

Vaizdavimo posistemis vykdo atvaizdavimą tarp duomenų šaltinių ir globalios ontologijos bei leidžia vykdyti semantines užklausas. Vaizdavimui gali būti naudojami tokie įrankiai: *NeonToolKit, 2RD, Protege* (naudojant *DataMaster* išplėtimą), *2DRQ* ir *W3C* kalbos [20] ir *Jena* ar *Sesam* adapteriai.

Analizės posistemis vykdo semantinių konfliktų nustatymo operacijas, o aptiktus konfliktus perduoda *sprendimo posistemiui*, kuris atlieka objektų transformavimo operacijas. Literatūroje [11] aptariami įvairūs galimi šio komponento įgyvendinimo ir diegimo atvejai. Tai gali būti *SOA* (angl. *service oriented architecture*). Vienas iš *SOA* atvejų *XML* technologijos pagrindu veikiantis servisas, kuriuo yra vykdomos objektų transformavimo operacijos. Kitas *RDF / OWL* pagrindu veikiantis servisas *SWS* (angl. *semantic web service*) [11]. Konfliktai paprastai nėra sprendžiami

visiškai automatizuotai, dažniausiai – pusiau automatizuotai, leidžiant naudotojui priimti konkrečiam atvejui tinkamą sprendimą.

Naudotojo sąsaja skirta sistemai konfigūruoti, ontologijai įkelti, konfliktams valdyti ir užklausoms į duomenų šaltinius siųsti.



14 pav. Integravimo sistemos architektūra

Komponentai gali būti sujungiami naudojant programavimo technologijas (pvz., *JAVA, .Net ir kt.*), kurios turi bibliotekas dirbti su *RDF / OWL,* pvz., *Java.API, DotNet.RDF* ir kt.

VI. IŠVADOS

Šiame straipsnyje išanalizuotos ontologijų panaudojimo galimybės duomenų bazių integravimo uždaviniams įgyvendinti. Aprašyti semantinių konfliktų tipai bei jų nustatymo galimybės taikant *OWL* kalbą. Automatizuotas reliacinių duomenų šaltinių integravimas yra komplikuotas uždavinys, nes integruojant schemas reikalinga nustatyti neatitikimus (sinonimai, antonimai, homonimai) tarp schemos elementų ir duomenų. Taikant vien schematinio integravimo metodą, be duomenų suderinimo, galima suderinti skirtingas schemas, kurios gali būti semantiškai skirtingos, bet struktūriškai panašios; tačiau daugelyje taikomųjų uždavinių svarbu pasiekti abiejų lygių suderinamumą. Todėl prieš integruojant duomenis tikslinga identifikuoti ir išspręsti semantinius konfliktus. Tam tikslui pasiūlyta semantinius konfliktus ir jų tipus aprašyti *OWL* kalba, duomenims suderinti – įvykdyti duomenų transformacijas *XML* ar *SWS* servisais, panaudojant *SOA* architektūrą.

Tolesniuose tyrimuose numatoma išanalizuoti duomenų lygmens semantinius konfliktus, jų sprendimų metodus, panaudoti prototipą duomenų šaltiniams sujungti, taikant globalios schemos architektūrą, bei sudaryti algoritmą automatizuotai nustatyti schemos elementus ir iš dalies automatizuotai juos spręsti.

[1] S. Ram, J. Park, „Semantic Conflict Resolution Ontology (SCROL): An Ontology for Detecting and Resolving Data and Schema-Level Semantic Conflict“, IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 2, pp. 189-202, 2004.

[2] F. Giunchiglia, A. Autayeu, J. Pane, „S-MATCH: „An Open Source Framework for Maching Lightweight Ontologines“, 2010, DISI - Via Sommarive 14 - 38123 Povo - Trento (Italy).

[3] M. Gagnon, „Ontology-Based Integration of Data Sources“, Information Fusion, 10th International Conference, ISBN: 978-0-662-45804-3, pp. 1-8, 2007.

[4] J. Wen, S. Zhang, Z. Yan, „SLCO and DLCO: Two Ontologies for Detecting and Resolving Schema and Data-Level Semantic Conflicts“, Information and Automation, 2009. ICIA '09. International Conference, pp. 637-642, 2009.

[5] C. Li, T. Wang Ling, „OWL-Based Semantic Conflicts Detection and Resolution for Data Interoperability“, Conceptual Modelling for Advanced Application Domains Lecture Notes in Computer Science, vol. 3289, pp. 266-277, 2004.

[6] Y. Zhao, S. Zhang, „Ontology-based Model for Resolving the Data-Level and Semantic-Level Conflicts“, Information and Automation, 2009. ICIA '09. International Conference, pp. 455-459, 2009; M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[7] H. Wache, T. Vogele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, S. Hubner, „Ontology-Based Integration of Information – A Survey of Existing Approaches“, Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing. Seattle, WA, pp. 108-117, 2001.

[8] J. Wang, J. Lu, Y. Zhang, et al., „Integrating Heterogeneous Data Source using Ontology “, Journal of Software, vol. 4, no. 8, pp. 843-850, 2009.

[9] P. Shvaiko, J. Euzenat, „A Survey of Schema-Based Matching Approaches“, Journal on Data Semantics, vol. IV, pp.146-171, 2005.

[10] X. Liu, S. Akram, A. Bouguettaya, Change Management for Semantic Web Services. Springer Science+Business Media. LLC, pp. 14-30, 2011.

[11] D. Gaševič, D. Djurič, V. Devedžič, Model Driven Engineering and Ontology Development. 2nd Edition, Springer, pp. 110-138; 133-141, 2009.

[12] D. Allemanng, J. Hendler, Semantic Web for Working Ontologist, Efective Modeling in RDFS and OWL. Second Edition, Elsevier Inc., pp. 13-22, 125-279, 2011.

[13] S. Busse, R. D. Kutsche, U. Leser, H. Weber, Federated Information Systems: Concepts, Terminology and Architectures. Technische Universität Berlin, Fachbereich 13-Informatik, 1999.

[14] W3C, OWL 2 Web Ontology Language Document Overview (Second Edition). Prieiga per internetą: http://www.w3.org/TR/owl2-overview/.

[15] W3C, RDF Vocabulary Description Language 1.0: RDF Schema. Prieiga per internetą: http://www.w3.org/TR/2003/PR-rdf-schema-20031215/.

[16] V. Uzdanavičiūtė, R. Butleris, „Oracle DBVS galimybės ontologijomis grindžiamo duomenų integravimo procese“. Proceedings of the 15th Master and PhD conference „Information Society and University studies“ (IVUS 2010), 13th May 2010, Kaunas, ISSN-4824, pp. 169-174, 2011.

[17] V. Uzdanavičiūtė, R. Butleris, „Ontology-Based Foundations for Data Integration“. BUSTECH' 2011: Proceedings of The First International Conference on Business Intelligence and Technology, 2011.

[18] D2R Server: Accessing Databases with SPARQL and as Linked Data. Prieiga per internetą: http://d2rq.org/d2r-server.

[19] The D2RQ Mapping Language. Prieiga per internetą: http://d2rq.org/d2rq-language.

[20] OWL 2 Web Ontology Language Mapping to RDF Graphs. Prieiga per internetą: http://www.w3.org/TR/owl2-mapping-to-rdf/.

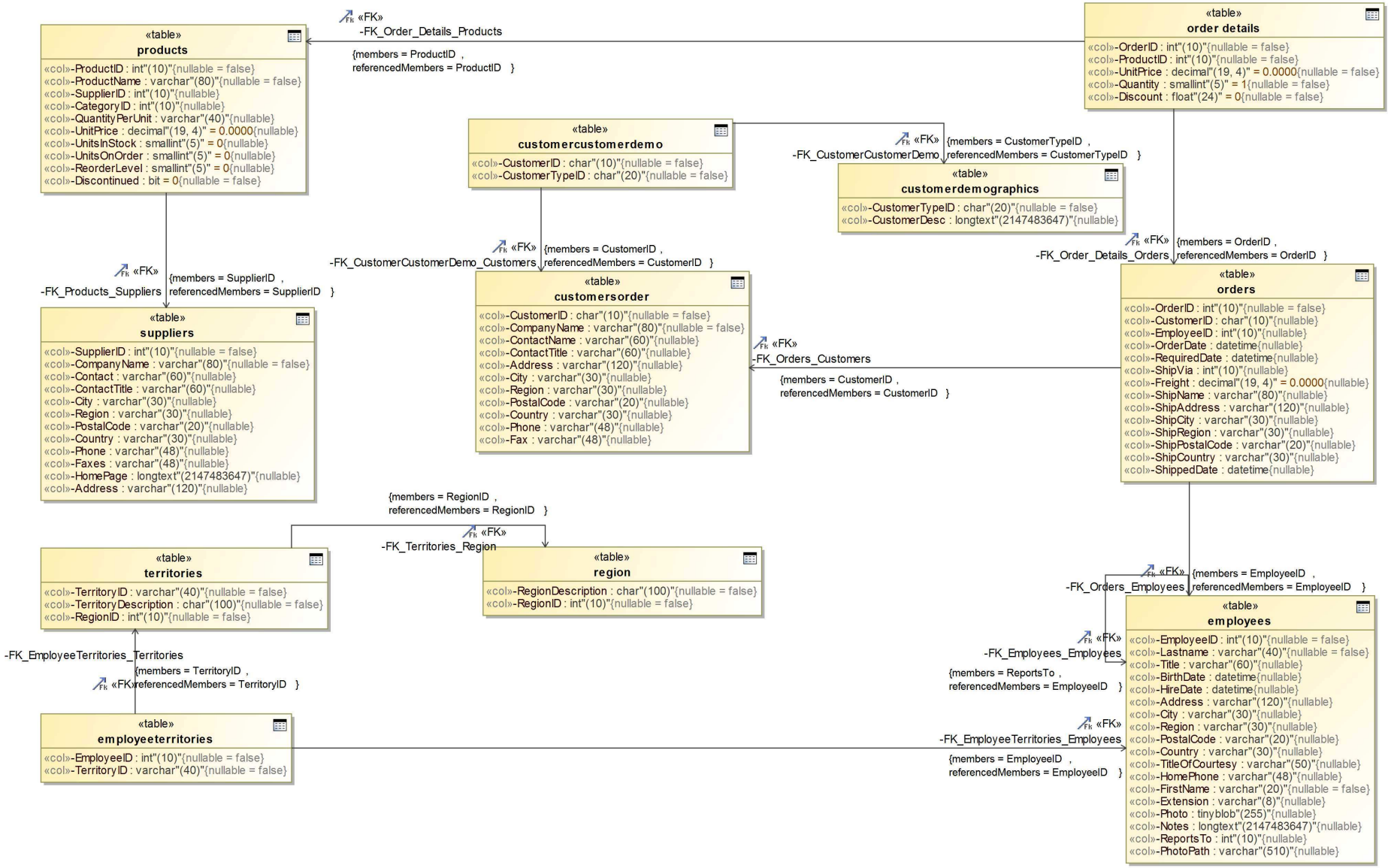
[21] J. F. Sequeda, M. Arenas, D. P. Miranker, „On Directly Mapping Relational Databases to RDF and OWL“, Proceedings of the 21st International Conference on World Wide Web, April 16-20, Lyon, France, pp. 649-658, 2012.

[22] SPARQL Query Language for RDF. Prieiga per internetą: http://www.w3.org/TR/rdf-sparql-query/.

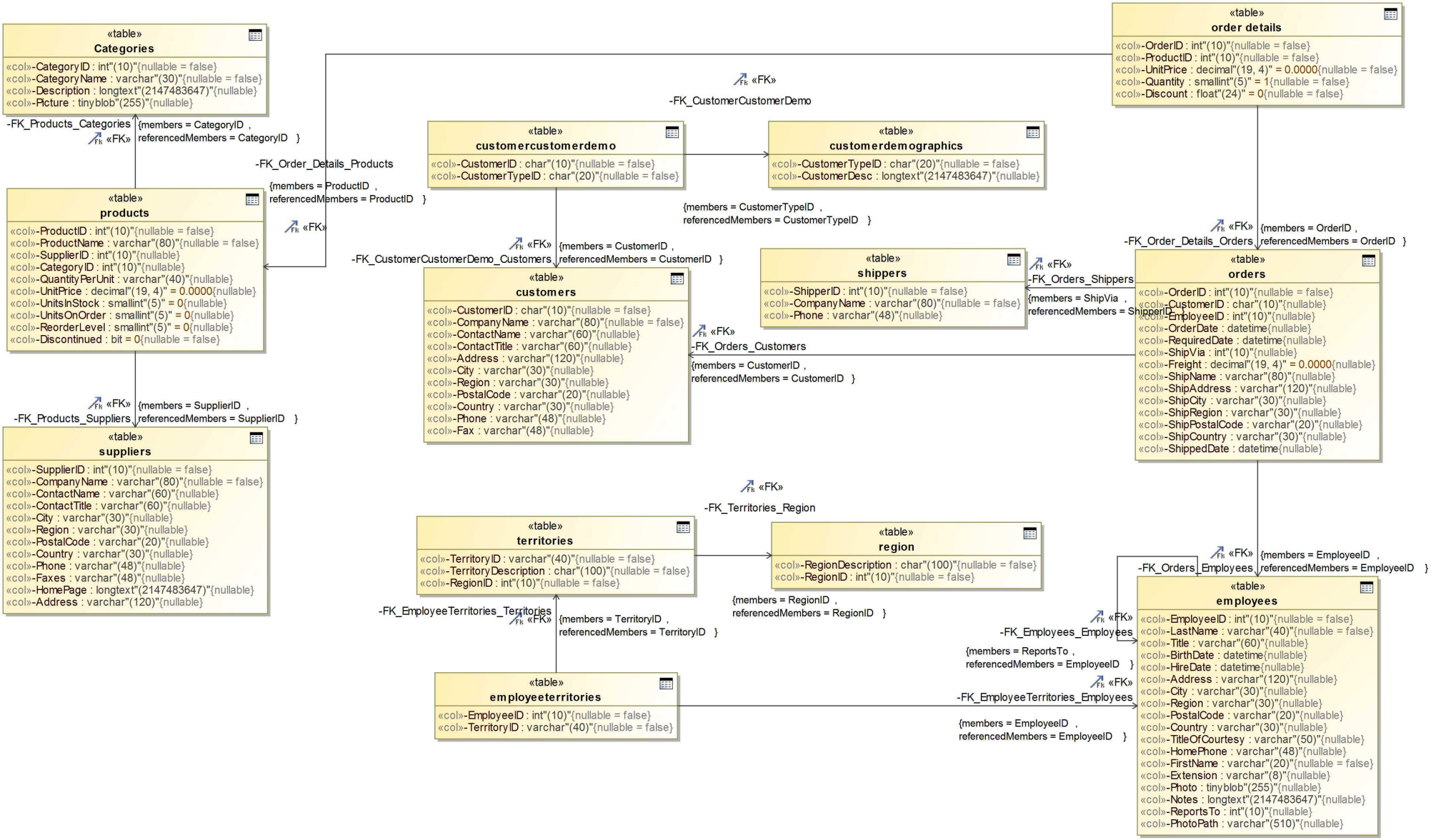
[23] C. E. Naiman, A. M. Ouksel, „A Classification of Semantic Conflicts in Heterogeneous Database Systems“, Journal of Organizational Computing, vol. 5, no. 2, pp. 167-193, 1995.

9.2. Integruojamų duomenų bazių schemos

9.2.1. DB2 schema.



9.2.2. DB1 schema



9.3. Ontologijos aprašas

Bazinės ontologijos aprašas

1

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
	<!ENTITY foaf "http://xmlns.com/foaf/0.1/">
	<!ENTITY owl "http://www.w3.org/2002/07/owl#">
	<!ENTITY xsd
"http://www.w3.org/2001/XMLSchema#">
	<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-
schema#">
	<!ENTITY wgs84_pos
"http://www.w3.org/2003/01/geo/wgs84_pos#">
	<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-
syntax-ns#">
	<!ENTITY ontology
"http://localhost/tutorial/Northwind/ontology/">
]>
<rdf:RDF
xml:base="http://localhost/tutorial/Northwind/ontology/"
xmlns="http://localhost.com/tutorial/Northwind/ontology/"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:ontology="http://localhost/tutorial/Northwind/ontology/"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:virtrdf="http://www.localhost.com/schemas/virtrdf#"
xmlns:wgs84_pos="http://www.w3.org/2003/01/geo/wgs84_pos#
" xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
	<owl:Ontology
rdf:about="http://.localhost/tutorial/Northwind/ontology/">
		<rdfs:label>Northwind</rdfs:label>

	<virtrdf:catName>Northwind</virtrdf:catName>
		<rdfs:comment>Northwind database classes
and properties</rdfs:comment>
		<virtrdf:version>1.00</virtrdf:version>
	</owl:Ontology>
	<!--
  ///////////////////////////////////////////////////////////////////////////////
  //
  // Annotation properties
  //
  ///////////////////////////////////////////////////////////////////////////////
   -->
	<owl:AnnotationProperty rdf:about="UzsakymoData">
		<rdfs:label>UzsakymoData</rdfs:label>
		<rdfs:cardinality>1</rdfs:cardinality>
		<rdfs:domain rdf:resource="Uzsakymas"/>
	</owl:AnnotationProperty>
	<owl:AnnotationProperty
rdf:about="&foaf;GimimoDiena"/>
	<owl:AnnotationProperty rdf:about="turi_salies_koda">
		<rdfs:label>SaliesKodas</rdfs:label>
		<rdfs:cardinality>1</rdfs:cardinality>
		<rdfs:domain
rdf:resource="&ontology;Apskritis"/>
	</owl:AnnotationProperty>
	<owl:AnnotationProperty rdf:about="Nuolaida">
		<rdfs:label>Nuolaida</rdfs:label>
		<rdfs:cardinality>1</rdfs:cardinality>
		<rdfs:domain rdf:resource="Pristatymai"/>
	</owl:AnnotationProperty>
	<owl:AnnotationProperty rdf:about="turi_darbuotoja">
		<rdfs:label>Darbuotojas</rdfs:label>
		<rdfs:cardinality>1</rdfs:cardinality>
		<rdfs:domain rdf:resource="Uzsakymas"/>
	</owl:AnnotationProperty>
	<owl:AnnotationProperty
rdf:about="PristatymoRajonas">
```

2

```
  ///////////////////////////////////////////////////////////////////////////////
   -->
	<!-- http://localhost/tutorial/Northwind/ontology/Address
-->
	<owl:DatatypeProperty rdf:about="Adresas">
		<rdfs:label>Adresas</rdfs:label>
		<rdfs:cardinality>1</rdfs:cardinality>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<!--
http://localhost/tutorial/Northwind/ontology/BirthDate -->
	<owl:DatatypeProperty rdf:about="GImimoData">
		<rdfs:label>GImimoData</rdfs:label>
		<rdfs:cardinality>1</rdfs:cardinality>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<!--
http://localhost/tutorial/Northwind/ontology/CategoryName -->
	<owl:DatatypeProperty
rdf:about="KategorijuPavadinimas">
		<rdfs:label>KategorijuPavadinimas</rdfs:label>
		<rdfs:cardinality>1</rdfs:cardinality>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<!-- http://localhost/tutorial/Northwind/ontology/City -->
	<owl:DatatypeProperty rdf:about="Miestas">
		<rdfs:label>Miestas</rdfs:label>
		<rdfs:cardinality>1</rdfs:cardinality>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<!-- http://localhost/tutorial/Northwind/ontology/Code -->
	<owl:DatatypeProperty rdf:about="Kodas">
		<rdfs:label>Kodas</rdfs:label>
		<rdfs:cardinality>1</rdfs:cardinality>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<!--
http://localhost/tutorial/Northwind/ontology/CompanyName -->
	<owl:DatatypeProperty rdf:about="FirmosPavadinimas">
		<rdfs:label>FirmosPavadinimas</rdfs:label>
		<rdfs:cardinality>1</rdfs:cardinality>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<!--
http://localhost/tutorial/Northwind/ontology/ContactName -->
	<owl:DatatypeProperty rdf:about="KontaktinisAsmuo">
		<rdfs:label>KontaktinisAsmuo</rdfs:label>
		<rdfs:cardinality>1</rdfs:cardinality>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<!--
http://localhost/tutorial/Northwind/ontology/ContactTitle -->
	<owl:DatatypeProperty
rdf:about="KontaktoPavadinimas">
		<rdfs:label>KontaktoPavadinimas</rdfs:label>
		<rdfs:cardinality>1</rdfs:cardinality>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<!--
http://localhost/tutorial/Northwind/ontology/Description -->
	<owl:DatatypeProperty rdf:about="Aprasymas">
		<rdfs:label>Aprasymas</rdfs:label>
		<rdfs:cardinality>1</rdfs:cardinality>
		<rdfs:range rdf:resource="&xsd;string"/>
	</owl:DatatypeProperty>
	<!--
```

```
                <rdfs:label>PristatymoApskritis</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Uzsakymas"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="&ontology;FirmosPavadinimas">
                <rdfs:label>FirmosPavadinimas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Darbuotojai"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty rdf:about="Neivykdyti">
                <rdfs:label>Neivykdyti</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Produktai"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty rdf:about="Vardas">
                <rdfs:label>Vardas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Darbuotojai"/>
                <rdfs:subPropertyOf
rdf:resource="&foaf;Vardas"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="&ontology;Sandelyje">
                <rdfs:label>VienetaiSandelyjek</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Produktai"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="http://xmlns.com/foaf/0.1/Pavarde"/>
        <owl:AnnotationProperty rdf:about="Rajonas">
                <rdfs:label>Rajonas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Klientai"/>
                <rdfs:domain rdf:resource="Darbuotojai"/>
                <rdfs:domain rdf:resource="Tiekejas"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty rdf:about="GImimoData">
                <rdfs:label>GImimoData</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Darbuotojai"/>
                <rdfs:subPropertyOf
rdf:resource="&foaf;GimimoData"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty rdf:about="Aprasymas">
                <rdfs:label>Aprasymas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Kategorijos"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty rdf:about="PastoKodas">
                <rdfs:label>PastoKodas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Klientai"/>
                <rdfs:domain rdf:resource="Darbuotojai"/>
                <rdfs:domain rdf:resource="Tiekejas"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="PristatymoMiestas">
                <rdfs:label>PristatymoMiestas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Uzsakymas"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty rdf:about="turi_kategorija">
                <rdfs:label>Kategorija</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Produktai"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="turi_uzsakymo_numeri">
                <rdfs:label>Uzsakymas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Pristatymai"/>
http://localhost/tutorial/Northwind/ontology/Discontinued -->
        <owl:DatatypeProperty rdf:about="Neivykdyti">
                <rdfs:label>Neivykdyti</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;integer"/>
        </owl:DatatypeProperty>
        <!-- http://localhost/tutorial/Northwind/ontology/Discount
-->
        <owl:DatatypeProperty rdf:about="Nuolaida">
                <rdfs:label>Nuolaida</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;double"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/Extension -->
        <owl:DatatypeProperty rdf:about="Ispletimas">
                <rdfs:label>Ispletimas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!-- http://localhost/tutorial/Northwind/ontology/Fax -->
        <owl:DatatypeProperty rdf:about="Faksas">
                <rdfs:label>Faksas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/FirstName -->
        <owl:DatatypeProperty rdf:about="Vardas">
                <rdfs:label>Vardas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!-- http://localhost/tutorial/Northwind/ontology/Freight --
>
        <owl:DatatypeProperty rdf:about="Pristatytojas">
                <rdfs:label>Pristatytojas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;double"/>
        </owl:DatatypeProperty>
        <!-- http://localhost/tutorial/Northwind/ontology/HireDate
-->
        <owl:DatatypeProperty rdf:about="Nuoma">
                <rdfs:label>Nuoma</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/HomePage -->
        <owl:DatatypeProperty rdf:about="WebPuslapis">
                <rdfs:label>WebPuslapis</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/LargeFlagDAVResourc
eName -->
        <owl:DatatypeProperty
rdf:about="&ontology;ResursoPavadinimas">
                <rdfs:label>ResursoPavadinimas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/LargeFlagDAVResourc
eURI -->
        <owl:DatatypeProperty
rdf:about="&ontology;ResursoPavadinimas">
                <rdfs:label>ResursoPavadinimas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
```

```
        </owl:AnnotationProperty>
        <owl:AnnotationProperty rdf:about="&foaf;Vardas"/>
        <owl:AnnotationProperty
rdf:about="http://xmlns.com/foaf/0.1/Pareigos"/>
        <owl:AnnotationProperty rdf:about="Miestas">
                <rdfs:label>Miestas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Klientai"/>
                <rdfs:domain rdf:resource="Darbuotojai"/>
                <rdfs:domain rdf:resource="Tiekejas"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty rdf:about="Ispletimas">
                <rdfs:label>Ispletimas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Darbuotojai"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="&ontology;ResursoPavadinimas">
                <rdfs:label>ResursoPavadinimas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Salis"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="KontaktoPavadinimas">

        <rdfs:label>KontaktoPavadinimas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Klientai"/>
                <rdfs:domain rdf:resource="Tiekejas"/>
                <rdfs:subPropertyOf
rdf:resource="http://xmlns.com/foaf/0.1/Pareigos"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty rdf:about="Kodas">
                <rdfs:label>Kodas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Salis"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="&ontology;VienetoKaina">
                <rdfs:label>VienetoKaina</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Pristatymai"/>
                <rdfs:domain rdf:resource="Produktai"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="&rdfs;cardinality"/>
        <owl:AnnotationProperty rdf:about="PristatymoSalis">
                <rdfs:label>PristatymoSalis</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Uzsakymas"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="http://xmlns.com/foaf/0.1/Vardas"/>
        <owl:AnnotationProperty rdf:about="Pranesimai">
                <rdfs:label>Pranesimai</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Darbuotojai"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="&ontology;ResursoPavadinimas">
                <rdfs:label>ResursoPavadinimas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Salis"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty rdf:about="Faksas">
                <rdfs:label>Faksas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Klientai"/>
                <rdfs:domain rdf:resource="Tiekejas"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty rdf:about="Gavejas">
                <rdfs:label>Gavejas</rdfs:label>
```

```
<!-- http://localhost/tutorial/Northwind/ontology/LastName -->
        <owl:DatatypeProperty rdf:about="Pavarde">
                <rdfs:label>Pavarde</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!-- http://localhost/tutorial/Northwind/ontology/Name --
>
        <owl:DatatypeProperty rdf:about="&ontology;Vardas">
                <rdfs:label>Vardas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!-- http://localhost/tutorial/Northwind/ontology/Notes -->
        <owl:DatatypeProperty rdf:about="Pastabos">
                <rdfs:label>Pastabos</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/OrderDate -->
        <owl:DatatypeProperty rdf:about="UzsakymoData">
                <rdfs:label>UzsakymoData</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!-- http://localhost/tutorial/Northwind/ontology/Phone --
>
        <owl:DatatypeProperty rdf:about="Telefonas">
                <rdfs:label>Telefonas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/PostalCode -->
        <owl:DatatypeProperty rdf:about="PastoKodas">
                <rdfs:label>PastoKodas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/ProductName -->
        <owl:DatatypeProperty
rdf:about="ProduktoPavadinimas">
                <rdfs:label>ProduktoPavadinimas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/ProvinceName -->
        <owl:DatatypeProperty rdf:about="ApskritiesPav">
                <rdfs:label>ApskritiesPav</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!-- http://localhost/tutorial/Northwind/ontology/Quantity
-->
        <owl:DatatypeProperty rdf:about="Kiekis">
                <rdfs:label>Kiekis</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;integer"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/QuantityPerUnit -->
        <owl:DatatypeProperty rdf:about="KiekisPakuoteje">
                <rdfs:label>KiekisPakuoteje</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>

<!-- http://localhost/tutorial/Northwind/ontology/Region -->
        <owl:DatatypeProperty rdf:about="Rajonas">
```

```
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Uzsakymas"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty rdf:about="Pastabos">
                <rdfs:label>Notes</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Darbuotojai"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty rdf:about="&ontology;Salis">
                <rdfs:label>Salis</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Salis"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="&ontology;ResursoPavadinimas">
                <rdfs:label>ResursoPavadinimas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Salis"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="&ontology;ResursoPavadinimas">
                <rdfs:label>ResursoPavadinimas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Salis"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="&ontology;UzsakymoKiekisr">
                <rdfs:label>UzsakymoKiekis</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Produktai"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="PristatymoAdresas">
                <rdfs:label>PristatymoAdresas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Uzsakymas"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty rdf:about="turi_pirkeja">
                <rdfs:label>Pirkejas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Uzsakymas"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="http://xmlns.com/foaf/0.1/Telefonas"/>
        <owl:AnnotationProperty
rdf:about="ProduktoPavadinimas">

        <rdfs:label>ProduktoPavadinimas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Produktai"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="KontaktinisAsmuo">
                <rdfs:label>KontaktinisAsmuo</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Klientai"/>
                <rdfs:domain
rdf:resource="KlientuKontaktai"/>
                <rdfs:domain rdf:resource="Tiekejas"/>
                <rdfs:subPropertyOf
rdf:resource="http://xmlns.com/foaf/0.1/Vardas"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="KategorijuPavadinimas">

        <rdfs:label>KategorijuPavadinimas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:domain rdf:resource="Kategorijos"/>
        </owl:AnnotationProperty>
        <owl:AnnotationProperty
rdf:about="turi_produkto_koda">
                <rdfs:label>Produktas</rdfs:label>
                <rdfs:label>Rajonas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/ReorderLevel -->
        <owl:DatatypeProperty
rdf:about="PapildomasUsakymas">
                <rdfs:label>PapildomasUsakymas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;integer"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/ReportsTo -->
        <owl:DatatypeProperty rdf:about="Pranesimai">
                <rdfs:label>Pranesimai</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;integer"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/RequiredDate -->
        <owl:DatatypeProperty rdf:about="PristatymoData">
                <rdfs:label>PristatymoData</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/ShipAddress -->
        <owl:DatatypeProperty rdf:about="PristatymoAdresas">
                <rdfs:label>PristatymoAdresas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!-- http://localhost/tutorial/Northwind/ontology/ShipCity
-->
        <owl:DatatypeProperty rdf:about="PristatymoMiestas">
                <rdfs:label>PristatymoMiestas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/ShipCountry -->
        <owl:DatatypeProperty rdf:about="PristatymoSalis">
                <rdfs:label>PristatymoSalis</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/ShipName -->
        <owl:DatatypeProperty rdf:about="Gavejas">
                <rdfs:label>Gavejas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/ShipPostalCode -->
        <owl:DatatypeProperty
rdf:about="PristatymoPastoKodas">
                <rdfs:label>PristatymoPastoKodas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/ShipRegion -->
        <owl:DatatypeProperty rdf:about="PristatymoRajonas">
  <rdfs:label>PristatymoRajonas</rdfs:label>
                <rdfs:cardinality>1</rdfs:cardinality>
                <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/ShippedDate -->
        <owl:DatatypeProperty
```

```
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:domain rdf:resource="Pristatymai"/>
                        <rdfs:domain rdf:resource="Vykdytojas"/>
                        <rdfs:domain rdf:resource="Tiekejas"/>
                        <rdfs:subPropertyOf
rdf:resource="http://xmlns.com/foaf/0.1/Telefonas"/>
                </owl:AnnotationProperty>
                <owl:AnnotationProperty rdf:about="ApskritiesPav">
                        <rdfs:label>ApskritiesPav</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:domain rdf:resource="Apskritis"/>
                </owl:AnnotationProperty>
                <owl:AnnotationProperty rdf:about="&ontology;Lng">
                        <rdfs:label>Lng</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:domain rdf:resource="Salis"/>
                </owl:AnnotationProperty>
                <owl:AnnotationProperty
rdf:about="SaliesPavadinimas">
                        <rdfs:label>SaliesPavadinimas</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:domain rdf:resource="Klientai"/>
                        <rdfs:domain
rdf:resource="KlientuKontaktai"/>
                        <rdfs:domain rdf:resource="Darbuotojai"/>
                        <rdfs:domain rdf:resource="Tiekejas"/>
                </owl:AnnotationProperty>
                <owl:AnnotationProperty
rdf:about="&ontology;Pavadinimas">
                        <rdfs:label>Pavadinimas</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:domain rdf:resource="Darbuotojai"/>
                        <rdfs:subPropertyOf
rdf:resource="http://xmlns.com/foaf/0.1/Pareigos"/>
                </owl:AnnotationProperty>
                <owl:AnnotationProperty rdf:about="WebPuslapis">
                        <rdfs:label>WebPuslapis</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:domain rdf:resource="Tiekejas"/>
                </owl:AnnotationProperty>
                <owl:AnnotationProperty
rdf:about="FirmosPavadinimas">
                        <rdfs:label>FirmosPavadinimas</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:domain rdf:resource="Klientai"/>
                        <rdfs:domain rdf:resource="Vykdytojas"/>
                        <rdfs:domain rdf:resource="Tiekejas"/>
                        <rdfs:subPropertyOf
rdf:resource="http://xmlns.com/foaf/0.1/Vardas"/>
                </owl:AnnotationProperty>
                <owl:AnnotationProperty rdf:about="Adresas">
                        <rdfs:label>Adresas</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:domain rdf:resource="Klientai"/>
                        <rdfs:domain rdf:resource="Darbuotojai"/>
                        <rdfs:domain rdf:resource="Tiekejas"/>
                </owl:AnnotationProperty>
                <owl:AnnotationProperty rdf:about="Nuoma">
                        <rdfs:label>Nuoma</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:domain rdf:resource="Darbuotojai"/>
                </owl:AnnotationProperty>
                <!--
   ///////////////////////////////////////////////////////////////////////////////
   //
   // Datatypes
   //
   ///////////////////////////////////////////////////////////////////////////////
    -->
                <!--
   ///////////////////////////////////////////////////////////////////////////////
   //
```

```
rdf:about="&ontology;PristatymoData">
                        <rdfs:label>PristatymoData</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:range rdf:resource="&xsd;string"/>
                </owl:DatatypeProperty>
                <!--
http://localhost/tutorial/Northwind/ontology/SmallFlagDAVResourc
eName -->
                <owl:DatatypeProperty
rdf:about="&ontology;ResursoPavadinimas">
                        <rdfs:label>ResursoPavadinimas</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:range rdf:resource="&xsd;string"/>
                </owl:DatatypeProperty>
                <!--
http://localhost/tutorial/Northwind/ontology/SmallFlagDAVResourc
eURI -->
                <owl:DatatypeProperty
rdf:about="&ontology;ResursoPavadinimas">
                        <rdfs:label>ResursoPavadinimas</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:range rdf:resource="&xsd;string"/>
                </owl:DatatypeProperty>
                <!-- http://localhost/tutorial/Northwind/ontology/Title -->
                <owl:DatatypeProperty rdf:about="&ontology;Antraste">
                        <rdfs:label>Antraste</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:range rdf:resource="&xsd;string"/>
                </owl:DatatypeProperty>
                <!--
http://localhost/tutorial/Northwind/ontology/TitleOfCourtesy -->
                <owl:DatatypeProperty
rdf:about="&ontology;VisitoAntraste">
                        <rdfs:label>VisitoAntraste</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:range rdf:resource="&xsd;string"/>
                </owl:DatatypeProperty>
                <!-- http://localhost/tutorial/Northwind/ontology/UnitPrice
-->
                <owl:DatatypeProperty
rdf:about="&ontology;VienetoKaina">
                        <rdfs:label>VienetoKaina</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:range rdf:resource="&xsd;double"/>
                </owl:DatatypeProperty>
                <!--
http://localhost/tutorial/Northwind/ontology/UnitsInStock -->
                <owl:DatatypeProperty
rdf:about="&ontology;VienetaiSandelyje">
                        <rdfs:label>VienetaiSandelyje</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:range rdf:resource="&xsd;integer"/>
                </owl:DatatypeProperty>
                <!--
http://localhost/tutorial/Northwind/ontology/UnitsOnOrder -->
                <owl:DatatypeProperty
rdf:about="&ontology;UzsakytiVienetaiSandelyje">

                        <rdfs:label>UzsakytiVienetaiSandelyje</rdfs:label>
                        <rdfs:cardinality>1</rdfs:cardinality>
                        <rdfs:range rdf:resource="&xsd;integer"/>
                </owl:DatatypeProperty>
                <!--
 ///////////////////////////////////////////////////////////////////////////////
   //
   // Classes
   //
   ///////////////////////////////////////////////////////////////////////////////
    -->
                <!-- http://localhost/tutorial/Northwind/ontology/Category
-->
                <owl:Class rdf:about="Kategorijos">
```

```
    // Object Properties
    //
    //////////////////////////////////////////////////////////////////////////////////
     -->
        <!--
http://localhost/tutorial/Northwind/ontology/CountryName -->
        <owl:ObjectProperty rdf:about="SaliesPavadinimas">
            <rdfs:label>Salis</rdfs:label>
            <rdfs:cardinality>1</rdfs:cardinality>
            <rdfs:range rdf:resource="Salis"/>
        </owl:ObjectProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/has_category -->
        <owl:ObjectProperty rdf:about="turi_kategorija">
            <rdfs:label>Kategorija</rdfs:label>
            <rdfs:cardinality>1</rdfs:cardinality>
            <rdfs:range rdf:resource="Kategorijos"/>
        </owl:ObjectProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/has_country_code -->
        <owl:ObjectProperty rdf:about="turi_salies_koda">
            <rdfs:label>Salis Code</rdfs:label>
            <rdfs:cardinality>1</rdfs:cardinality>
            <rdfs:range rdf:resource="Salis"/>
        </owl:ObjectProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/has_customer -->
        <owl:ObjectProperty rdf:about="turi_pirkeja">
            <rdfs:label>Klientas</rdfs:label>
            <rdfs:cardinality>1</rdfs:cardinality>
            <rdfs:range rdf:resource="Klientai"/>
        </owl:ObjectProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/has_employee -->
        <owl:ObjectProperty rdf:about="turi_darbuotoja">
            <rdfs:label>Darbuotojas</rdfs:label>
            <rdfs:cardinality>1</rdfs:cardinality>
            <rdfs:range rdf:resource="Darbuotojai"/>
        </owl:ObjectProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/has_order_id -->
        <owl:ObjectProperty
rdf:about="turi_uzsakymo_numeri">
            <rdfs:label>Uzsakymas</rdfs:label>
            <rdfs:cardinality>1</rdfs:cardinality>
            <rdfs:range rdf:resource="Uzsakymas"/>
        </owl:ObjectProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/has_product_id -->
        <owl:ObjectProperty rdf:about="turi_produkto_koda">
            <rdfs:label>Produktas</rdfs:label>
            <rdfs:cardinality>1</rdfs:cardinality>
            <rdfs:range rdf:resource="Produktai"/>
        </owl:ObjectProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/has_supplier -->
        <owl:ObjectProperty
rdf:about="&ontology;turi_tiekejar">
            <rdfs:label>Tiekejas</rdfs:label>
            <rdfs:cardinality>1</rdfs:cardinality>
            <rdfs:range rdf:resource="Tiekejas"/>
        </owl:ObjectProperty>
        <!--
http://localhost/tutorial/Northwind/ontology/order_ship_via -->
        <owl:ObjectProperty rdf:about="uzsakyteas_per">
            <rdfs:label>Pristatytojas</rdfs:label>
            <rdfs:cardinality>1</rdfs:cardinality>
            <rdfs:range rdf:resource="Vykdytojas"/>
        </owl:ObjectProperty>
        <!--
    //////////////////////////////////////////////////////////////////////////////////
    //
```

```
            <rdfs:label>Kategorijos</rdfs:label>
            <rdfs:subClassOf rdf:resource="Produktai"/>
        </owl:Class>
        <!-- http://localhost/tutorial/Northwind/ontology/Country
-->
        <owl:Class rdf:about="Salis">
            <rdfs:label>Rajonas</rdfs:label>
            <rdfs:subClassOf
rdf:resource="&wgs84_pos;Produktas"/>
        </owl:Class>
        <!--
http://localhost/tutorial/Northwind/ontology/Customer -->
        <owl:Class rdf:about="Klientai">
            <rdfs:label>Klientas</rdfs:label>
            <rdfs:subClassOf
rdf:resource="http://xmlns/foaf/01/Organizacija"/>
        </owl:Class>
        <!--
http://localhost/tutorial/Northwind/ontology/CustomerContact -->
        <owl:Class rdf:about="KlientuKontaktai">
            <rdfs:label>KlientuKontaktai</rdfs:label>
            <rdfs:subClassOf
rdf:resource="http://xmlns/foaf/01/Asmuo"/>
        </owl:Class>
        <!--
http://localhost/tutorial/Northwind/ontology/Employee -->
        <owl:Class rdf:about="Darbuotojai">
            <rdfs:label>Darbuotojas</rdfs:label>
            <rdfs:subClassOf
rdf:resource="http://xmlns/foaf/01/Asmuo"/>
        </owl:Class>
        <!-- http://localhost/tutorial/Northwind/ontology/Order -->
        <owl:Class rdf:about="Uzsakymas">
            <rdfs:label>Uzsakymas</rdfs:label>
        </owl:Class>
        <!--
http://localhost/tutorial/Northwind/ontology/OrderLine -->
        <owl:Class rdf:about="Pristatymai">
            <rdfs:label>UzsakymoKiekis</rdfs:label>
            <rdfs:subClassOf rdf:resource="Uzsakymas"/>
        </owl:Class>
        <!-- http://localhost/tutorial/Northwind/ontology/Product -
->
        <owl:Class rdf:about="Produktai">
            <rdfs:label>Produktai</rdfs:label>
        </owl:Class>
        <!-- http://localhost/tutorial/Northwind/ontology/Province
-->
        <owl:Class rdf:about="Apskritis">
            <rdfs:label>Apskritis</rdfs:label>
            <rdfs:subClassOf
rdf:resource="&wgs84_pos;SpatialThing"/>
        </owl:Class>
        <!-- http://localhost/tutorial/Northwind/ontology/Shipper -
->
        <owl:Class rdf:about="Vykdytojas">
            <rdfs:label>PristatymoVykdytojas</rdfs:label>
        </owl:Class>
        <!-- http://localhost/tutorial/Northwind/ontology/Supplier
-->

<owl:Class rdf:about="Tiekejas">
            <rdfs:label>Tiekejas</rdfs:label>
        </owl:Class>
        <!--
http://wwww3org/2003/01/geo/wgs84_pos#SpatialThing -->
        <owl:Class rdf:about="&wgs84_pos;SpatialThing"/>
        <!-- http://xmlns/foaf/01/Organization -->
        <owl:Class
rdf:about="http://xmlns/foaf/01/Organizacija"/>
        <!-- http://xmlns/foaf/01/Person -->
        <owl:Class rdf:about="http://xmlns/foaf/01/Asmuo">
```

// Data properties //	<rdfs:subClassOf> <owl:Class> <owl:unionOf rdf:parseType="Collection"> <rdf:Description rdf:about="KlientuKontaktai"/> <rdf:Description rdf:about="Darbuotojai"/> </owl:unionOf> </owl:Class> </rdfs:subClassOf> </owl:Class> </rdf:RDF> <!-- Generated by the OWL API (version 3231824) http://owlapisourceforgenet -