Univerza v Mariboru

Fakulteta za elektrotehniko računalništvo in informatiko

Doktorska disertacija

Integracija podatkovnih virov na osnovi ontologij s semantično označenimi storitvami

September 2011

Boštjan Grašič

Univerza v Mariboru

Fakulteta za elektrotehniko računalništvo in informatiko

Doktorska disertacija

Integracija podatkovnih virov na osnovi ontologij s semantično označenimi storitvami

Boštjan Grašič

Mentor: izr. prof. dr. Vili Podgorelec

September 2011

UDK: 004.62:004.775(043.2)

Doktorska disertacija

Boštjan Grašič – grasic.b@gmail.com
Mentor: izr. prof. dr. Vili Podgorelec

September 2011

Študijski program: Računalništvo in informatika, podiplomski
Smer: Informatika

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Smetanova ulica 17
2000 Maribor

Inštitut za informatiko
Laboratorij za informacijske sisteme

Univerza v Mariboru

Maribor, 29. 3. 2011
Številka: DR 17/2011/425-MGM

Na osnovi 287., 140., 142. in 144. člena Statuta Univerze v Mariboru (Statut UM-UPB8, Ur. l. RS, št. 1/2010) ter sklepa 40. redne seje Senata Univerze v Mariboru z dne 29. 3. 2011 v zvezi z vlogo doktorskega kandidata Boštjana Grašiča za sprejem odločitve o predlagani temi doktorske disertacije in mentorja,

izdajam naslednji

SKLEP

Odobri se tema doktorske disertacije Boštjana Grašiča s Fakultete za elektrotehniko, računalništvo in informatiko z naslovom »Integracija podatkovnih virov na osnovi ontologij s semantično označenimi storitvami«. Za mentorja se imenuje izr. prof. dr. Vili Podgorelec. Kandidat mora članici predložiti izdelano doktorsko disertacijo v zadostnih izvodih najpozneje do 28. 3. 2015.

Obrazložitev:

Kandidat Boštjan Grašič je dne 19. 11. 2010 na Fakulteti za elektrotehniko, računalništvo in informatiko vložil vlogo za potrditev teme doktorske disertacije z naslovom »Integracija podatkovnih virov na osnovi ontologij s semantično označenimi storitvami«. Za mentorja je bil predlagan izr. prof. dr. Vili Podgorelec.

Senat Fakultete za elektrotehniko, računalništvo in informatiko je na osnovi pozitivnega mnenja komisije za oceno teme doktorske disertacije, ki je ugotovila, da kandidat izpolnjuje pogoje za pridobitev doktorata znanosti, in ocenila, da je predlagana tema ustrezna, sprejel pozitivno mnenje in poslal predlog teme doktorske disertacije s predlogom mentorja v odobritev Senatu univerze.

Senat Univerze v Mariboru je po proučitvi vloge in na osnovi določil Statuta Univerze v Mariboru sprejel svojo odločitev o predlagani temi doktorske disertacije in imenoval mentorja, kot izhaja iz izreka.

V skladu s 144. členom Statuta Univerze v Mariboru mora kandidat za pridobitev doktorata znanosti najpozneje v štirih letih od dneva izdaje tega sklepa, članici predložiti izdelano doktorsko disertacijo v zadostnih izvodih. Kandidatu je bil določen rok glede na datum sprejetja teme na pristojnem organu.

Pouk o pravnem sredstvu:
Zoper ta sklep je možna pritožba na Senat Univerze v Mariboru v roku 8 dni od prejema tega sklepa.

Rektor:
Prof. dr. Ivan Rozman

UNIVERZA V MARIBORU

Obvestiti:
1. Kandidata.
2. Fakulteto.
3. Arhiv.

HR EXCELLENCE IN RESEARCH

Zahvala

Doktorat posvečam svojim staršem, ki so mi omogočili brezskrben študij in me ves čas nesebično podpirali. Hvala vama za vajin trud in razumevanje. Posebno zahvalo izrekam Saši, ki mi je bila potrpežljivo v podporo po marsikateri dolgi noči za računalnikom. Zahvaljujem se tudi Urški, ki je požrtvovalno pripomogla k boljši berljivosti moje doktorske disertacije.

Iskreno se zahvaljujem tudi svojemu mentorju, izr. prof. dr. Viliju Podgorelcu, za usmerjanje in nasvete v času podiplomskega študija. Najino sodelovanje mi je bilo v veselje.

Zahvaljujem se sodelavcem Laboratorija za informacijske sisteme za njihovo pomoč in prijetnejše delo na Inštitutu v času podiplomskega študija.

Povzetek

Količina informacij, ki jih hranimo v različnih informacijskih sistemih, raste eksponentno. Ključni izziv, ki se poraja v obdobju vse večje informacijske zasičenosti, je kako na čim bolj učinkovit način obvladovati ter združevati informacije, ki so porazdeljene med heterogenimi podatkovnimi viri.

V okviru doktorske disertacije smo razvili pristop, ki omogoča za uporabnika transparentno integracijo heterogenih podatkovnih virov na osnovi ontologij. Inovativnost pristopa predstavlja uporaba storitev v vlogi podatkovnih virov. Storitve so označene s semantičnimi opisi, ki omogočajo samodejno odkrivanje, kompozicijo ter proženje storitev. Takšen pristop omogoča integracijo praktično kateregakoli podatkovnega vira z uporabo odprtih in standardiziranih tehnologij.

Ključni prispevki našega raziskovalnega dela so formalni model za integracijo s semantično označenimi storitvami ter algoritmi za distribuirano povpraševanje po podatkih, ki jih zagotavljajo storitve. Na osnovi definiranega modela in algoritmov smo razvili prototip ter izvedli eksperiment, kjer smo razviti prototip primerjali s standardnimi pristopi za delo s semantično označenimi podatki.

Na osnovi rezultatov eksperimenta smo izvedli statistično analizo. Rezultati statistične analize so pokazali, da je razvit prototip učinkovitejši od primerjanih pristopov. Razviti prototip se po učinkovitosti signifikantno ne razlikuje le od dveh najbolj učinkovitih pristopov. Na podlagi izvedene raziskave, lahko sklenemo, da uporaba semantično označenih storitev za integracijo podatkovnih virov na osnovi ontologij izboljša fleksibilnost in nima negativnega vpliva na učinkovitost.

Ključne besede: *Integracija podatkov, semantična integracija, ontologija, storitveno orientirana arhitektura, semantične storitve*

UDK: *004.62:004.775(043.2)*

Ontology based information integration using semantically annotated services

Abstract

We are living in a period of increasing saturation of information. The amount of information stored in different information systems is still growing exponentially. In this manner, the key challenge is to find the most efficient way to manage and integrate the information that is distributed among heterogeneous data sources.

In this dissertation we have developed a mediator based information integration approach, allowing users transparent integration of heterogeneous data sources using ontologies. The novelty of our approach is the use of services as data providers. Services are annotated with semantic descriptions that enable automated discovery, composition and service invocation. This approach allows integration of practically any data source using open and standardized technologies.

The key contributions of our research work are a formal model for information integration using semantically annotated services, and algorithms for querying distributed data sources, where services are used as data providers. We have developed a prototype based on the developed model and algorithms. Using this prototype, we have evaluated our approach by comparing it to existing approaches and technologies.

Statistical analysis has been performed based on the results of the experiment. The analysis proved that the prototype is more efficient than the compared approaches. The prototype efficiency does not differ significantly only from two most efficient approaches. Based on the conducted research, we can conclude that using services as data providers for ontology based information integration improves flexibility and does not have a negative impact on efficiency.

Keywords: *Information integration, semantic integration, ontology, service oriented architecture, semantic services*

Kazalo

Slike

Tabele

Poglavje 1

Uvod

1.1 Integracija podatkovnih virov na osnovi ontologij

Število informacij, ki jih hranimo v različnih informacijskih sistemih, raste eksponentno. Analize so pokazale, da se količina digitalno shranjenih informacij podeseteri vsakih pet let. Z upoštevanjem dejstva, da računalniki v sodobnem življenju pridobivajo osnovno podporno vlogo, preobrata trenda rasti količine informacij ni realno pričakovati[27][20].

Ključni izziv, ki se poraja v obdobju vse večje informacijske zasičenosti, je kako na čim bolj učinkovit način obvladovati informacije, ki so shranjene v različnih informacijskih virih. Nerealno je pričakovati, da so vse informacije shranjene na enem mestu ali celo znotraj enega informacijskega vira [4]. Ravno nasprotno, skozi zgodovino se je izkazalo, da so informacije večinoma porazdeljene med heterogenimi informacijskimi viri.

Področje integracije podatkov se v prvi vrsti ukvarja z izzivi, kako združiti podatke, ki so porazdeljeni v različnih informacijskih virih, ter kako zagotoviti holističen pogled nad celotnim informacijskim prostorom, neodvisno od spodaj ležeče tehnološke infrastrukture oz. posameznih informacijskih sistemov [55].

Področje integracije podatkov predstavlja večplasten problem, ki ga je potrebno obravnavati vsaj z dveh vidikov, tehnološkega in vsebinskega. Ključni izziv tehnološkega vidika je, na kakšen način ter s kakšnimi tehnologijami za-

gotoviti izmenjavo podatkov med različnimi informacijskimi viri, ki so po eni strani med seboj fizično ločeni, po drugi strani pa implementirani z različnimi tehnologijami. Osnovni cilj tehnološkega vidika je zagotovitev interoperabilnosti med heterogenimi informacijskimi sistemi.

Drugi ključni vidik integracije podatkov je vsebinski vidik. Med predstavitvijo podatkov v različnih sistemih obstajajo razhajanja. Govorimo o t.i. heterogenosti podatkov. Ločimo štiri različne vrste heterogenosti podatkov: sintaktično, terminološko, konceptualno in pragmatično heterogenost [79].

Sintaktična heterogenost nastopa zaradi uporabe različnih tehnologij ali jezikov za predstavitev podatkov (npr. relacijske tabele, preglednice, tekstovne datoteke, ipd.). Terminološka heterogenost se pojavlja zaradi različnih poimenovanj enakih stvari (npr. v enem informacijskem sistemu je obrazec predstavljen s terminom obrazec, v drugem pa s terminom formular). Konceptualna heterogenost izhaja iz različnega modeliranja informacij znotraj neke domene. Različni informacijski sistemi lahko enako področje predstavljajo z različne perspektive, z različno stopnjo podrobnosti ali z različno pokritostjo. Pragmatična heterogenost je najzahtevnejša oblika heterogenosti, predstavlja različno interpretacijo entitet v podatkovnem modelu v odvisnosti od konteksta oziroma načina uporabe.

Skladno z izpostavljenimi vidiki mora sistem za integracijo podatkov zagotoviti interoperabilnost med informacijskimi viri za transparentno izmenjavo podatkov kot tudi doseči konsenz glede strukture podatkovnega modela tako, da odpravi heterogenosti med podatkovnimi modeli različnih informacijskih virov. Formalno gledano lahko integracijo podatkov definiramo kot problem združevanja podatkov iz različnih virov ter predstavitev teh podatkov uporabniku z enotnim pogledom. Sistem za integracijo podatkov je v splošnem sestavljen iz treh komponent [14]:

- globalne sheme, ki predstavlja globalni podatkovni model,
- izvorne sheme, ki predstavlja podatkovne modele posameznih informacijskih virov, ter iz
- preslikav, ki definirajo transformacijo podatkov med izvorno in globalno shemo.

Razlika med konvencionalnimi sistemi za integracijo podatkov in sistemi za integracijo podatkov na osnovi ontologij je v globalni shemi. V sistemih za integracijo podatkov na osnovi ontologij je globalna shema definirana z ontologijo, medtem ko so izvorne sheme predstavljene s poljubno tehnologijo – odvisno od informacijskih sistemov, katerih podatke želimo integrirati.

Ontologije v računalništvu in informatiki izvirajo iz področja umetne inteligence in so sredstvo za formalno predstavitev znanja. Splošno sprejeta definicija ontologij je: »ontologija je formalna in eksplicitna specifikacija skupne konceptualizacije« [77], kjer se posamezni termini interpretirajo kot:

- konceptualizacija se nanaša na abstraktni model nekega pojava v svetu na podlagi identificiranih konceptov tega pojava,
- eksplicitna pomeni, da so tipi konceptov in omejitve pri njihovi uporabi eksplicitno definirani,
- formalna se nanaša na dejstvo, da je ontologija strojno berljiva in enoznačna,
- skupna odraža stališče, da naj ontologija zajema splošno oz. skupno znanje, to je znanje, ki ni lastno posamezniku, temveč ga je sprejela določena skupina ljudi.

Ontologije v okviru integracije podatkov služijo kot sredstvo za opis pomena informacijskih virov na ekspliciten način ter kot sredstvo za specifikacijo globalne sheme. Uporabniki izvajajo povpraševanja nad združenim modelom z uporabo konceptov iz ontologije. Naloga centralnega sistema za integracijo podatkov je, da avtonomno pridobi podatke iz lokalnih podatkovnih virov na osnovi preslikav. Konvencionalno se v okviru integracije podatkov za specifikacijo globalne sheme uporabljajo relacijski modeli. Prednosti, ki jih prinašajo ontologije kot sredstvo za specifikacijo globalne sheme so:

- predstavitev podatkovnega modela z ontologijo je človeku bližje kot relacijski model, ki je bolj primeren za strojno procesiranje,
- globalna shema je predstavljena eksplicitno, posledično so eksplicitno predstavljeni tudi posamezni koncepti ter relacije med njimi,

- ontologije omogočajo razmeroma enostavno razširljivost – v kolikor želimo v globalno shemo vključiti nove stvari, v večini primerov ni potrebno posegati v obstoječo strukturo,
- predstavitev podatkov z ontologijo je precej bolj naravna kot npr. z relacijskimi bazami, zaradi tega je v večini primerov enostavneje vključiti podatke iz »nerelacijskih virov« (npr. datoteke, el. pošta, ipd.).

1.2 Storitvena orientacija in semantično označevanje storitev

Sorodno področje integracije podatkov je integracija aplikacij. Integracija podatkov se, kot smo že izpostavili, osredotoča na integracijo podatkov med heterogenimi informacijskimi viri ter ponuja enoten vmesnik za dostop do združenih podatkov. Integracija aplikacij se na drugi strani ukvarja z integracijo heterogenih aplikacij med seboj [55]. Ključni izziv na področju integracije aplikacij je, kako določeni aplikaciji zagotoviti dostop do podatkov in funkcionalnosti druge aplikacije. V večini primerov se integracija aplikacij ne ukvarja z vzpostavitvijo globalnega pogleda.

Splošno vse bolj sprejet in razširjen pristop k integraciji aplikacij predstavlja storitveno orientirana arhitektura. Osrednji element storitveno orientirane arhitekture so, kot nakaže že samo ime, storitve. Posamezne aplikacije izpostavijo svoje funkcionalnosti in podatke v obliki storitev, do katerih je možno dostopati preko standardiziranega vmesnika. Takšna arhitektura omogoča posameznim aplikacijam dostop do podatkov ali funkcionalnosti drugih aplikacij s proženjem storitev preko standardiziranih vmesnikov. Poglavitne prednosti, ki jih prinaša pristop storitvene orientacije v primerjavi s predhodnimi pristopi integracije aplikacij, so povišanje stopnje interoperabilnosti med aplikacijami, zvišanje stopnje ponovne uporabe ter zmanjšanje sklopljenosti med aplikacijami [23]. Kljub temu, da je storitvena orientacija k področju integracije aplikacij precej prispevala, ostaja nekaj izzivov še zmeraj nerešenih.

Ena od osnovnih prednosti storitvene orientacije je ponovna uporaba. Storitve, ki so bile predhodno razvite za namen združitve dveh aplikacij, lahko

ponovno uporabimo v tretji aplikaciji. Težava, ki pri tem nastopa, je da je potrebno potencialne storitve najprej poiskati, nato pa razviti vmesnike, ki bodo storitve prožile. Naslednje področje, kjer so možne izboljšave, je avtomatizacija kompozicije storitev. Atomarne storitve morda ne ponujajo funkcionalnosti, ki bi jo potrebovali. Želene funkcionalnosti je pa včasih možno pridobiti s kompozicijo oz. zaporednim proženjem več storitev.

Oba zgoraj podana izziva naslavlja področje semantičnega označevanja storitev. Osnovna zamisel semantičnega označevanja storitev je, da posamezne storitve dodatno opišemo z metapodatki, ki označujejo pomen storitev z vidika funkcionalnosti, vhodnih in izhodnih podatkov, ter pogojev, pod katerimi lahko prožimo storitve. Ustrezno označene storitve je nato možno avtomatizirano poiskati, jih prožiti ali celo združiti v kompozicijo [31].

1.3 Integracija podatkovnih virov in semantično označene storitve

Pomanjkljivost obstoječih pristopov za integracijo podatkov na osnovi ontologij je, da zahtevajo predhodno pridobivanje podatkov. V kolikor želimo uporabiti podatke na osnovi globalne podatkovne sheme, je potrebno podatke predhodno izvoziti iz izvornih podatkovnih virov ter jih uvoziti v globalni podatkovni model. Pri tem lahko nastopijo težave z integriteto podatkov, kajti ni zagotovila, da so podatki, shranjeni v globalnem podatkovnem modelu, enaki podatkom v izvornem modelu, saj je lahko v času od zadnjega izvoza prišlo do spremembe podatkov v izvornem podatkovnem modelu [30].

Integriteto podatkov lahko zagotovimo z vzpostavitvijo mehanizmov, ki analizirajo spremembe podatkovnih virov. Takšen pristop predstavlja izredno težak problem predvsem zaradi heterogenosti informacijskih virov, ki so lahko datoteke, relacijske baze, spletne strani ali kakšna druga tehnologija za shranjevanje podatkov. Alternativna rešitev je dinamično pridobivanje podatkov v času povpraševanja globalne sheme.

Konvencionalen pristop za dinamično pridobivanje podatkov je prevajanje povpraševanj iz globalne v lokalno shemo. Kadar uporabnik izvede povpraše-

vanje po globalni shemi, sistem za integracijo podatkov prevede povpraševanje skladno z lokalno shemo ter ga izvede na lokalnem viru podatkov. Slabost tega pristopa je, da je uporaben samo takrat, kadar je mogoče prevesti povpraševanje v lokalno shemo. Klasično se takšni sistemi uporabljajo za integracijo relacijskih baz. Alternativo prevajanju povpraševanj lahko dosežemo z dinamičnim pridobivanjem podatkov, tako da posamezni informacijskih sistemi izpostavijo svoje podatke v obliki storitev. Kadar uporabnik izvede povpraševanje po globalni shemi, sistem za integracijo identificira storitve, ki nudijo potrebne podatke, jih avtonomno proži ter vrne integrirane podatke uporabniku [32].

Eden izmed ciljev doktorske disertacije je uporabiti prednosti, ki jih prinašajo semantično označene storitve, za namene integracije podatkov na osnovi ontologij. Predpostavljamo, da je semantično označene storitve možno uporabiti kot mehanizem za pridobivanje podatkov iz drugih informacijskih sistemov ter jih avtomatizirano prožiti. Osnovna raziskovalna vprašanja, ki se pri tem pojavljajo, so:

- Na kakšen način je potrebno modelirati ontologije in označiti storitve, da je mogoče podatke pridobiti dinamično v času povpraševanja?
- Kakšen je algoritem za povpraševanje po globalni shemi, ki omogoča povpraševanje po podatkih, kjer je del podatkov shranjen znotraj ontologije, del podatkov pa je porazdeljen po drugih informacijskih sistemih, do katerih je možno dostopati s proženjem storitev?
- Na kakšen način je potrebno semantično označiti storitve, da jih je možno avtomatizirano prožiti, in podatke, ki jih storitve vrnejo, avtomatizirano integrirati v globalno shemo?
- Na kakšen način je možno oceniti učinkovitost sistemov za integracijo podatkov na osnovi ontologij?

1.4 Cilji doktorske disertacije

1.4.1 Teza doktorske disertacije

Cilj doktorske disertacije je definiranje in razvoj modela za integracijo podatkovnih virov na osnovi ontologij z uporabo semantično označenih storitev, ki omogočajo dinamično in avtomatizirano pridobivanje ter integracijo podatkov v času izvajanja povpraševanj. Za naše raziskovalno delo smo postavili naslednje hipoteze:

H1 Semantično označene storitve omogočajo distribuirano in dinamično pridobivanje podatkov v sistemih za integracijo podatkovnih virov na osnovi ontologij.

H2 Sistemu za integracijo podatkovnih virov na osnovi ontologij, ki uporablja semantično označene storitve za pridobivanje podatkov, je možno ovrednotiti učinkovitost ter ga na ta način primerjati s konvencionalnimi pristopi.

H3 Vpeljava dodatnega nivoja semantično označenih storitev v večini primerov ohranja učinkovitost sistemov za integracijo podatkov na osnovi ontologij.

1.4.2 Izvirni znanstveni prispevki

Izvirni znanstveni prispevki našega raziskovalnega dela so:

- Model za integracijo podatkovnih virov na osnovi ontologij, ki uporablja semantično označene storitve za dinamično pridobivanje in integracijo podatkov.
- Algoritem za povpraševanje po distribuiranih podatkovnih virih v obliki storitev na osnovi ontologije, ki predstavlja globalno shemo.
- Model za evalvacijo učinkovitosti sistemov za integracijo podatkov na osnovi ontologij.
- Sistematični pregled področja integracije podatkov na osnovi ontologij.

- Sistematični pregled področja semantičnega označevanja storitev.
- Prototipni sistem za integracijo podatkovnih virov na osnovi ontologij.
- Ovrednotenje učinkovitosti razvitega modela ter podrobna analiza primerjave z obstoječimi pristopi.

1.5 Struktura disertacije

Doktorska disertacija je organizirana v šest poglavij. Drugo poglavje podaja pregled raziskovalnega področja ter teoretično osnovo doktorske disertacije. V njem povzamemo dosežke na področju integracije podatkovnih virov na osnovi ontologij, podamo teoretično ozadje ontologij ter semantičnih tehnologij, povzamamo področje semantičnih spletnih storitev ter podamo pregled sorodnih del.

Tretje poglavje predstavlja model, ki smo ga razvili v okviru doktorske disertacije. V njem predstavimo teoretični model, postopek transformacije povpraševanj ter algoritme za distribuirano izvajanje povpraševanj nad storitvami. Četrto poglavje podaja model za evalvacijo sistemov za integracijo heterogenih podatkovnih virov, predstavlja eksperiment za primerjavo učinkovitosti sistemov za integracijo heterogenih podatkovnih virov na osnovi ontologij, ter podaja podrobno analizo rezultatov eksperimenta.

Peto poglavje vsebuje razpravo. Na osnovi rezultatov raziskovalnega dela podamo odgovore na raziskovalna vprašanja ter analiziramo hipoteze, ki smo jih postavili v okviru doktorske disertacije. Zadnje, šesto pa zaključi doktorsko disertacijo.

Poglavje 2

Pregled raziskovalnega področja

2.1 Integracija podatkov

Področje integracije podatkov naslavlja izzive, ki se porajajo, kadar aplikacije izvajajo povpraševanja nad več avtonomnimi in heterogenimi podatkovnimi viri [38]. Podatki, ki jih želimo združiti, se lahko nahajajo v klasičnih sistemih za hrambo podatkov (npr. relacijske baze), lahko pa se nahajajo tudi v kakšni drugačni obliki (npr. elektronska pošta, preglednice, XML datoteke, objektne baze,...). Osnovi problem integracije podatkov je združevanje podatkov, ki se nahajajo na različnih podatkovnih virih, in predstavitev teh podatkov z enotnim pogledom [54].

V znanstveni kot tudi v strokovni literaturi se pojavljajo predvsem trije osnovni pristopi k integraciji podatkov: navigacijski pristop, pristop podatkovnega skladišča in mediatorski pristop [43][81][6]. Posamezni sistemi za integracijo podatkov lahko podatke hranijo lokalno, tako da jih predhodno prekopirajo iz izvornih virov in jih nato iz svoje hrambe posredujejo uporabniku (govorimo o tako imenovani materializaciji podatkov). Alternativna možnost je, da sistem uporabniku podatke posreduje neposredno. Tako v času izvajanja opravlja povpraševanja na izvornih virih. V kolikor sistem prekopira podatke in hrani lokalno kopijo, rečemo, da je podatke materializiral. Materializacija podatkov predstavlja ključno razliko med posameznimi pristopi. Pristop podatkovnega skladišča temelji na materializaciji podatkov, medtem ko preostala

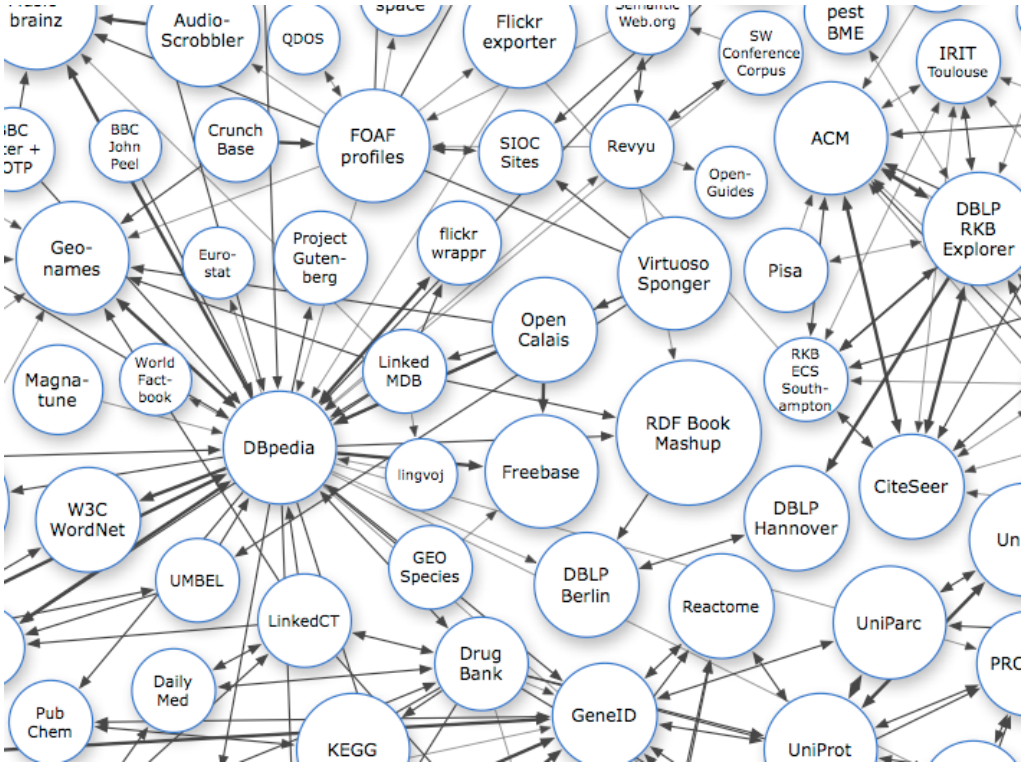
pristopa dostopata do podatkov neposredno.

Navigacijski pristop temelji na statičnih povezavah med podatki v posameznih podatkovnih virih [43]. Tipično v takšnem sistemu uporabnik brska po podatkih. V kolikor za posamezen podatek obstajajo še informacije o tem podatku v drugem informacijskem viru, sistem nudi povezavo do tega vira. Gre za najpreprostejšo obliko integracije podatkov, saj do same integracije podatkov ne pride na nivoju informacijskega sistema, temveč podatke dejansko združi uporabnik sam.

V to kategorijo lahko uvrstimo projekt Linking open data (LOD). LOD ni predstavnik integracije podatkov v klasičnem pomenu besede, saj ne ponuja sistema za integracijo podatkov, temveč nudi zgolj povezave med različnimi podatkovnimi viri, ki so prosto dostopni na svetovnem spletu. Povezave so definirane na nivoju posameznih zapisov med podatkovnimi viri. Izpis 2.1 prikazuje konkretne povezave med podatki DBPedie (Wikipedia, zapisana s tehnologijami semantičnega spleta) in spletnim servisom za hranjenje fotografij Flickr-om. Izpis predstavlja del RDF datoteke, ki definira povezave med člankom o Beli hiši na DBPedii in slikami Bele hiše na Flickru. Slika 2.1 prikazuje povezave med posameznimi informacijskimi viri v okviru LOD.

```
<rdf:Description rdf:about="http://dbpedia.org/resource/White_House">
   <foaf:depiction
    rdf:resource="http://farm1.static.flickr.com/2/1875227_a611feceaf_m.jpg"/>
</rdf:Description>
<rdf:Description rdf:about="http://farm1.static.flickr.com/2/1875227_a611fece
af_m.jpg">
   <foaf:page
    rdf:resource="http://www.flickr.com/photos/41894170584@N01/1875227"/>
</rdf:Description>
<rdf:Description rdf:about="http://dbpedia.org/resource/White_House">
   <foaf:depiction
    rdf:resource="http://farm1.static.flickr.com/2/1875221_22ddc6ad95_m.jpg"/>
</rdf:Description>
<rdf:Description rdf:about="http://farm1.static.flickr.com/2/1875221_22ddc6ad
95_m.jpg">
   <foaf:page
    rdf:resource="http://www.flickr.com/photos/41894170584@N01/1875221"/>
</rdf:Description>
```

Izpis 2.1: Primer povezav v LOD med DBPedio in Flickrom



Slika 2.1: Povezave med posameznimi podatkovnimi viri v okviru LOD

Pristop podatkovnega skladišča uporablja materializacijo podatkov. V večini primerov je lokalna shramba relacijska podatkovna baza, kamor uvozimo podatke iz podatkovnih virov, ki jih želimo integrirati. Pri tem pristopu so ključna tako imenovana ETL orodja (Extract-Transform-Load). Ta skrbijo za izvoz podatkov iz oddaljenega podatkovnega vira, transformacijo podatkov v obliko, skladno s shemo podatkovnega skladišča, in za uvoz podatkov v podatkovno skladišče [6].

Ključni izziv pri pristopu podatkovnega skladišča predstavlja materializacija podatkov. Sistem za integracijo podatkov deluje na kopiji podatkov, tako da v času povpraševanja ni zagotovila, da so podatki v podatkovnem skladišču skladni s podatki v izvornih podatkovnih virih. Težave se lahko pojavijo tudi zaradi lastništva podatkov. V kolikor subjekt, ki vzpostavlja podatkovno skladišče, ni lastnik izvornega podatkovnega vira, lahko pride do težav zaradi pravic uporabe ali kopiranja podatkov.

Mediatorski pristop ponuja uporabnikom globalno shemo, na osnovi katere uporabniki izvajajo povpraševanja. Oddaljeni oz. izvorni podatkovni viri vsebujejo podatke, medtem ko globalna shema predstavlja usklajen, integriran virtualen pogled na spodaj ležeče podatkovne vire [54]. Ključen izziv pri tem pristopu predstavlja modeliranje relacij med podatkovnimi viri in globalno shemo. Mediatorski pristop ne uporablja materializacije podatkov, temveč v času izvajanja dinamično in transparentno transformira povpraševanja. Vhodna povpraševanja, ki so podana na osnovi globalne sheme, transformira v povpraševanja, ki so skladna z shemami izvornih podatkovnih virov. Na ta način sta hkrati zagotovljena:

- **transparentnost uporabe** - uporabnikom ni treba poznati shem izvornih podatkovnih virov, ni jim potrebno vedeti, kje se nahajajo izvorni podatkovni viri, niti jim ni potrebno vedeti kateri izvorni podatkovni viri se bodo uporabili pri njihovem povpraševanju,
- **integriteta podatkov** - mediatorski pristop dostopa do izvornih podatkovnih virov neposredno v času izvajanja povpraševanj, zaradi tega uporabnik venomer dobi aktualne podatke.

Navigacijski pristop lahko, v primerjavi z ostalima pristopoma, označimo za

inferiornega, saj dejansko integracijo podatkov izvaja uporabnik sam. Sistem zgolj ponuja povezavo med materializiranimi podatki. V industriji je najbolj uporabljan pristop podatkovnega skladišča, saj je najbolj enostaven za implementacijo in trenutno najbolj preizkušen na realnih problemih. Mediatorski pristop je precej bolj aktualen v raziskovalnih sferah, saj še zmeraj predstavlja množico odprtih izzivov. Kljub temu pa se tudi že vse bolj uveljavlja v industriji [6].

V literaturi obstaja več izrazov, ki se nanašajo na mediatorski pristop k integraciji podatkov. V angleškem izrazoslovju je najpogosteje zaslediti izraz **information integration**, redkeje pa tudi **data integration** [49]. V industriji se je za mediatorski pristop uveljavil predvsem izraz **enterprise information integration** (**EII**). EII je postal precej razširjena in obetavna panoga znotraj informatike [6] [38]. V slovenskem izrazoslovju se predvsem pojavljata izraza **integracija podatkov** in **integracija podatkovnih virov**. V okviru te doktorske disertacije bomo za mediatorski pristop k integraciji podatkov dosledno uporabili izraz integracija podatkovnih virov.

Podatkovna skladišča in integracija podatkovnih virov predstavljajo dva osnovna nasprotujoča si pristopa k integraciji podatkov. V okviru naše doktorske disertacije smo se ukvarjali izključno z integracijo podatkovnih virov (mediatorski pristop).

2.1.1 Integracija podatkovnih virov

Sistemi za integracijo podatkovnih virov so tipično sestavljeni iz globalne sheme, izvornih podatkovnih virov ter preslikav med globalno shemo in shemami izvornih podatkovnih virov. **Globalna shema** je shema, na podlagi katere uporabniki podajajo povpraševanja. Le-ta predstavlja usklajen, integriran virtualen pogled na spodaj ležeče podatkovne vire. **Lokalna shema** je shema podatkovnega modela posameznega izvornega podatkovnega vira. Sisteme za integracijo podatkovnih virov delimo v tri kategorije [54][38]:

- **LAV** (ang. local-as-view, slv. lokalno kot pogled): globalna shema je izražena neodvisno od izvornih podatkovnih virov, relacije med globalno shemo in izvornimi podatkovnimi viri so vzpostavljene z definiranjem vsakega izvornega vira kot pogleda na globalno shemo,

- **GAV** (ang. global-as-view, slv. globalno kot pogled): globalna shema je izražena s termini iz izvornih podatkovnih virov,

- **P2P** (ang. peer-to-peer, slv. vsak z vsakim): pristop je nastal po vzoru deljenja datotek po principu vsak z vsakim. Skladno s to paradigmo, globalna shema ne obstaja. Posamezni podatkovni viri izrazijo semantične preslikave le za nekaj sosednjih podatkovnih virov. Kompleksnejša integracija se razvije, ko sistem sledi t.i. semantičnim sledem med posameznimi podatkovnimi viri.

Integracija podatkovnih virov na osnovi ontologij (ang. ontology based information integration) je pristop k integraciji podatkovnih virov, kjer je globalna shema izražena z ontologijo. V nadaljevanju predstavljamo teoretični model za integracijo podatkovnih virov, ki ga je razvil Lenzerini [54]. Osnovne komponente sistema za integracijo podatkovnih virov so globalna shema, podatkovni viri in preslikave med globalno shemo ter izvornimi podatkovnimi viri. Skladno s tem je Lenzerini formaliziral sistem za integracijo podatkovnih virov I kot trojček

$$I = \langle G, S, M \rangle \tag{2.1}$$

kjer je:

- G - globalna shema, izražena v jeziku L_G nad abecedo A_G. Abeceda vsebuje simbol za vsak element iz G (relacijo, v kolikor je G relacijska shema, razred, v kolikor je G objektno orientirana shema ali ontologija,...).

- S - lokalna shema, izražena v jeziku L_S nad abecedo A_S. Abeceda vsebuje simbol za vsak element iz množice podatkovnih virov.

- M - preslikava med G in S, ki je definirana kot množica izjav oblike

$$\begin{aligned} q_S &\rightarrow q_G \\ q_G &\rightarrow q_S \end{aligned} \tag{2.2}$$

kjer je preslikava $q_S \rightarrow q_G$ definirana kot preslikava povpraševanja q, izražena v L_S, v enakovredno povpraševanje v L_G. Analogno temu preslikava $q_G \rightarrow q_S$ transformira povpraševanje q, izraženo v L_G, v enakovredno povpraševanje v L_S.

Lokalno kot pogled (LAV)

V kolikor je sistem za integracijo podatkovnih virov $I = \langle G, S, M \rangle$ zasnovan na LAV pristopu, potem preslikava M ustvari relacijo med vsakim elementom s iz lokalne sheme S in povpraševanjem q_G nad globalno shemo G. LAV preslikave vsebujejo izjave, ki imajo natanko en simbol iz abecede A_S. Iz tega sledi, da je LAV preslikava množica izjav, kjer za vsak element s iz S obstaja ena izjava oblike:

$$s \rightarrow q_G$$

LAV pristop je zasnovan na zamisli, da s preslikavami vsakemu element s določimo pogled q_G nad globalno shemo G. Poenostavljeno lahko rečemo, da posamezni element izvornega vira definiramo kot pogled nad globalno shemo.

Globalno kot pogled (GAV)

Pri pristopu GAV preslikava M definira relacijo med vsakim elementom g iz G in povpraševanje q_S nad S. GAV preslikave vsebujejo izjave, ki imajo natanko en simbol iz abecede A_G. Preslikava GAV je množica izjav, kjer za vsak element g iz G obstaja natanko ena izjava oblike:

$$g \rightarrow q_S$$

Pristop GAV je osnovan okoli ideje, da za vsak element g iz globalne sheme G definiramo povpraševanje q_S nad lokalnimi podatkovnimi viri. Iz tega sledi, da za vsak element g preslikava eksplicitno definira način pridobivanja podatkov iz izvornih podatkovnih virov.

Vsak izmed pristopov ima svoje prednosti. LAV pristop je bolj fleksibilen, saj je lažje razširljiv. V kolikor želimo dodati nov podatkovni vir, ni potrebno redefinirati globalne sheme, temveč zgolj dodamo povpraševanja q_G za vsak na

novo dodan element s. V kolikor želimo dodati podatkovni vir v GAV sistemu, je potrebno spremeniti globalno shemo, saj je le ta definirana kot pogled nad izvornimi viri[54].

Po drugi strani je obdelovanje povpraševanj v GAV precej bolj enostavno kot v LAV. Pri pristopu GAV, že same preslikave neposredno specificirajo katera povpraševanja nad izvornimi podatkovnimi viri se skladajo s posameznim elementom iz globalne sheme. Večina GAV pristopov uporablja enostavno strategijo odvijanja. Pri LAV je položaj obrnjen, saj so lokalne sheme definirane s pogledi nad globalno shemo. Ko uporabnik poda povpraševanje na osnovi globalne sheme, ni trivialno izpeljati, katere izvorne vire (in na kakšen način) je potrebno prožiti, da lahko odgovorimo na povpraševanje, saj imamo opravka z nepopolnimi informacijami [54].

V kolikor primerjamo oba pristopa z vidika načrtovanja, lahko sklenemo, da je pri pristopu LAV osnovna naloga specifikacija vsebine izvornih virov na osnovi globalne sheme. GAV predstavlja pri načrtovanju večji izziv, saj je potrebno za vsak element globalne sheme določiti, kako pridobiti podatke iz izvornih virov na osnovi povpraševanj.

2.2 Ontologije in predstavitev znanja

Ontologije so eden od pristopov predstavitve znanja (ang. *knowledge representation* - KR). Osnovni namen, s katerim se ukvarja področje predstavitve znanja, je snovanje računalniških sistemov, ki so sposobni sklepati na strojno-berljivi predstavitvi sveta, in sicer na način, čim bolj podoben človeškemu sklepanju [33]. Davis je podal podrobnejšo definicijo, kjer je predstavitev znanja opisal na podlagi petih vlog, v katerih se le-ta pojavlja [19]:

- Predstavitev znanja je najbolj osnoven **nadomestek**, ki omogoča entitetam, da določijo posledice z razmišljanjem in ne z ukrepanjem (s sklepanje o svetu namesto z izvajanjem dejanj v njem).
- KR je **množica ontoloških obvez** (v smislu filozofske definicije ontologije, npr. odgovarja na vprašanja, kot so: "Pod kakšnimi pogoji naj razmišljam o svetu?").

- KR je **delna teorija inteligentnega sklepanja**, ki je izražena s tremi komponentami: (i) osnovna konceptualizacija inteligentnega sklepanja, (ii) množica sklepov, ki predstavljajo sankcije in (iii) množica sklepov, ki jih priporoča.
- KR je **medij za pragmatično učinkovito procesiranje** – procesno okolje, v katerem je izvršeno razmišljanje/sklepanje. KR prispeva k pragmatični učinkovitosti v obliki smernic za organizacijo informacij z namenom lažjega podajanja priporočenih sklepov.
- KR je **medij človeške izraznosti** – ponuja jezik, v katerem lahko izrazimo stvari o svetu.

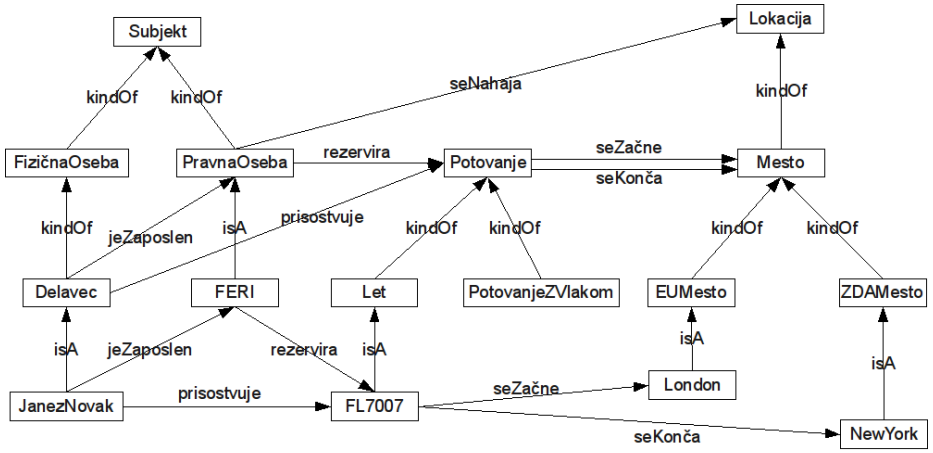
Povzete vloge KR so podrobneje opisane v [19]. Tehnologije, ki zadostujejo potrebam opisanih vlog in se uporabljajo za namene predstavitve znanja, so semantične mreže, pravila, logika in nenazadnje ontologije. Opis načinov predstavitve znanja in za te namene uporabljene formalizme povzemamo po [29] in [33].

2.2.1 Semantične mreže

Semantične mreže se osredotočajo na izražanje taksonomskih struktur kategorij objektov in relacij med njimi. Semantična mreža je graf, katere vozlišča predstavljajo koncepte, povezave pa relacije med temi koncepti. Semantične mreže omogočajo strukturno predstavitev izjav neke domene. Primer semantične mreže iz domene poslovnega potovanja je prikazan na sliki 2.2 [33].

Semantične mreže ponujajo sredstva za abstrakcijo iz naravnega jezika v obliko, ki je bolj primerna za strojno procesiranje. Slika 2.2 predstavlja znanje iz domene poslovnih potovanj. V semantični mreži je predstavljeno znanje, ki je zajeto v naslednjem prostem tekstu (primer smo povzeli in priredili po [33]).

> *Delavci, ki so zaposleni pri pravnih osebah, so fizične osebe. Pravne in fizične osebe so subjekti. Pravne osebe rezervirajo potovanja za svoje zaposlene. Potovanja so lahko leti ali potovanja z vlakom. Vsako potovanje se začne ali konča v mestu, ki je bodisi v EU ali ZDA. Pravne*



Slika 2.2: Primer semantične mreže

osebe se nahajajo na poljubni lokaciji.
Pravna oseba FERI je rezervirala let FL7007 iz Londona v New York za Janeza Novaka.

Semantična mreža iz podanega primera vsebuje tudi relacije, ki niso vezane na izbrano domeno, temveč so namenjene predstavitvi taksonomije konceptov. Te vrste relacij nismo slovenili, njihov pomen pa je nasleden:

kindOf povezuje med seboj dva koncepta in predstavlja generalizacijo oz. specializacijo. V našem primeru koncept `Let` predstavlja specializacijo koncepta *Potovanje*.

partOf predstavlja relacijo med dvema konceptoma, ki nakazuje povezavo del-celota. Primer takšne relacije, ki ni zajet v podanem primeru, bi bil, da je `Krilo` del `Letala`.

isA določa primerke razredov. Izvorna vozlišča te relacije predstavljajo primerke koncepta, ki je ciljno vozlišče (`NewYork` je primerek `ZDAMesto`).

2.2.2 Pravila

Druga možnost strukturirane predstavitve znanja so pravila. Pravila izražajo kavzalnost in imajo obliko *IF-THEN* konstruktov. Za zgoraj podan primer bi bila predstavitev znanja v obliki pravil naslednja:

1. IF je nekaj let THEN je tudi potovanje
2. IF neka oseba prisostvuje na potovanju, ki ga je rezerviralo neko podjetje THEN ta oseba je zaposlena v tem podjetju
3. FACT[1] oseba JanezNovak prisostvuje na letu, ki ga je rezervirala pravna oseba FERI
4. IF začetna in končna lokacija potovanja sta skupaj THEN potovanje je z vlakom

Predstavljena pravila so zapisana v neformalni obliki in jih ni možno strojno procesirati. Za formalno predstavitev pravil se uporabljajo logika (predikati in spremenljivke). Unarni predikati predstavljajo koncepte, medtem ko binarni ali večarni predikati predstavljajo relacije med koncepti, podobno, kot pri semantičnih mrežah. Pogosto se IF del pravil imenuje telo pravila, medtem ko se THEN del pravil imenuje glava pravila. Dejstva so posebna vrsta pravil, ki imajo telo pravila prazno. Pravila, ki so predstavljena na takšen način, lahko strojno procesiramo. Zgoraj navedena pravila lahko v formalni obliki predstavimo na naslednji način:

1. $Let(?p) \rightarrow Potovanje(?p)$
2. $Potovanje(?p) \wedge rezervira(?f, ?p) \wedge PravnaOseba(?f) \wedge prisostvuje(?o, ?p)$ $\wedge\, FizicnaOseba(?o) \rightarrow Delavec(?p) \wedge jeZaposlen(?p, ?f)$
3. $\rightarrow FizicnaOseba(JanezNovak) \wedge prisostvuje(JanezNovak, FL7007)$ $\wedge\, Let(FL7007) \wedge rezervira(FERI, FL7007) \wedge PravnaOseba(FERI)$
4. $Potovanje(?p) \wedge seZacne(?zacetek) \wedge seKonca(?konec)$ $\wedge\, jeBlizu(?zacetek, ?konec) \rightarrow PotovanjeZVlakom(?p)$

Pravila, podana v zgornjem primeru, so dovolj enostavna, da ne potrebujejo dodatne razlage. Izpostavili bi samo 4. pravilo, ki izraža, da se naj uporabi potovanje z vlakom, v kolikor sta začetna in končna lokacija blizu. Semantične

[1]FACT označuje dejstvo, to je znanje, ki je znano že pred izvajanjem pravil (eksplicitno znanje)

mreže so zelo dobre pri taksonomizaciji konceptov in relacij med njimi, vendar so šibke pri natančnejši specifikaciji posameznih konceptov. Tako recimo ne omogočajo predstavitve znanja, ki je izraženo s pravilom 4.

2.2.3 Logika

Semantične mreže in pravila lahko formalno predstavimo z uporabo logike, ki definira natančen pomen. Brez natančne formalizacije je predstavitev znanja *neopredeljiva* in *dvoumna*, kar onemogoča *strojno procesiranje*. Logika je tako sredstvo, s katerim lahko predstavimo tako semantične mreže, kot tudi pravila, na formalen način, ki omogoča strojno procesiranje. Ni nujno, da se pri predstavitvi znanja omejimo zgolj na semantične mreže ali pravila. Uporabimo oz. definiramo lahko poljubne logične izraze [33].

Za ponazoritev dvoumnosti izpostavimo relacijo `rezervira` iz semantične mreže, ki povezuje koncepta `PravnaOseba` in `Potovanje`. Iz predstavljene semantične mreže ni razvidno, ali rezervira potovanja vsaka pravna oseba ali samo nekatere. Prav tako ni jasno, ali pravne osebe rezervirajo potovanja samo za svoje zaposlene ali tudi za zaposlene v drugih organizacijah. V nadaljevanju bomo predstavili formalizme, ki se najpogosteje uporabljajo za predstavitev znanja, izpostavljeno dvoumnost pa odpravljamo v primeru 3, v razdelku o logiki prvega reda.

2.2.4 Sklepanje

V okviru predstavitve znanja je naloga sklepanja izpeljevanje novih izjav na podlagi že podanih izjav. Na ta način pridobimo novo znanje na podlagi podanega (implicitno znanje), kar je tudi eden od osnovnih namenov predstavitve znanja. Algoritme sklepanja opisujemo z dvema karakteristikama [70]:

- **pravilnost** (ang. *soundness*) – vse izjave, ki jih algoritem izpelje, so pravilne (v kolikor mu podamo pravilne predpostavke),
- **popolnost** (ang. *completeness*) – algoritem je sposoben izpeljati *vse* pravilne izjave v bazi znanja.

Pri procesu sklepanja stremimo k *pravilnim* in *popolnim* algoritmom sklepanja. Obstaja še karakteristika za baze znanja, to je *konsistentnost*. V kolikor baza znanja ne vsebuje nasprotujočih si dejstev, pravimo, da je *konsistentna*. V kolikor bi baza znanja iz zgornjega primera vsebovala dejstva, da je primerek `FERI` tako `PravnaOseba` kot `FizičnaOseba`, potem baza znanja ne bi bila konsistentna, saj se koncepta `PravnaOseba` in `FizičnaOseba` izključujeta.

2.2.5 Formalizmi za predstavitev znanja

2.2.5.1 Logika prvega reda

Najosnovnejši in najbolj znan logični formalizem za predstavitev znanja je logika prvega reda (ang. first-order logic). Koncepti so v logiki prvega reda predstavljeni z unarnimi predikati, medtem ko so relacije predstavljene z binarnimi ali večarnimi predikati. Osnov logike prvega reda se v okviru tega dela ne bomo lotevali, za ilustracijo predstavitve znanja z logiko prvega reda, podajamo nekaj primerov, povzetih po [33].

Primer 1 Za prvi primer lahko podamo relacijo `kindOf` iz že podane semantične mreže, ki povezuje koncepta `Delavec` in `FizičnaOseba`. Omenjena relacija izraža, da je delavec specializacija koncepta fizična oseba; torej je vsak delavec tudi fizična oseba. Z uporabo logike prvega reda lahko izrazimo omenjeni primer na naslednji način.

$$\forall x : (Delavec(x) \rightarrow FizicnaOseba(x)) \tag{2.3}$$

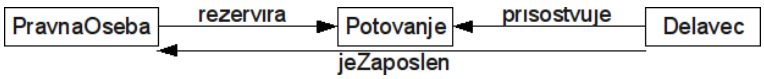
Primer 2 Za drug primer vzamimo relacijo `rezervira` iz semantične mreže, ki povezuje koncepta `PravnaOseba` in `Potovanje`. Ta relacija izraža, da neka pravna oseba rezervira potovanje. Relacijo lahko izrazimo tudi z logičnimi izrazi 2.4 in 2.5.

$$\forall x, y : (rezervira(x, y) \rightarrow PravnaOseba(x) \wedge Potovanje(y)) \tag{2.4}$$

$$\forall x \exists y : (Potovanje(x) \rightarrow PravnaOseba(y) \wedge rezervira(y, x)) \tag{2.5}$$

Izraz 2.4 izraža t.i. domeno (ang. *domain*) in obseg (ang. *range*) relacije `rezervira`. Glede na izraz je domena relacije `rezervira` koncept `PravnaOseba`, medtem ko je obseg `Potovanje`. Izraz 2.5 izraža, da za vsako `Potovanje` obstaja `PravnaOseba`, ki je rezervirala potovanje. Glede na to predstavitev znanja lahko potovanja rezervirajo samo pravne osebe. Takšna predstavitev relacije `rezervira` odpravlja vse možne dvoumnosti, ki so izhajale iz semantične mreže, predstavljene zgolj z grafom (slika 2.2).

Primer 3 Za naslednji primer vzamimo odsek semantične mreže, ki je prikazan na sliki 2.3. Ta izpis izraža, da pravne osebe rezervirajo potovanja, na katerih prisostvujejo delavci. Delavci so zaposleni pri pravnih osebah.



Slika 2.3: Izpis semantične mreže, predstavljene na sliki 2.2

Prikazan izpis lahko predstavimo z logičnim izrazom 2.6. Izraz s predikatom `delodajalec(y)` izraža, da pravne osebe rezervirajo potovanja samo za svoje zaposlene in s tem odpravlja dvoumnosti semantične mreže, predstavljene z grafom.

$$\forall x : \exists y : (Potovanje(x) \rightarrow Delavec(y) \wedge prisostvuje(y, x) \\ \wedge rezervira(delodajalec(y), x)) \tag{2.6}$$

Primer 4 Formaliziranje pravil v logiko prvega reda je precej enostavno. Izraz 2.7 prikazuje pravilo 4, podano v poglavju o pravilih, z uporabo logike prvega reda.

$$\forall x, y, z : (Potovanje(x) \wedge seZacne(x, y) \wedge seKonca(x, z) \wedge jeBlizu(y, z) \\ \rightarrow PotovanjeZVlakom(x)) \tag{2.7}$$

Predstavitev znanja z logiko prvega reda omogoča proces sklepanja (ang.

deduction, inferencing). Sklepanje omogoča dostop do znanja, ki ni eksplicitno podano, ampak je implicitno predstavljeno s teorijo [33]. Za algoritme sklepanja želimo, da so *pravilni* in *popolni*. Za logiko prvega reda obstajajo pravilni in popolni algoritmi.

Področje predstavitve znanja podaja t.i. lastnost **rešljivost** za primerjavo formalizmov. Za posamezen formalizem pravimo, da je rešljiv, če algoritem za proces sklepanja vrne *pravilen rezultat v* ***končnem času***.

Logika prvega reda je delno rešljiva, kar pomeni, da je pravilne rezultate možno dobiti v končnem času, medtem ko se preverjanje nepravilnih izjav ne konča v končnem času. Tako lahko za nek problem proces sklepanja teče in teče ter se celo nikoli ne konča. V kolikor proces sklepanja teče dalj časa, ne moremo določiti, ali bo proces sklepanja izpeljal pravilno izjavo, ali pa je zaradi nepravilne izjave zašel v neskončno zanko.

2.2.5.2 Opisna logika

Logika prvega reda ni rešljiva, zaradi tega ni primerna za predstavitve znanja. Zaradi tega je bila izpeljana opisna logika (ang. *description logic*), ki uporablja rešljive dele logike prvega reda. Opisna logika je izrazno dovolj močna, da je postala glavna paradigma za predstavitev znanja. Osrednjo vlogo ima opisna logika v semantičnem spletu.

Opisna logika (DL) obstaja v več različicah, ki predstavljajo kompromis med izraznostjo in zmogljivostmi pogonov sklepanja. Osnovna različica opisne logike je $\mathcal{ALC}$. Opisno logiko sestavljajo koncepti, primerki in relacijami med njimi. Primerki v DL so primerljivi s spremenljivkami, koncepti pa z unarnimi predikati v logiki prvega reda. $\mathcal{ALC}$ DL podpira naslednje Boolove operatorje [2]:

- konjunkcija $\sqcap$, binarni predikat med dvema konceptoma,
- disjunkcija $\sqcup$, binarni predikat med dvema konceptoma,
- negacija $\neg$, unarni predikat.

Koncepti, izpeljani z logičnimi izrazi na podlagi zgoraj podanih operatorjev, so prav tako koncepti. Lahko so imenovani ali anonimni. Če privzamemo, da sta

C in D koncepta, potem lahko definiramo nov koncept kot izjavo $C \sqcap \neg D$. Tako definiran koncept lahko pretvorimo v izjavo $C(x) \wedge \neg D(x)$.

V okviru $\mathcal{ALC}$ DL lahko definiramo povezave med koncepti na naslednji način:

- vključenost $\sqsubseteq$, v logiki prvega reda je to implikacija $\rightarrow$, najpogosteje se uporablja za izražanje specifikacije,
- inverzna vključenost $\sqsupseteq$, v okviru logike prvega reda je to inverzna implikacija $\leftarrow$,
- ekvivalenca $\equiv$, v okviru logike prvega reda izražena kot ekvivalenca $\leftrightarrow$.

Proste spremenljivke v izjavah DL so univerzalno kvantificirane. Izraz $E \equiv C \sqcap \neg D$ lahko pretvorimo v naslednji izraz $\forall x : E(x) \rightarrow C(x) \wedge \neg D(x)$ logike prvega reda.

Opisna logika $\mathcal{ALC}$ definira dva posebna koncepta $\bot$ in $\top$. $\bot$ predstavlja prazno množico in je definiran kot $\bot \equiv C \sqcap \neg C$. $\top$ predstavlja univerzalni koncept, torej koncept, ki vključuje vse stvari. Definiran je kot $\top \equiv C \sqcup \neg C$. V OWL jeziku, enem od jezikov opisne logike, sta $\bot$ in $\top$ definirana kot koncept `Nothing` in `Thing`.

Opisna logika dopušča uporabo kvantifikatorjev tudi na relacijah t.i. omejitve vlog (ang. *role restriction*). Omejitev vlog je lahko predstavljena z univerzalnim ali eksistenčnim kvantifikatorjem, npr. $\forall r.C$ ali $\exists r.C$, kjer je r relacija oz. vloga, C pa koncept, ki predstavlja domeno relacije.

Za primer podajamo dve izjavi v DL iz domene poslovnih potovanj.

$$Delavec \sqsubseteq FizicnaOseba \tag{2.8}$$

$$Potovanje \sqsubseteq \exists rezervira.(PravnaOseba \sqcup FizicnaOseba) \tag{2.9}$$

Izraz 2.8 predstavlja specializacijo oz. generalizacijo in izraža, da je vsak `Delavec` tudi `FizicnaOseba`. Izraz 2.9 vsebuje znanja, da vsako potovanje rezervira ali `PravnaOseba` ali `FizicnaOseba`.

Posebnost opisne logike je, da deli predstavitev znanja in s tem model na dva dela:

- terminološki okvir(*TBox*) – teorija predstavitve znanja neke domene (definicija konceptov in relacij med njimi),
- izjavni okvir (*ABox*) – konkretni primerki razredov.

Ta delitev je bila vpeljana zaradi izboljšanja učinkovitosti pogonov sklepanj, saj zahtevajo teoretične osnove in primerki različne načine sklepanj o predstavljenih podatkih. TBox izjave predstavljata izraza 2.8 in 2.9. Primeri ABox izjav so predstavljeni z izrazoma 2.10 in 2.11. Izraz 2.10 izraža, da je FL7007 primerek razreda `Let`, izraz 2.11 pa, da je let FL7007 rezerviral FERI.

$$Let(FL7007) \tag{2.10}$$

$$rezervira(FL7007, FERI) \tag{2.11}$$

$\mathcal{ALC}$ različica opisne logike ni dovolj izrazna za univerzalno predstavitev znanja, zato obstaja več različic. V tabeli 2.1 so predstavljeni vsi možni konstrukti opisne logike, ki jih je možno kombinirati v jezik opisne logike. Kot je možno razbrati iz tabele, $\mathcal{ALC}$ jezik vsebuje naslednje konstrukte: koncept, imenske vloge, presek, vrednostne omejitve, omejeno eksistenčno kvantifikacija, univerzalni koncept $\top$, prazen koncept $\bot$, atomarno negacijo in negacijo[1].

2.2.5.3 Logično programiranje

Logično programiranje se je včasih obravnavalo kot način uporabe logike prvega reda kot programskega jezika. Da bi zagotovili učinkovito procesiranje, so logični izrazi spremenjeni v t.i. Hornove stavke, ki predstavljajo disjunkcijo predikatov. Tipičen Hornov stavek je recimo $Potovanje(p) \vee \neg Let(P)$, ki ga lahko pretvorimo v izjavo logike prvega reda $\forall p : Potovanje(p) \leftarrow Let(p)$. Ta izraz lahko zapišemo tudi v obliki logičnega pravila $Potovanje(?p) : -Let(?p)$.

[1] $\mathcal{ALC}$ in $\mathcal{ALCUE}$ sta ekvivalentna jezika, saj je unijo $(\mathcal{U})$ in eksistenčno omejitev $(\mathcal{E})$ možno izraziti s pomočjo negacije $(\mathcal{C})$ [29]

Konstrukt	**Sintaksa**	**Jezik**			
Koncept	C	$\mathcal{FL}_0$	$\mathcal{FL}_-$	$\mathcal{AL}$	$\mathcal{S}$[1]
Imenska vloga	r				
Presek	$C \sqcap D$				
Vrednostna omejitev	$\forall r.C$				
Omejena eksistenčna kvantifikacija	$\exists r$				
Univerzalni koncept $\top$	$\top$				
Prazen koncept $\bot$	$\bot$				
Atomarna negacija	$\neg A$				
Negacija	$\neg C$	$\mathcal{C}$			
Unija	$C \sqcup D$	$\mathcal{U}$			
Eksistenčna omejitev	$\exists r.C$	$\mathcal{E}$			
Številčna omejitev	$(\geq nr)(\leq nr)$	$\mathcal{N}$			
Enumeracija	$a_1 ... a_n$	$\mathcal{O}$			
Hierarhija vlog	$r \sqsubseteq s$	$\mathcal{H}$			
Inverzna vloga	r^{-1}	$\mathcal{I}$			
Kvalificirana številčna omejitev	$(\geq nr.C)(\leq nr.C)$	$\mathcal{Q}$			

Tabela 2.1: Konstrukti opisne logike (prirejeno po [29])

Kljub temu, da lahko Hornove stavke pretvorimo v izjave logike prvega reda, njihov pomen ni identičen. Glavna razlika je, da je sklepanje v logiki prvega reda monotono, medtem ko je pri logičnem programiranju nemonotono. Zaradi te lastnosti logično programiranje združujejo z opisno logiko. Možna sta dva načina: (i) hibridni jeziki, ki združujejo logično programiranje z opisno logiko (npr. WSDL) in (ii) dodajanje Hornovih pravil na baze znanja, ki temeljijo na opisni logiki (npr. SWRL).

2.2.6 Ontologije

Ontologije so eden od načinov predstavitve znanja. Za formalen zapis ontologij se najpogosteje uporablja opisna logika, redkeje tudi F-logika. Včasih se uporablja opisna logika v kombinaciji s Hornovimi pravili. V akademskem svetu konsenz o splošni definiciji ontologije še ni bil dosežen. Najbolj sprejeta in tudi citirana je definicija Gruberja iz leta 1993: "*Ontologija je eksplicitna specifikacija*

[1] $\mathcal{S}$ vsebuje $\mathcal{ALC}$ jezik in tranzitivne vloge

konceptualizacije"[1] [35].

Na definicijo je bilo podanih precej komentarjev, predvsem glede izrazov specifikacija in konceptualizacija, saj naj ne bi bila dovolj pomensko polna. Borst je leta 1997 dopolnil Gruberjevo definicijo ontologije: "Ontologije so definirane kot formalna specifikacija skupne konceptualizacije[2]"[9].

Obe definiciji je v [77] združil in razložil Studer et al.: "Ontologija je formalna in eksplicitna specifikacija skupne konceptualizacije", kjer se naj posamezni termini naj interpretirajo kot:

- **konceptualizacija** se nanaša na abstraktni model nekega pojava v svetu. Abstraktni model je izražen na podlagi identificiranih konceptov tega pojava,
- **eksplicitna** pomeni, da so tipi uporabljenih konceptov in omejitve pri njihovi uporabi eksplicitno definirani (omejitev je npr., da se `Let` in `Potovanje ZVlakom` izključujeta),
- **formalna** se nanaša na dejstvo, da bi naj ontologija bila strojno berljiva, kar izključuje naravni jezik,
- **skupna** odraža stališče, da naj ontologija zajema splošno oz. skupno znanje, to je znanje, ki ni lastno posamezniku, temveč ga je sprejela določena skupina ljudi.

Ontologija ni zgolj računalniška predstavitev neke domene, temveč izraža neko stopnjo konsenza o znanju iz te domene [77]. Gomez et al. poudarjajo, da nek domenski model ni nujno ontologija zaradi tega, ker je predstavljen v ontološkem jeziku (npr. OWL ali WSML), iz enakih razlogov, kot ni nek program sistem znanja zgolj zaradi tega, ker je napisan v prologu [29].

Za naš primer iz domene poslovnih potovanj lahko tako rečemo, da je ta model ontologija takrat, ko ga bomo formalno zapisali in bo dosežen konsenz o znanju, zajetem v ontologiji, med uporabniki tega modela. Studer v [77] navaja, da naj bo konsenz odvisen od konteksta, npr. če razvijamo ontologijo o boleznih

[1] An ontology is an explicit specification of a conceptualization.
[2] Ontologies are defined as a formal specification of a shared conceptualisation.

za bolnišnico, mora biti dosežen konsenz med vsemi zdravniki te bolnišnice. Če pa razvijamo nacionalni bibliografski sistem, pa naj bo dosežen dosežen na nacionalni ravni - vsak naj bi ontologijo sprejel kot veljavno.

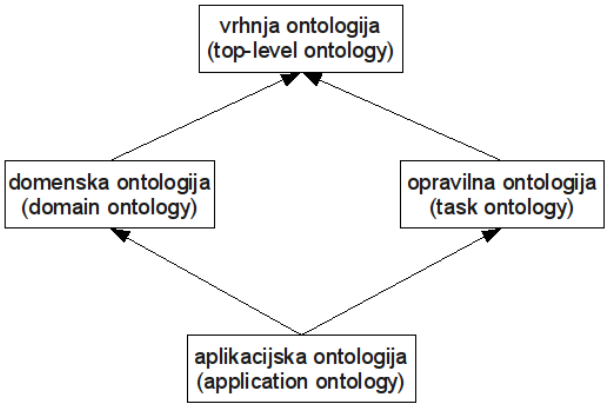
Najširše uporabljano kategorizacijo ontologij je podal Guarino [36]. V kategorizaciji, ki je prikazana na sliki 2.4, je ontologije razdelil na štiri vrste:

- **vrhnje ontologije** – opisujejo zelo splošne koncepte, kot so prostor, čas, stvari, objekti, dokodki,..., ki so neodvisni od posameznega problema ali domene,
- **domenske ontologije** – opisujejo besednjak glede na splošno domeno (potovanja, medicina, avtomobili,...), s specializacijo konceptov, definiranih v vrhnji ontologij,
- **opravilne ontologije** – so precej podobne domenskim ontologijam, z razliko, da opisujejo splošna opravila oz. aktivnosti (prodajanje, diagnosticiranje,...); koncepti opravilne ontologije so specializacija konceptov vrhnje ontologije,
- **aplikacijske ontologije** – združujejo koncepte, ki so odvisni od domene in opravil in so pogosto specializacija obeh povezanih ontologij. Ti koncepti se pogosto ujemajo z vlogami, v katerih nastopajo domenske entite ob opravljanju neke aktivnosti (poslovno potovanje, zamenljiva enota, nadomestni del,...).

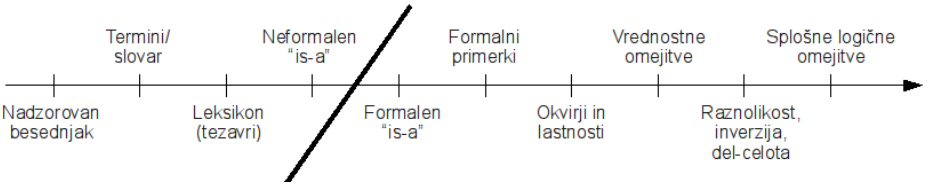
Naslednjo po našem mnenju pomembno klasifikacijo ontologij sta podala Lassila in McGuiness [51]. Avtorja sta se odločila kategorizirati ontologije zaradi različne uporabe izraza ontologija, ki ga veliko avtorjev "zlorablja". Ontologije sta glede na bogatost njihove interne strukture in konceptualizacije klasificirala kot linearen spekter. Omenjena kategorizacija je prikazana na sliki 2.5.

Slika prikazuje, da bolj ko se pomikamo po premici v desno, bolj bogata je ontologija. Točke na premici predstavljajo pogoste specifikacije ontologij, ki se pojavljajo v literaturi. Prečna črta razmejuje od, kod naprej je uporaba termina ontologija umestna.

Najosnovnejša oblika, ki sta jo avtorja odkrila pri uporabi izraza ontologija, je nadzorovan besednjak, ki predstavlja zgolj končen niz izrazov. Pomensko



Slika 2.4: Kategorizacija ontologij po Guarinu [36]



Slika 2.5: Kategorizacija ontologij po Lassilu in McGuinessu [51]

polnejši je slovar, ki predstavlja seznam terminov in njihov pomen. Takšna predstavitev ponuja več informacij, ki pa pogostno niso enoumne.

Leksikoni oz. tezavri dodajajo pomen v obliki relacij med termini. Tezavri ponujajo informacije o sopomenkah in tipično niso organizirani v hierarhijo.

Neformalna "is-a"hierarhija predstavlja neformalno generalizacijo, kar pomeni, da ni nujno, da je pripadnik specifičnega razreda hkrati tudi pripadnik generičnega (nadrejenega) razreda. V neformalni "is-a"hierarhiji so lahko recimo nadomestni deli podkategorija oz. podrazred avtomobilov, kljub temu, da nadomestni del ni hkrati tudi avtomobil.

Formalne "is-a"hierarhije vsebujejo striktne relacije "podrazred", kjer mora biti pripadnik specializiranega koncepta obvezno tudi pripadnik generalnega koncepta (podrazred koncepta avtomobil je lahko recimo koncept limuzina). Formalni primerki nadgrajujejo striktne "is-a"hierarhije tako, da striktno ločijo med koncepti in njihovimi primerki (razredi in objekti). Okvirji in lastnosti gredo korak dalje s specifikacijo lastnosti, ki jih lahko imajo koncepti in objekti.

Vrednostne omejitve podajajo "podatkovne tipe"posameznih lastnosti, ki so

lahko vrednosti. Mnogi jeziki omogočajo specifikacijo različnih vrst lastnosti (raznolikost, inverzija, del-celota,...) ali definicijo lastnosti in konceptov z matematičnimi ali logičnimi izrazi.

2.2.7 Jeziki za implementacijo ontologij

Jezike za implementacijo ontologij lahko delimo na dve kategoriji:

- tradicionalni jeziki in
- označevalni jeziki.

Dva od najbolj poznanih tradicionalnih jezikov sta KIF (Knowledge Interchange Format)[28], ki temelji na logiki prvega reda, ter njegova nadgradnja Ontolingua [25], ki je bil ustvarjen z namenom poenostaviti razvoj ontologij.

Drugi tradicionalni jeziki so še CycL [53], ki je prvi jezik za opis ontologij, ki združuje logiko okvirjev in logiko prvega reda, LOOM [58], ki prvotno ni bil namenjen za implementacijo ontologij, temveč za splošne baze znanja in temelji na opisni logiki v kombinaciji s produkcijskimi pravili. OCML [63] je bil razvit leta 1993 kot "operativna različica"Ontolingue. V letu 1995 je bil razvit FLogic, ki združuje logiko okvirjev in logiko prvega reda.

Razvoj svetovnega spleta je povzročil razvoj označevalnih jezikov za implementacijo ontologij. Prvi takšen jezik je bil SHOE (Simple HTML Ontology Extension)[56]. Obstajata dve različici jezika SHOE(HTML) in SHOE(XML). Eden od poskusov je bil tudi jezik XOL [44], ki pa ni dosegel večjih uspehov.

Danes najbolj razširjeni jeziki so RDF in RDFS, OWL ter WSML. Vsi temeljijo na XML jeziku in jih bomo podrobneje opisali v nadaljevanju. Preostalih jezikov ne bomo podrobneje opisovali, saj več niso tako aktualni, bralec pa lahko primerjalno analizo vseh omenjenih jezikov najde v [29].

2.2.7.1 RDF in RDFS

RDF [59] je jezik, ki temelji na XML in omogoča opisovanje spletnih virov na podlagi semantičnih mrež. Jezik je uradno priporočilo W3C organizacije in je postal de-facto standard za izmenjavo podatkov na semantičnem spletu. Jezik

ima zelo enostavno zgradbo. RDF dokument je sestavljen iz množice stavkov v obliki trojčkov. Vsak trojček je sestavljen iz subjekta, predikata in objekta, kjer sta subjekt in objekt koncepta, predikat pa relacija med tema konceptoma. Če združimo izjave, zapisane v RDF-u, dobimo semantično mrežo (usmerjen graf).

RDFS [10] oz. RDF Schema je jezik za definicijo strukture RDF dokumenta. S pomočjo RDFS lahko definiramo koncepte in relacije med njimi, jezik ločuje tudi med koncepti in primerki. Primeren je za gradnjo taksonomij, medtem ko za večje in pomensko bolj polne semantične mreže ni primeren.

2.2.7.2 OWL

OWL [61] (Web Ontology Language) je jezik za opis ontologij, ki temelji na RDF jeziku. Vsak OWL dokument je hkrati veljaven RDF dokument. Jezik je nastal je iz DAML (DARPA Agent Markup Language) in OIL (Ontology Interchange Language in Ontology Inference Layer) jezikov za opis ontologij. Prvotno se je kombinacija teh dveh jezikov imenovala DAML+OIL. OWL je nadaljevanje razvoja DAML+OIL jezika v okviru delovne skupine W3C Web Ontology. OWL je priporočilo W3C od februarja 2004 in je v tem času postal prevladujoč jezik za razvoj ontologij.

OWL 1 je prvotna različica OWL specifikacije in ima definirane 3 profile:

- OWL Lite
- OWL DL
- OWL Full

Najbolj uporabljana različica jezika je OWL DL, kjer DL označuje opisno logiko. Temelji na $\mathcal{SHOIN}$ opisni logiki (glej tabelo 2.1). OWL DL je rešljiv jezik, kar pomeni, da odgovor zmeraj dobimo v končnem času. OWL Lite predstavlja okrnjeno različico OWL DL jezika (uporaba kardinalnosti je omejena samo na 0 in 1, za specifikacijo konceptov ni mogoče uporabiti unije in preseka,...). V praksi OWL Lite ni pogosto uporabljan.

OWL Full omogoča določene funkcionalnosti, ki v okviru opisne logike niso možne. Najpomembnejša razlika je zmožnost meta-modeliranja. OWL Full

omogoča, da so primerki razredov (konceptov) lahko hkrati primerki in razredi. OWL Full ni rešljiv in ima slabo podporo pogonov sklepanja.

Z opisno logiko v določenih primerih ni mogoče celovito predstaviti znanja neke domene, zaradi tega so OWL DL razširili s podjezikom RuleML v jezik, imenovan SWRL [42] (Semantic Web Rule Language). SWRL tako omogoča predstavitev znanja na podlagi opisne logike ter z uporabo Hornovih pravil.

OWL 2 specifikacija je izšla leta 2009 kot razširitev prvotne specifikacije, izdane leta 2004. Osnovni konstrukti jezika so ostali enaki, prav tako je OWL 2 združljiv za nazaj. Vsaka veljavna OWL 1 ontologija je hkrati tudi veljavna OWL 2 ontologija. Ključna razlika je v profilih. OWL 1 definira tri profile: OWL Lite, OWL DL in OWL Full. Izkazalo se je, da v praksi OWL Lite in OWL Full nista bila pogosto uporabljana, saj je OWL Lite preveč omejen, medtem ko za OWL Full ne obstajajo zmogljivi pogoni sklepanja.

OWL 2 uvaja profile, vsak od njih prinaša določene prednosti za specifične scenarije oz. aplikacije. Definirani so trije profili: OWL 2 EL, OWL 2 QL in OWL 2 RL. Vsak izmed profilov predstavlja sintaktično omejitev OWL 2 specifikacije. Profili nudijo različno izraznost, vendar so vsi manj izrazni kot OWL DL [41].

OWL 2 EL profil zahteva pogone sklepanja, ki imajo polinomsko časovno zahtevnost. Profil je primeren za aplikacije, ki zahtevajo velike ontologije, kjer je bolj kot hitrost izvajanja povpraševanj pomembna izrazna moč. OWL 2 QL omogoča izvajanje konjunktivnih povpraševanj v logaritemski časovni zahtevnosti. Posebej je primeren za ontologije, ki imajo sorazmerno majhen terminološki okvir ter velik izrazni okvir. OWL 2 RL omogoča sklepanje v polinomskem času algoritmom sklepanja, ki so zasnovani na pravilih.

OWL 2 definira dve semantiki, ki določata pomen OWL 2 ontologij, in sicer neposredno semantiko in RDF-osnovano semantiko. Na osnovi semantike intepretiramo OWL 2 ontologije ter izvajamo proces sklepanja. Neposredna semantika dodaja pomen neposredno ontološki strukturi in je kompatibilna s $\mathcal{SROIQ}$ opisno logiko. Neposredna semantika je zelo blizu teoretičnim modelom opisne logike. Na RDF osnovana semantika podaja pomen neposredno RDF grafom. Vsako OWL 2 ontologijo lahko interpretiramo z RDF-osnovano semantiko. Neposredna semantika prinaša določene omejitve, zaradi tega je potrebno

določene ontologije transformirati.

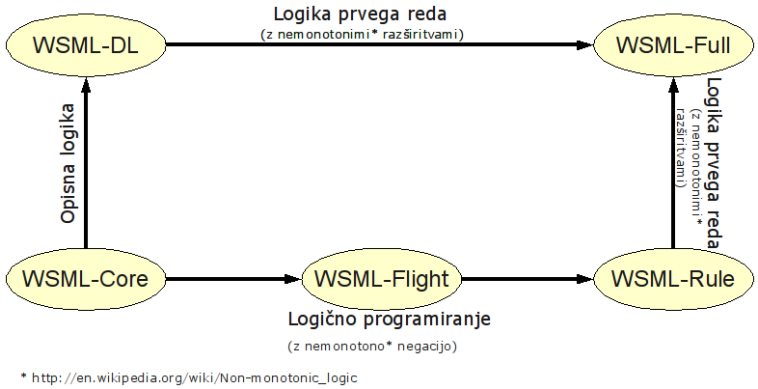
Neformalno označujemo ontologije, ki uporabljajo neposredno semantiko, z OWL 2 DL, medtem ko označujemo ontologije, ki uporabljajo RDF-osnovano semantiko, z OWL 2 Full. OWL 2 DL in OWL 2 Full tako označujeta katero semantiko bomo uporabili za interpretacijo, ter ju zaradi tega ne moremo enačiti s profili OWL DL in OWL Full. Profile, ki jih definira OWL 1 (OWL Lite, OWL DL in OWL Full) v celoti nadomeščajo profili OWL 2 (OWL 2 EL, OWL 2 QL, OWL 2 RL ter krovni neomejeni OWL 2).

2.2.7.3 WSML

WSML [11] je jezik, ki je nastal v okviru WSMO (Web Service Modeling Ontology). Avtorji WSML so si pri razvoju jezika zadali cilj raziskati različne formalizme (pretežno opisna logika in logično programiranje) v kontekstu ontologij in spletnih storitev. Zaradi tega cilja so se odločili razviti svoj jezik, ne pa uporabiti kakšnega obstoječega (npr. OWL) [11].

WSML se od obstoječih jezikov, ki služijo za opisovanje ontologij, v dveh ključnih stvareh, in sicer sta to: (i) uporaba dveh različnih paradigem jezikov logike (opisna logika in logično programiranje) v enotnem sintaktičnem ogrodju in (ii) abstrakcija sintaktične kompleksnosti na dve ravni – konceptualno in logično. Konceptualno modeliranje je namenjeno nezahtevnim uporabnikom, ki niso vajeni logičnega programiranja, medtem, ko je logična raven namenjena strokovnjakom in omogoča kompleksne logične definicije. Avtorji želijo na ta način doseči enostavnost uporabe in povečano hitrost učenja uporabe jezika z uporabo konceptualne ravni, hkrati pa z logično ravnjo omogočajo napredno uporabo.

Podobno kot OWL je tudi WSML razdeljen na več nivojev, glede na željeno izraznost jezika. Posebnost WSML jezika je, da ima izraznost jezika razvejeno v dve smeri: (i) razširitve z uporabo konceptov opisne logike in (ii) razširitve z uporabo konceptov logičnega programiranja. Ti dve veji, ki temeljita na različnih paradigmah jezikov logike, med seboj nista združljivi, imata pa dve stični točki: (1) WSML-Core je najmanj izrazen jezik in predstavlja presek izraznosti oz. zmožnosti jezikov po obeh platformah in (2) WSML-Full, ki je izrazno najbolj



Slika 2.6: Različice WSML jezika (prirejeno po [11])

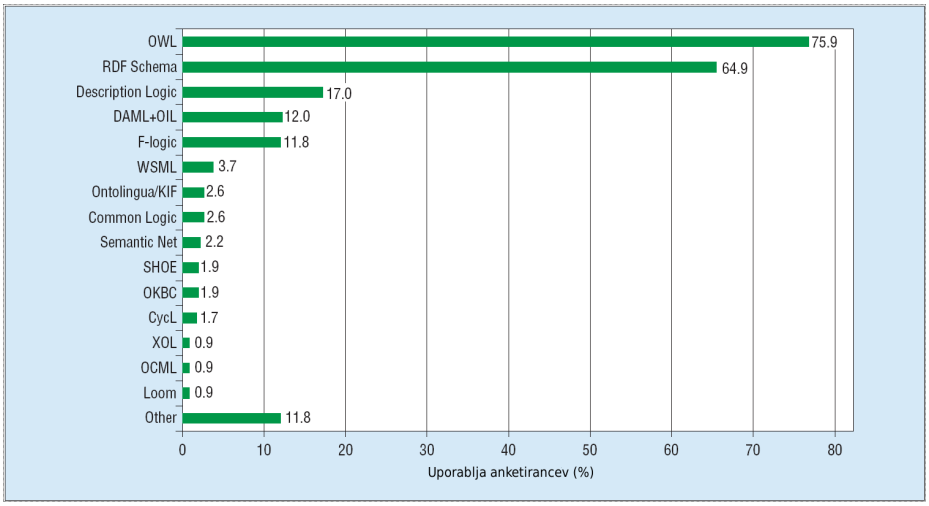
močen in vsebuje konstrukte obeh paradigem – pri tem je potrebno opomniti, da WSML-Full še ni v celoti definiran [11]. Različice WSML jezika so prikazane na sliki 2.6.

Ključne funkcionalnosti WSML jezika, ki jih avtorji izpostavljajo, so: (1) enotno sintaktično okolje za množico večplastnih jezikov – opisana večplastnost in uporaba različnih paradigem jezikov logike, (2) normativna, človeško berljiva sintaksa – sintaksa je človeško berljiva, hkrati WSML definira tudi XML in RDF sintakso, za strojno izmenjavo, (3) delitev na konceptualno in logično raven modeliranja, (4) semantika je osnovana na dobro poznanih formalizmih – uporablja logične formalizme, kot so datalog, opisna logika, logično programiranje, (5) jezik za splet – WSML gradi na podobnih standardih kot tehnologije semantičnega spleta, in sicer IRI, XQuery in XML, omogoča pa tudi serializacijo v RDF z uporabo RDF in RDFS konstruktov. WSML ni kompatibilen z OWL.

2.2.7.4 Razširjenost uporabe ontoloških jezikov in orodij

Jorge Cardoso je v letu 2007 objavil rezultate zanimive raziskave [16], kjer je anketiral 627 raziskovalcev in strokovnjakov s področja semantičnega spleta. Glavni cilj raziskave je bil ugotoviti razširjenost uporabe ontoloških jezikov in orodij za razvoj ontologij ter uporabo metodologij za razvoj ontologij. V tem razdelku podajamo izsledke o uporabi jezikov in orodij.

Na sliki 2.7 je prikazana uporaba jezikov za implementacijo ontologij. Iz slike je možno razbrati, da sta OWL in RDF Schema po deležu uporabe daleč



Slika 2.7: Uporaba jezikov za implementacijo ontologij [16]

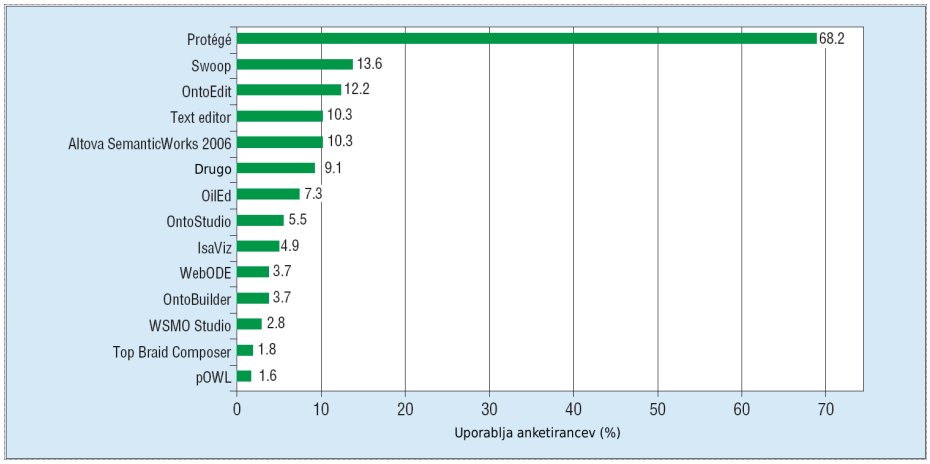
v ospredju. Presenetljiva je ugotovitev, da DAML+OIL, katerega naslednik je OWL, še zmeraj uporablja 12 % anketirancev. Tudi uporaba WSML jezika je presenetljiva zaradi njegove relativne mladosti presenetljiva (raziskava narejene med januarjem 2006 in januarjem 2007, WSML pa je bil objavljen komaj junija 2005).

Slika 2.8 prikazuje delež anketirancev, ki uporabljajo posamezno orodje za razvoj ontologij. Iz slike lahko razberemo, da je daleč v ospredju Protégé, ki je odprtokodno orodje za razvoj ontologij. Prvotno je bil uporabljen kot meta orodje za razvoj sistemov znanja, in sicer že leta 1987. Presenetljiva je ugotovitev, da 10 % anketirancev za razvoj ontologij uporablja preproste urejevalnike besedil (npr. beležnica).

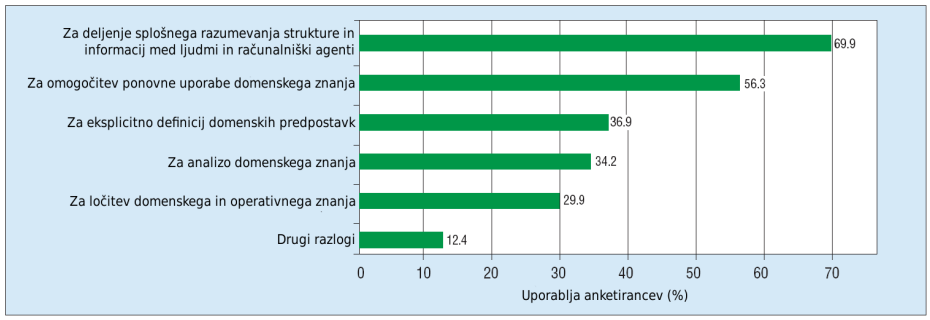
Na sliki 2.9 je prikazan namen uporabe ontologij v računalniških sistemih. Iz slike je razvidno, da sta glavna motivatorja za uporabo ontologij (i) deljenje splošnega razumevanja strukture podatkov in informacij med ljudmi in računalniškimi agenti ter (ii) omogočitev ponovne uporabe domenskega znanja.

2.2.8 Semantični splet

Ideja semantičnega spleta se je pojavila konec 90-ih let. Osnovna zamisel je bila predstavitev podatkov na svetovnem spletu v obliki, ki je razumljiva strojem,



Slika 2.8: Uporaba orodij za razvoj ontologij [16]



Slika 2.9: Razlogi za razvoj ontologij [16]

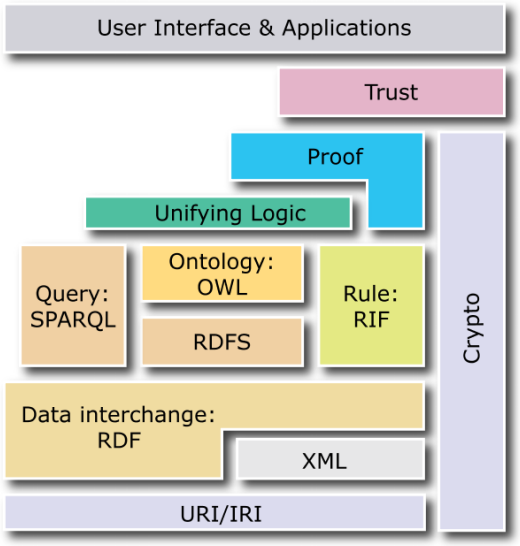
tako da bi le-ti lahko avtonomno pridobili podatke, jih združili ter jih posredovali uporabniku. Avtorji prvotne ideje govorijo o t.i. spletu podatkov (ang. *web of data*), kjer nimamo več množice med seboj povezanih dokumentov (spletnih strani), temveč množico med seboj povezanih podatkov. Podatki na spletu podatkov bi naj bili označeni z metapodatki, zaradi česar bi imeli stroji zavedanje o pomenu podatkov in bi jih lahko avtonomno procesirali ter izmenjevali med seboj [5].

Prvotna ideja je pridobila precej podpornikov in izoblikovala se je zamisel semantičnega spleta (ang. *semantic web*). Semantični splet je še zmeraj ideja oz. vizija, po kateri podatkom na svetovnem spletu določimo pomen na osnovi metapodatkov. Na osnovi vizije semantičnega spleta je nastala vrsta tehnologij, ki omogočajo specifikacijo pomena podatkov ter sklepanje nad tako označenimi podatki. Te tehnologije imenujemo tehnologije semantičnega spleta in so po navadi predstavljene z večplastnim diagramom (slika 2.10).

Tehnologije semantičnega spleta temeljijo na XML in URI/IRI tehnologijah. Jezik za izmenjavo podatkov je RDF, ki smo ga že opisali. Eden izmed ključnih gradnikov semantičnega spleta so ontologije, ki so podane bodisi v RDFS bodisi v OWL jeziku. Jezik za izmenjavo pravil RIF omogoča specifikacijo pravil. Pravila vzpostavljajo dodaten nivo sklepanja, vzporedno k ontologijam. Za namene povpraševanja po tako organiziranem znanju je nastal povpraševalni jezik sparql. Omenjene tehnologije so nastale v okviru semantičnega spleta in so trenutno najbolj razširjene tehnologije za predstavitev znanja.

2.2.8.1 Sparql

Sparql je jezik za povpraševanje po RDF grafih. Sintaktično je podoben SQL jeziku, vendar obstajajo odstopanja zaradi prilagoditve povpraševanju po grafih. Sparql specifikacija [66] definira povpraševalni jezik kot tudi protokol za izvajanje povpraševanj. Posamezna podatkovna shramba lahko na osnovi te specifikacije izpostavi sparql končno točko, ki omogoča izvajanje sparql povpraševanj po sparql protokolu. Izpis 2.2 prikazuje strukturo sparql povpraševanja. Osnovni elementi vsakega sparql povpraševanja so:



Slika 2.10: Diagram tehnologij semantičnega spleta

- **Deklaracija predpon** določa predpone URI naslovov, ki bodo uporabljeni v vzorcu povpraševanja. Ta del je opcijski in je namenjen poenostavitvi povpraševanj. Uporaba predpon omogoča krajšanje URI naslovov v vzorcu povpraševanja.

- **Deklaracija podatkovnih množic** določa od kod naj povpraševanje pridobi podatke. Kot deklaracija podatkovnih množic je lahko podan naslov sparql podatkovne točke ali naslov RDF datotek, ki vsebujejo podatke, na katerih želimo izvesti povpraševanje.

- **Tip povpraševanja** določa obliko rezultatov, ki jih bo povpraševanje vrnilo. Nabor tipov je *SELECT, CONSTRUCT, ASK, DESCRIBE*. Najbolj uporabljana oblika je *SELECT*, ki vrača rezultate v podobni obliki kot SQL povpraševanja.

- **Vzorec povpraševanja** določa vzorce, ki jih naj sistem išče v grafu. Na ta način natančno določa podatke, ki jih želimo pridobiti s povpraševanjem.

- **Modifikatorji povpraševanja** omogočajo preoblikovanje rezultatov povpraševanja (omejitev rezultatov, razvrstitev, preoblikovanje...).

```
# deklaracija predpon
PREFIX foo: <http://example.com/resources/>
...
# deklaracija podatkovnih mnozic
FROM ...
# tip povprasevanja
SELECT ...
# vzorec povprasevanja
WHERE {
    ...
}
# modifikatorji povprasevanja
ORDER BY ...
```

Izpis 2.2: Struktura SPARQL povpraševanja

Povpraševanje po RDF grafih se konceptualno razlikuje od povpraševanja po relacijskih bazah. V relacijskih bazah so podatki združeni v tabele, ki so med seboj povezane. Ko izvajamo povpraševanje po relacijski bazi določimo tabele iz katerih želimo pridobiti podatke. Opcijsko lahko dodamo pogoje, ki omejijo rezultat na določene vrednosti atributov. RDF model ne organizira podatkov v skupine, temveč so vsi podatki shranjeni v enotnem grafu. Zaradi tega se sparql povpraševanje izvaja nad celotno podatkovno množico.

Sparql povpraševanje omejimo tako, da podamo vzorce, ki jih naj sistem poišče v RDF grafih, ki so podani z deklaracijo podatkovnih množic. RDF model je sestavljen iz množice izjav v obliki trojčkov, ki imajo naslednjo strukturo: *(subjekt, predikat, objekt)*. Subjekt določa vozlišče grafa, za katero je podana izjava. Predikat določa lastnost, ki jo želimo podati za določen subjekt, objekt pa določa vrednost lastnosti. Izjavo "nebo je modre barve"lahko predstavimo z naslednjim trojčkom *("nebo", "ima barvo", "modra")*. Trojčki, ki so osnovni element RDF modela, so tudi osnovni element sparql vzorca povpraševanj.

Vzorci povpraševanj so edini način za selekcijo podatkov v sparql povpraševanjih. Vsak vzorec povpraševanja je sestavljen iz množice trojčkov. Posamezne elemente trojčka (subjekt, predikat, objekt) v vzorcu povpraševanja pa lahko nadomestimo s spremenljivko. V tem primeru pogon za povpraševanje v grafu

poišče vrednosti, ki so eksplicitno podane, spremenljivkam pa dodeli vrednosti na podlagi podatkov v RDF grafu.

V izpisu 2.3 je podano povpraševanje, ki v RDF grafu poišče vse osebe ter izpiše njihova imena in naslov elektronske pošte. Vzorec povpraševanja vsebuje tri spremenljivke, *?x, ?ime* in *?eposta*. Na osnovi tega vzorca pogon povpraševanja poišče v grafu vsa vozlišča, ki imajo definirana predikata *foaf:name* in *foaf:mbox*. Za vsako tako vozlišče sistem priredi vrednosti teh predikatov spremenljivkama *?ime* in *?eposta*.

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

Izpis 2.3: Primer SPARQL povpraševanja

2.2.8.2 Sparql algebra

Sparql specifikacija [66] natančno specificira procesa izvajanja povpraševanj. Le-ta ni obvezujoč, vendar je kljub temu splošno sprejet med odprtokodnimi sparql pogoni. Specifikacija predvideva transformacijo vhodnega povpraševanja v abstraktno sparql povpraševanje (SAQ), ki je definirano kot nterica $\langle E, DS, R\rangle$, kjer je E izraz podan v sparql algebri, DS je RDF podatkovna množica, R pa je tip povpraševanja (*SELECT, CONSTRUCT, ASK, DESCRIBE*).

Specifikacija [66] podaja algoritme za pretvorbo vhodnega povpraševanja, podanega v sparql sintaksi, v povpraševanje, ki je izraženo v sparql algebri. Sparql algebro sestavljajo operatorji, ki jih delimo na vzorce grafa in modifikatorje rešitve (tabela 2.2). Operatorji tipa vzorci grafa se uporabljajo za pridobivanje rešitev, medtem ko operatorji modifikatorji rešitve zgolj preoblikujejo obstoječe rešitve.

Povpraševanje v sparql poteka tako, da pogon povpraševanja v RDF grafu išče vzorce, ki so podani s povpraševanjem. Podatki, ki jih želi uporabnik pridobiti s povpraševanjem, so podani s spremenljivkami v vzorcu povpraševanja. Skladno s tem, rešitve v sparql algebri predstavljajo množico preslikav med

spremenljivkami in dejanskimi podatki iz grafa (imenujmo to množico, množica vzorcev rešitev). Operatorji tipa vzorci grafa se tako uporabljajo za pridobivanje množice vzorcev rešitev, operatorji modifikatorji rešitve pa preoblikujejo množico vzorcev rešitev.

Vzorci grafa	Modifikatorji rešitve
bgp	toList
join	orderBy
leftJoin	project
filter	distinct
union	reduced
graph	slice
triple	

Tabela 2.2: Operatorji sparql algebre

V sparql algebri so definirani naslednji operatorji:

- **bgp** (ang. basic graph pattern - osnovni vzorec povpraševanja) je operator za določanje vzorca povpraševanja. Operator *bgp* dejansko predstavlja množijo izjav v obliki trojčkov. Skladno s tem je vsaka operacija *bgp* sestavljena iz množice operacij *triple*,
- **join** se uporablja za zlivanje dveh vzorcev rešitve. Zlivanje poteka na osnovi preseka spremenljivk obeh vzorcev rešitev. Operator je semantično enakovreden operatorju *JOIN* v SQL,
- **leftJoin** se uporablja za pripojitev enega vzorca rešitve k drugemu. Pripojitev poteka na osnovi preseka spremenljivk obeh vzorcev rešitev, lahko pa je tudi podana z izrazom. Operator je semantično enakovreden operatorju *LEFT JOIN* v SQL,
- **filter** se uporablja za omejevanje vzorcev rešitev tako, da izraz podan v operaciji *filter* določa vzorec ali vrednost posameznih spremenljivk v vzorcu rešitev. Operator je semantično ekvivalenten operatorju *WHERE* v SQL,
- **union** se uporablja za združevanje dveh množic vzorca rešitve. Izvede se unija množic vzorcev rešitev,

- **graph** se uporablja za specifikacijo grafa, iz katerega naj poteka pridobivanje množice vzorcev rešitev,
- **triple** se uporablja za specifikacijo trojčka, ki je sestavni del operacije *bgp*,
- **toList** pretvori množico vzorca rešitev v seznam, ki je primeren za izpis,
- **orderBy** razvrsti podatke v vzorcu rešitve glede na določeno spremenljivko,
- **project** iz množice vzorcev rešitev izbere podatke za samo izbrane spremenljivke,
- **distinct** v množici vzorcev rešitev izloči podvojene podatke,
- **reduced** ima podobno vlogo kot operator *distinct*. Razlika je, da operator *distinct* odstrani iz množice vzorcev rešitev vse podvojene rešitve, medtem ko operator *reduced* zgolj dovoli odstranitev podvojitev. Pri uporabi operatorja *reduced* lahko pogoni povpraševanja podvojene rešitve odstranijo ali pa tudi ne,
- **slice** množico vzorcev rešitev razreže glede na podane argumente (npr. prvih sto rešitev).

V izpisu 2.4 je podano zgornje povpraševanje zapisano v sparql algebri. V okviru doktorske disertacije smo razširili sparql algebro tako, da omogoča pridobivanje podatkov (torej množico vzorcev rešitev) iz spletnih storitev. Razširitev sparql algebre omogoča proženje klasičnih spletnih storitev neposredno iz sparql povpraševanj. Razvili smo tudi algoritme, ki na osnovi semantičnih opisov storitev transformirajo povpraševanja zapisana v sparql algebri tako, da poteka pridobivanje in integracija podatkov iz spletnih storitev povsem samodejno.

```
(project (?ime ?eposta)
  (bgp
    (triple ?x foaf:name ?ime)
    (triple ?x foaf:mbox ?eposta)
  ))
```

Izpis 2.4: Primer povpraševanja v sparql algebri

2.3 Semantično označevanje storitev

2.3.1 Semantične spletne storitve (SWS)

Spletne storitve vzpostavljajo standardizirane vmesnike za izmenjavo podatkov med aplikacijami [1]. V ta namen je nastala množica specifikacij, ki omogočajo izmenjavo podatkov nedovisno od platforme ter programskega jezika. Obstajata dve osnovni paradigmi: SOAP in REST storitve. Osnovna gradnika SOAP storitev sta WSDL jezik [17] (jezik za opis storitev) ter SOAP protokol (protokol za izmenjavo sporočil). Izmenjava sporočil med storitvami poteka na osnovi XML jezika. REST storitve predstavljajo manj kompleksno alternativo SOAP storitvam. Pri REST storitvah poteka izmenjava podatkov na osnovi standardnega HTTP protokola [3]. Jezik za izmenjavo sporočil ni določen, pogosto se uporabljata JSON ali XML.

Spletne storitve so široko sprejete in izredno učinkovite pri komunikaciji med aplikacijami. Ključna pomanjkljivost spletnih storitev (tako SOAP kot REST) je, da so opisi storitev zgolj na sintaktični ravni. To pomeni, da opisi storitev definirajo zgolj podatke za proženje storitev ter strukturo vhodnih in izhodnih sporočil. Semantične spletne storitve združujejo koncepte semantičnega spleta ter spletnih storitev, tako da storitvam dodajajo semantične opise, ki omogočajo avtomatizacijo odkrivanja storitev, kompozicije ter proženja [65].

V kolikor želimo uporabiti klasične spletne storitve, je te potrebno poiskati ter razviti odjemalce, ki bodo prožili posamezno storitev ali kompozicijo več storitev. Semantični opisi omogočajo avtomatizacijo tega procesa, tako da lahko računalniki oz. inteligentni agentje avtonomno poiščejo ustrezne storitve, izvedejo kompozicijo ter jih tudi prožijo [30].

Grosof loči dve interpretaciji spletnih storitev. Interpretiramo jih lahko kot storitve semantičnega spleta ali kot semantične spletne storitve [34]. Pri prvi interpretaciji je osrednja tehnologija semantični splet ter integracija informacij, storitve služijo zgolj kot tehnologija za pridobivanje podatkov. Druga interpretacija postavlja v ospredje storitve, kjer je semantični splet zgolj podporna tehnologija za avtomatizacijo odkrivanja, kompozicije in proženja storitev.

Osrednji pristopi, ki realizirajo idejo semantičnih spletnih storitev, so WSMO,

OWL-S in SAWSDL. WSMO in OWL-S sodita v drugo interpretacijo koncepta (interpretacija semantičnih spletnih storitev), medtem ko SAWSDL ni mogoče uvrstiti ne v eno ne v drugo kategorijo, saj ne definira načina uporabe. WSMO in OWL-S izvirata bolj iz konceptov semantičnega spleta in večinoma izpostavljata problematiko, na kakšen način semantično opisati storitve, medtem ko SAWSDL pristop izhaja iz tehnologij spletnih storitev in se ukvarja s problematiko, kako povezati opise spletnih storitev s koncepti v neki ontologiji.

Kljub različnim izhodiščem imajo vse tri tehnologije skupen cilj:

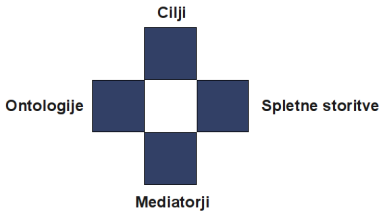
- enostavnejše in boljše odkrivanje storitev,
- avtomatizirana kompozicija,
- avtomatizirano proženje storitev.

2.3.2 WSMO

WSMO (Web Service Modeling Ontology) je meta-model za modeliranje spletnih storitev, ki omogoča avtomatizacijo odkrivanja, kompozicije ter proženja spletnih storitev [52]. WSMO strogo ločuje izraza storitev ter spletna storitev. Izraz storitev označuje storitve, ki predstavljajo neko vrednost ali storitev v realnem življenju. Spletna storitev pa je zaključena računska entiteta, ki s proženjem zadovolji cilje uporabnika [68]. WSMO ponuja sredstva za opis spletnih storitev, ki ponujajo dostop do storitev v realnem življenju (npr. nakupovanje, iskanje,...).

V okviru WSMO je definiran tudi lasten jezik za implementacijo ontologij WSML (Web Services Modeling Language) [11]. Referenčna implementacija WSMX omgoča avtomatizirano odkrivanje, proženje in kompozicijo storitev, opisanih v WSML jeziku [39]. WSMO sestavljajo štirje osnovni elementi (slika 2.11)[69]:

- **Ontologije** služijo za specifikacijo globalne sheme ter kot stična točka med heterogenimi podatkovnimi modeli. WSMO ne predpisuje, kateri jezik je potrebno uporabiti za predstavitev ontologije, ampak zgolj predpisuje zmožnosti, ki jih mora takšen jezik podpirati. Referenčni jezik za predstavitev ontologij v sklopu WSMO je WSML.



Slika 2.11: Glavni WSMO elementi

- **Spletne storitve** vsebujejo množico opisov spletnih storitev, ki opisujejo vmesnik spletne storitve, modelirajo njeno notranje delovanje ter podajajo njene zmožnosti.

- **Cilji** služijo za odkrivanje spletnih storitev. Uporabnik poda svoja pričakovanja/cilje skladno z ontologijo, sistem pa poišče storitve, ki izpolnjujejo te cilje.

- **Mediatorji** omogočajo odpravo heterogenosti med ontologijami, cilji ter spletnimi storitvami.

2.3.2.1 WSMX

WSMX [39] je referenčna implementacija izvajalnega okolja WSMO metamodela, ki uporablja WSML jezik. Služi kot register storitev in izvajalno okolje hkrati. Trenutna implementacija omogoča registracijo, odkrivanje in proženje storitev ter mediacijo med ontologijami, ne podpira pa drugih vrst mediacij in kompozicije storitev. Izvajalno okolje omogoča odkrivanje storitev na osnovi podanih ciljev. V kolikor je potrebno, izvajalno okolje opravi mediacijo med ontologijami. Sistem nato avtonomno opravi proženje izbrane spletne storitve, opravi potrebno mediacijo in vrne rezultat uporabniku.

Obstaja še ena implementacija izvajalnega okolja za WSMO, imenuje se IRS vendar avtorji v [39] opozarjajo na pomembno pomanjkljivost, to je nezmožnost avtomatiziranega odkrivanja v času izvajanja. V IRS je v času načrtovanja potrebno definirati mediatorje med cilji uporabnika in zmožnostmi storitev.

2.3.2.2 Modelirno okolje

Avtorji WSMO ponujajo tudi nabor orodij, ki omogočajo modeliranje in objavo storitev. Obstajata dve orodji, ki ponujata celovit sklop funkcionalnosti za delo s storitvami na podlagi WSMO, to sta (1) WSMO studio [22] in (2) WSMT – Web Service Modeling Toolkit [45]. Obe orodji temeljita na Eclipse razvojnem okolju in sta dostopni kot dodatka za to okolje.

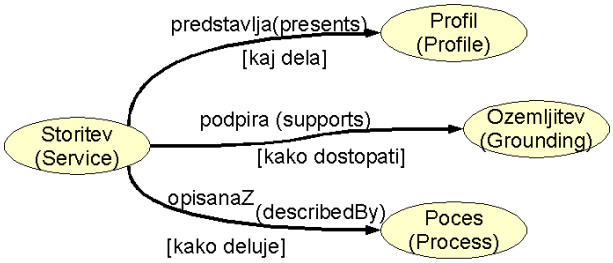
2.3.3 OWL-S

OWL-S [60] je ontologija za opisovanje semantičnih spletnih storitev, zapisana v OWL jeziku. Opisovanje storitev poteka prav tako v OWL jeziku, čeprav avtorji za podajanje kompleksnejših izjav (predvsem za specifikacijo predpogojev in rezultatov storitev) dopuščajo možnost uporabe SWRL, KIF ali DRS jezika [52].

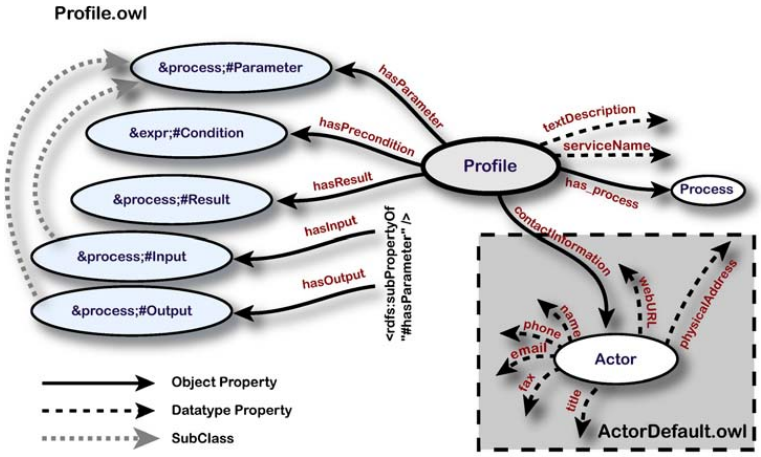
Namen OWL-S in WSMO pristopov je podoben. Oba pristopa podajata semantične opise storitev z namenom avtomatizacije odkrivanja, kompozicije ter proženja storitev. Slika 2.12 prikazuje štiri osnovne koncepte, ki sestavljajo OWL-S ontologijo. Ti so:

- **storitev** je vrhnji koncept OWL-S ontologije. Primerki tega koncepta so storitve, za katere podajamo opise,
- **profil** podaja funkcionalne in nefukncionalne opise storitve na višjem nivoju. Na podlagi profila storitve poteka proces odkrivanja storitev,
- **proces** opisuje notranje delovanje storitve,
- **ozemljitev** podaja povezavo med semantičnim opisom storitve v OWL-S ter konkretno storitvijo. Ozemljitev specificira podatke, ki omogočajo proženje storitve.

Odkrivanje storitev v OWL-S poteka tako, da uporabnik definira profil storitve, ki jo želi prožiti. Profil, ki ga je podal uporabnik, predstavlja cilj uporabnika. Izvajalno okolje primerja cilje uporabnika s profili storitev, ki se nahajajo v repozitoriju. OWL-S definira dva nivoja jezikov, enega za izmenjavo med računalniki, ta temelji na RDF/XML sintaksi, in enega v človeško berljivi sintaksi.



Slika 2.12: Osnovni koncepti OWL-S (prirejeno po [60])



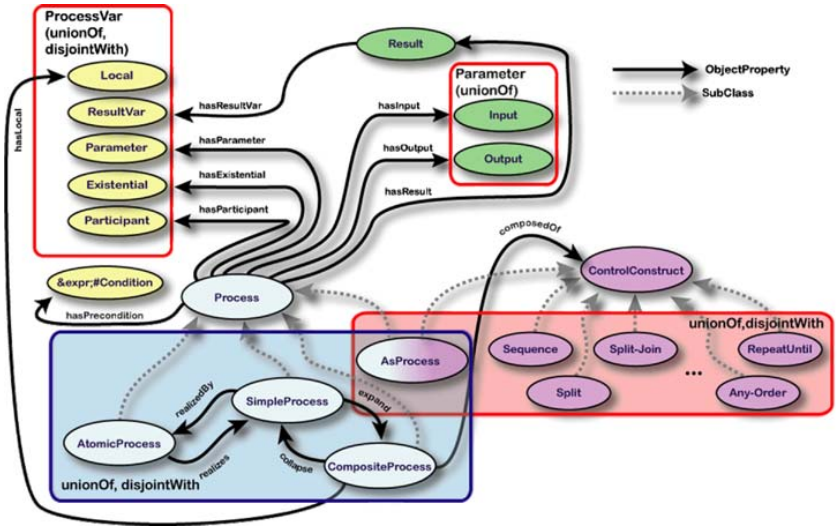
Slika 2.13: Podontologija profila storitve v OWL-S (vir: [60])

2.3.3.1 Profil storitve

Profil storitve je množica konceptov za opis zmožnosti storitve, katerih cilj je omogočitev odkrivanja storitev na podlagi zmožnosti storitev. Profil storitve sestavljajo funkcionalni in nefunkcionalni opisi storitve. Funkcionalni opis storitve predstavlja opis vhodov in izhodov iz storitve (predstavljeni so s koncepti v ontologiji) ter opis predpogojev in rezultatov storitve (logični izrazi). Nefunkcionalni opisi vsebujejo tekstovni opis storitve, ime storitve in kontaktne informacije storitve. Ontologija profila storitve je prikazana na sliki 2.13.

2.3.3.2 Proces storitve

Ontologija proces definira delovanje posamezne storitve ter podaja možen vzorec interakcije s storitvijo. OWL-S loči med atomarnim in kompozitnim procesom.



Slika 2.14: Podontologija procesa v OWL-S (vir: [60])

Atomarni proces nima interne strukture in gre za enostavno proženje spletne storitve – enkratno izmenjavo vhodov in izhodov. Pri kompozitnem procesu gre za kompozicijo storitev, kjer v procesu načrtujemo proženje več storitev, gre za t.i. orkestracijo storitev. Slika 2.14 prikazuje pomembnejše koncepte ontologije proces.

2.3.3.3 Ozemljitev

Ozemljitev storitve določa potek izmenjave abstraktnih informacij, kot jih definira atomarni proces. Ozemljitev ni vezana na konkretno tehnologijo, kljub temu pa je najbolj uporabljan pristop ozemljitve s SOAP spletnimi storitvami. V takšnem primeru ustvarimo pare povezav med atomarnim procesom in operacijo spletne storitve na eni strani ter med vhodi in izhodi v OWL-S ter sporočili v WSDL na drugi strani. Na podlagi ozemljitve poteka transformacija vhodnih in izhodnih podatkov ter proženje spletne storitve.

2.3.3.4 Orodja za delo z OWL-S storitvami

Za delo z OWL-S ne obstaja celovito okolje za razvoj storitev, vendar pa obstaja množica orodij različnih avtorjev, ki se dopolnjujejo v funkcionalnosti. Orodja za

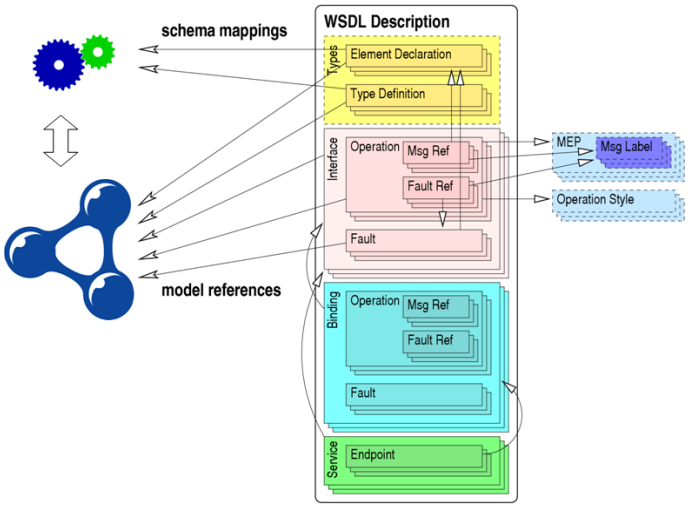
delo z OWL-S na žalost ne dosegajo kakovosti orodij WSMO [75]. Sami smo opazili, da so pogosto slabo vzdrževana in dogodi se lahko, da niso več kompatibilna z okolji od katerih so odvisna (npr. OWL-S editor, ki je namenjen modeliranju storitev in je realiziran kot dodatek za okolje Protégé, je kompatibilen zgolj z Protégé različice 3.2, ki pa ni več dosegljiva; zadnja stabilna različica je 3.3.1, v delu pa je že 4.0).

Najbolj uporabna orodja za delo z OWL-S so: (1) OWL-S editor – modeliranje storitev, dodatek za Protégé, (2) OWL-S editor – samostojno orodje za razvoj, validacijo in vizualizacijo storitev, (3) OWL-S razvojno okolje (CODE) – dodatek za modeliranje in objavo storitev v UDDI registru za razvojno okolje Eclipse, (4) WSDL2OWL-S – generiranje OWL-S opisa storitev iz WSDL dokumenta, (5) OWL-S API – programske knjižnjice za delo z OWL-S dokumenti, (6) OWL-S VM – izvajalno okolje za OWL-S storitve, glavna funkcionalnost je proženje storitev.

2.3.4 SAWSDL

Do sedaj smo obravnavali WSMO in OWL-S semantične spletne storitve, ki definirajo celovito ogrodje za opis storitev. Oba pristopa izhajata iz semantičnih opisov, ki jih nato s pomočjo ozemljitve povežeta s konkretnimi storitvami. SAWSDL [24] (Semantic Annotations for WSDL and XML Schema) ubira obratno pot. Izhaja iz opisa spletnih storitev (WSDL), ki ga semantično označi. Mnogi avtorji imajo SAWSDL za inkrementalni pristop k semantičnim spletnim storitvam. Poglavitna razlika v primerjavi z do sedaj obravnavanima konceptoma je, da SAWSDL ne predpisuje načina opisa storitev, temveč zgolj ponuja tehnologijo za povezavo obstoječih WSDL opisov storitev s semantičnimi koncepti.

Zaradi načrtovanih omejitev je uporaba SAWSDL precej enostavna. Specifikacija SAWSDL prinaša v WSDL [17] zgolj tri dodatne konstrukte: (1) modelReference – s pomočjo te lastnosti v XML elementu lahko povežemo operacije spletne storitve ali podatkovne tipe, definirane v XML Schema, s konceptom poljubne ontologije, (2) liftingSchema – podatkovni tip, definiran v XML Schema, povežemo s shemo, ki poskrbi za transformacijo iz XML oblike, ki jo vrača spletna storitev, v ustrezno semantično obliko, ter (3) loweringSchema – s pomočjo katere povežemo podatkovni tip s shemo, ki poskrbi za transformacijo podatkov



Slika 2.15: Povezava med WSDL opisi in SAWSDL anotacijami [47]

iz semantične oblike v XML obliko. Slika 2.15 prikazanuje odvisnost med WSDL opisi ter novostmi, ki jih uvaja SAWSDL.

2.3.4.1 Orodja za delo s SAWSDL

Uporabo SAWSDL omogočajo naslednja orodja: (1) WSMO Studio – semantično označevanje WSDL dokumentov, (2) SAWSDL4J – programski vmesnik, ki omogoča razvoj aplikacij na podlagi SAWSDL in WSDL 1.1 specifikacij, (3) Woden4SAWSDL – programski vmesnik, ki omogoča delo s SAWSDL in WSDL 2.0 specifikacijami, (4) Radiant – dodatek za Eclipse, ki omogoča ustvarjanje in objavljanje SAWSDL storitvenih vmesnikov, (5) Lumina – dodatek za Eclipse, ki s pomočjo UDDI in semantičnih opisov omogoča odkrivanje storitev.

2.3.5 Primerjava tehnologij SWS

Tabela 2.3 prikazuje povzetek primerjalne analize vodilnih tehnologij za realizacijo semantičnih spletnih storitev. Primerjali smo OWL-S, WSMO in SAWSDL pristop za katere smo v tabeli podali ključne lastnosti. OWL-S in WSMO sta sorodna pristopa, ki definirata celotno ogrodje za semantično označevanje storitev. Nasprtono temu, SAWSDL zgolj ponuja sredstvo za dodajanje semantičnih opisov k posameznim spletnim storitvam.

Tabela 2.3: Primerjava različnih tehnologij SWS (Povzeto po [1], [75], [46])

	OWL-S	**WSMO**	**SAWSDL**
Komentar	Vrhnja OWL ontologija za modeliranje storitev.	Ontologija za modeliranje storitev, izražena v WSML.	Uporaba razširitvenih elementov WSDL za semantično anotacijo.
Namen	Avtomatizirano odkrivanje, kompozicija in proženje storitev.	Avtomatizirano odkrivanje, kompozicija in proženje storitev.	Obogatitev WSDL opisov s semantiko.
Semantični jezik	OWL	WSML	Nedefinirano (OWL, WSMO, UML,...)
Formalizem	opisna logika	opisna logika, logika prvega reda in logično programiranje (F-Logic)	odvisno od uporabljene ontologije
Povezava z WSDL	Definira povezavo z WSDL preko ozemljitvenega modela, vendar se prekrivajo definicije vhodov, izhodov in operacij.	Definira povezavo z WSDL preko ozemljitvenega modela, vendar se prekrivajo definicije vhodov, izhodov in operacij.	Definira anotacije direktno v WSDL dokumentu.
Izvajalno okolje	OWL-S orodja	WSMX	Radiant, Wooden4J, SAWSDL4J
Odkrivanje	Uporabnik definira abstraktno definicijo spletne storitve, izvajalno okolje poišče konkretno storitev, ki ustreza abstraktni.	Uporabnik definira cilje, izvajalno okolje išče storitve s sposobnostmi, ki zadovoljujejo cilje.	Ni določeno. V praksi se pogosto SAWSDL uporablja v kombinaciji z UDDI registrom.
Kompozicija	OWL-S podpira orkestracijo, ampak ne koreografije. Orodje za te namene ne obstaja. Možno je uporabiti BPEL4WS.	WSMO definira orkestracijo in koreografijo, vendar WSMX implementira zgolj koreografijo	Za orkestracijo je možno uporabiti BPEL4WS.
Mediacija	Ni podpore na konceptualnem in implementacijskem nivoju.	Podpira podatkovno in procesno mediacijo.	Podatkovna mediacija (loweringSchema in liftingSchema).

2.4 Sorodno delo

V doktorski disertaciji se ukvarjamo z integracijo heterogenih podatkovnih virov na osnovi ontologij z uporabo semantično označenih storitev. Sorodna dela lahko uvrstimo v dve kategoriji, prva kategorija obsega integracijo podatkovnih virov na osnovi ontologij, medtem ko druga kategorija obsega porazdeljeno povpraševanje na osnovi semantičnih tehnologij. V nadaljevanju podajamo povzetek vseh sorodnih del iz obeh kategorij.

2.4.1 Integracija podatkovnih virov na osnovi ontologij

Predstavljamo sorodna dela na področju integracije podatkovnih virov na osnovi ontologij. Pri pregledu literature smo odkrili še več pristopov, ki uporabljajo ontologije za integracijo podatkov, vendar smo izločili pristope, ki ne opisujejo podrobno sistema za integracijo ali ne podajajo vloge ontologij v sistemu.

2.4.1.1 OI1 - Levy et al.

Pristop [55] predstavlja eno izmed pionirskih del na področju integracije podatkovnih virov. Prispevek smo vključili med sorodna dela kljub temu, da avtorji ne uporabljajo ontologij, saj v času, ko je članek nastal (leto 1996), ontologije še niso bile razširjena tehnologija. Ne glede na to avtorji za globalno shemo uporabljajo podatkovni model, ki je konceptualno podoben ontologijam. Podatkovni model je kombinacija relacijskega modela (RM) in objektno orientiranega (OO) pristopa. Uporabljajo razrede iz OO ter relacije iz RM. Razrede je možno gnezditi v hierarhijo z dedovanjem. Podatkovni model, ki so ga definirali avtorji je na konceptualnem nivoju precej podoben sodobnemu OWL modelu.

Pristop k integraciji podatkovnih virov, ki so ga predstavili avtorji, velja za prvi LAV pristop (poimenovanje LAV je bilo naknadno določeno v akademski skupnosti, avtorji sami so svoj pristop poimenovali "Information Manifold"). Pristop rešuje problem integracije spletnih podatkovnih virov. Osnovna zamisel pristopa je, da spletni viri (npr. spletne strani, portali) izpostavijo svoje podatke preko t.i. "vmesniški programov". Ti vmesniški progami so nato označeni s koncepti

in relacijami iz globalne sheme. Vlogo in zasnovo vmesniških programov lahko primerjamo s spletnimi storitvami, ki so danes precej razširjene.

Način interakcije s sistemom je sledeč. Uporabnik poda povpraševanje skladno z globalno shemo. Sistem na osnovi povpraševanja ter oznak zmožnosti vmesniških programov identificira vire (vmesniške programe), ki ponujajo podatke potrebne za odgovor na povpraševanje. Sledi izvajanje povpraševanj na vsakem izmed identificiranih virov, integracija rezultatov ter prikaz integriranega rezultata uporabniku. Ta prispevek je postavil temelje za sodobne pristope integracije heterogenih podatkovnih virov. Scenarij uporabe tega pristopa je konceptualno praktično enak scenariju pristopa, ki smo ga razvili v okviru doktorske disertacije.

2.4.1.2 OI2 - Calvanese et al.

Avtorji predstavljajo GAV pristop za integracijo podatkovnih virov [15]. Za globalno shemo uporabljajo ontologijo v opisni logiki. V ta namen predstavljajo svojo izpeljanko opisne logike $DL-Lite_A$. $DL-Lite_A$ opisno logiko so zasnovali tako, da vključuje samo aksiome, ki omogočajo učinkovito izvajanje povpraševanj na večji množici podatkov.

Ta pristop je tipičen GAV pristop. Globalna shema je definirana v $DL-Lite_A$ opisni logiki, izvorni podatkovni viri so relacijske baze, preslikave pa so definirane kot povpraševanja nad relacijsko bazo. Za vsak koncept in relacijo iz globalne sheme, obstaja preslikava, ki je definirana kot povpraševanje nad relacijsko bazo. Primer povpraševanj je prikazan na spodnjem primeru. Pristop omogoča sklepanje na nivoju $DL-Lite_A$ opisne logike. Pomanjkljivost pristopa je, da kot lokalne vire obravnava samo relacijske baze, ter ne naslavlja distribuiranega povpraševanja, saj ta del prepuščen komercialni rešitvi za SQL federacijo.

M_1: `SELECT S_CODE` $\rightsquigarrow$ *Student*(**st**(S_CODE))
`FROM STUDENTS`
`WHERE DOB <= '1990/01/01'`

M_2: `SELECT S_CODE,S_NAME FROM STUDENTS WHERE DOB <= '1990/01/01'` $\rightsquigarrow$ *Student*(**st**(S_CODE),S_NAME:xsd:string)

M_3: `SELECT S_CODE,C_CODE FROM COURSES` $\rightsquigarrow$ *Student*(**st**(S_CODE),**co**(C_CODE))

2.4.1.3 OI3 - Sahoo et al.

Avtorji so realizirali sistem za integracijo podatkov na osnovi ontologije [71]. Tri podatkovne baze iz domene bioinformatike so združili po pristopu podatkovnega skladišča. Definirali so globalno shemo v obliki ontologije, podatke so nato izvozili iz treh baz in jih transformirali v XML/RDF format, ki je skladen z ontologijo, ki so jo definirali. Tako transformirane podatke so naložili v klasično RDF hrambo podatkov. Sistem omogoča izvajanje enostavnejših sklepanj ter povpraševanje nad integriranimi podatki, kar predhodno ni bilo mogoče.

2.4.1.4 OI4 - Nachouki et al.

Avtorji predstavljajo pristop za združevanje storitev in podatkov iz spletnih strani [64]. Integracija poteka na nivoju XML jezika. Sheme izvornih podatkovnih virov so definirane z DTD jezikom. Avtorji so po vzoru DTD jezika definirali svoj jezik MDL, ki je namenjen izražanju sheme med seboj povezanih XML podatkovnih virov. Jezik za izvajanje povpraševanj je XQuery. Sistem integracije podatkov ne izvaja samodejno, temveč mora uporabnik eksplicitno navesti vire ter povezave med viri. Sistem nato samodejno izvede povpraševanja in na osnovi deklaracij v povpraševanju združi podatke.

Pristop obravnava tudi heterogenost strukture podatkov. V ta namen uporabljajo ontologijo, ki je povezana z DTD shemami. Avtorji navajajo, da v primeru heterogenosti podatkov sistem samodejno opravi mediacijo podatkov na osnovi ontologije. Podrobnosti, kako sistem izvede mediacijo, niso podane.

2.4.1.5 OI5 - Verstichtel et al.

Gre za praktičen primer integracije podatkovnih virov na osnovi ontologij [78]. Avtorji so integrirali podatke iz dveh podatkovnih virov (relacijska baza in RDF hramba podatkov). Pristopa, ki so ga izbrali avtorji, ni mogoče natančno umestiti v področje integracije podatkovnih virov, kljub temu je najbližje GAV pristopu.

Avtorji so definirali dva nivoja ontologij, globalno in lokalne ontologije. Povpraševanje poteka v SPARQL jeziku na nivoju globalne ontologije. Lokalne ontologije so dejansko specializacija globalne ontologije. Lokalni podatkovni viri pa so bodisi predstavljeni skladno s specializirano lokalno ontologijo ali pa obstajajo ovojnice, ki transformirajo podatke skladno z lokalno ontologijo. Ko uporabnik poda povpraševanje, skladno z globalno ontologijo, sistem enostavno posreduje povpraševanja lokalnim podatkovnim virov. Ker so lokalne ontologije specializacije globalne ontologije, preslikave med globalno in lokalnimi ontologijami niso potrebne.

2.4.1.6 OI6 - Cruz in Xiao

Avtorji so razvili GAV sistem za integracijo podatkovnih virov na osnovi ontologij [18]. Tako globalna shema kot lokalni podatkovni viri so predstavljeni z RDFS ontologijami. Med globalno in lokalnimi ontologijami so preslikave, ki omogočajo transformacijo povpraševanj. Preslikave temeljijo na skupnem slovarju. Uporabnik poda povpraševanje v RQL jeziku, sistem na osnovi preslikav prevede podano povpraševanje v lokalna povpraševanja (v skladu z lokalnimi ontologijami). Sistem nato združi podatke iz vseh lokalnih virov ter vrne uporabniku rezultat v RDF obliki.

2.4.1.7 OI7 - Seng in Kong

Avtorji so predstavili model za integracijo heterogenih podatkovnih virov, ki uporablja XML in ontologije kot globalno shemo [74]. Integracija poteka na nivoju XML, jezik za povpraševanja pa je XQuery. Sistem ima vzporedno dve globalni shemi, eno v XML in eno, predstavljeno kot OWL ontologijo. Sistem temelji na GAV pristopu. Preslikave so definirane med globalno XML shemo

in lokalnimi XML shemami. Vzporedno z XML shemo je ontologija. Namen ontologije v okviru tega pristopa je, da podrobneje definira relacije med koncepti v globalni shemi. Ontologija se uporablja za odkrivanje implicitnih relacij v XQuery povpraševanju. Podrobnejše informacije o transformaciji povpraševanja na osnovi ontologije niso podane.

2.4.1.8 OI8 - Lam et al.

Avtorji so predstavili GAV pristop za integracijo podatkovnih virov na osnovi ontologij [48]. Za odpravo heterogenosti podatkovnih virov pristop uporablja transformatorje iz relacijske podatkovne baze v RDF. Na osnovi transformacij sistem generira lokalno ontologijo za vsak vir. Na osnovi dobljenih lokalnih ontologij pristop predvideva specifikacijo globalne ontologije in določitev preslikav z uporabo *owl:sameAs* predikata. Povpraševanj ni potrebno prevajati, saj so viri in globalna shema neposredno povezani. Sistem v času povpraševanja vhodno povpraševanje posreduje vsem virov, nakar izvede agregacijo rezultatov.

2.4.2 Porazdeljeno povpraševanje na osnovi semantičnih tehnologij

Sorodno delo iz te kategorije vključuje pristope, ki se ukvarjajo s povpraševanjem po podatkovnih virih, ki so porazdeljeni po omrežju. Ti pristopi se omejujejo zgolj na uporabo semantičnih tehnologij. Ključni izziv je tako, kako zagotoviti čim bolj učinkovito povpraševanje nad porazdeljenimi podatki na nivoju svetovnega spleta z uporabo semantičnih tehnologij, ne pa, kako zagotoviti povpraševanje po heterogenih podatkovnih virih, kar je izziv prejšnje kategorije sorodnega dela.

2.4.2.1 DQ1 - DARQ

Avtorji so naredili sistem za povpraševanje po podatkih, ki so distribuirani med več SPARQL končnimi točkami [67]. Pristop temelji na transformaciji osnovnega povpraševanja v več sekvenčnih podpovpraševanj, ki jih kasneje združijo. Sestavljen je iz štirih korakov:

- razčlenjevanje - razčlenitev in preoblikovanje povpraševanja v drevesni sparql model,
- načrtovanje povpraševanja - pogon povpraševanja identificira potencialne sparql končne točke, ki bi lahko odgovorile na posamezne dele povpraševanja. Na osnovi pridobljenih informacij pogon povpraševanja razgradi podano povpraševanje na več podpovpraševanj - eno podpovpraševanje za vsak vir,
- optimizacija - optimizator povpraševanj na osnovi pod-povpraševanj zgradi in optimira načrt izvajanja povpraševanj,
- izvajanje povpraševanj - pogon za povpraševanja izvede vsa podpovpraševanja in integrira rezultate.

Posamezne SPARQL končne točke so označene z zmožnostmi in zahtevami. Zmožnosti označujejo, katere podatke nudi posamezna končna točka, zahteve pa določajo, kateri vhodni podatki so potrebni, da lahko končna točka poda rezultat. Zmožnosti posameznega vira so definirane kot seznam predikatov, ki jih ponuja končna točka. Slabost tega pristopa je, da morajo biti v izvornem povpraševanju podani vsi predikati, sicer ne zmore identificirati končnih točk, ki so potrebne za pridobitev odgovora na povpraševanje. Avtorji podajajo tudi implementacijo predlaganega pristopa, ki temelji na razširitvi odprtokodnega pogona za izvajanja sparql povpraševanj - ARQ [1].

2.4.2.2 DQ2 - Networked graphs

Avtorji so razvili pristop, ki omogoča povezavo med porazdeljenimi RDF podatkovnimi viri (datotekami) [72]. V ta namen so uvedli nov koncept omreženih grafov (ang. networked graphs). Omreženi grafi razširjajo RDF standard, tako da je mogoče znotraj posamezne RDF datoteke podati SPARQL povpraševanje do druge RDF datoteke. Na takšen način je mogoče deklarativno povezati dve RDF datoteki. Ob izvajanju povpraševanj nad datoteko se avtomatično rekurzivno izvedejo povpraševanja do povezanih RDF datotek.

[1] http://jena.sourceforge.net/ARQ

Pristop, ki so ga uvedli avtorji, razširja RDF na način, ki je nazaj kompatibilen s standardnim RDF modelom. Sistemi, ki niso zmožni procesiranja omreženih grafov, bodo pridobili samo informacije, ki so eksplicitno podane; sistemi, ki pa so zmožni procesiranja omreženih grafov, pa bodo še pridobili podatke iz povezanih virov. Izpis 2.5 prikazuje omrežen graf v RDF N3 notaciji, razširitev avtorjev je podana v zadnjem trojčku (predikat g:definedBy).

```
:bobFOAF {
:bobFOAF foaf:primaryTopic
:Bob.
:Bob
foaf:name
"Bob"^^xsd:String.
:Bob
foaf:currentProject :K-Space.
:K-Space foaf:fundedBy
:EU.
:Bob
foaf:phone
"+49-261-287-2868".
:bobFOAF g:definedBy
"CONSTRUCT␣{:Bob␣foaf:knows␣?person}
FROM␣:mikesProject
WHERE␣{␣?person␣foaf:currentProject␣:SemWebProject
FILTER␣(?person␣!=␣:Bob)␣}"^^g:query.
}
```

Izpis 2.5: Primer omreženega grafa zapisanega v RDF N3 notaciji

2.4.2.3 DQ3 - SemWIQ

Avtorji so razvili pristop za federacijo SPARQL končnih točk [49][50]. Za vsako končno točko je potrebno definirati koncepte, za katere končna točka ponuja rezultate. Na osnovi teh definicij avtorji izvedejo transformacijo povpraševanja na podoben način kot pri pristopu DQ1. Poglavitna razlika je, da so pri pristopu DQ1 zmožnosti virov definirane s predikati, medtem ko so pri tem pristopu definirane s koncepti. Pristop DQ1 zahteva da so vezani vsi predikati v povpraševanju, medtem ko DQ3 tega ne zahteva, temveč opravi identifikacijo potencialnih

SPARQL končnih točk samo na nevezanih objektih trojčkov povpraševanja.

Gre za GAV pristop, kjer so izvorni podatkovni viri oviti v SPARQL ovojnice, globalna shema pa je avtomatično pridobljena na osnovi slovarjev, ki jih uporabljajo ovojnice. Ovojnice morajo biti zasnovane tako, da tvorijo enovito globalno shemo. Pristop ne obravnava heterogenosti strukture podatkovnega modela posameznih virov.

2.4.2.4 DQ4 - Battle & Benson

Avtorja sta razvila pristop, ki omogoča federacijo med SPARQL končnimi točkami ter REST storitvami [3]. Vhodi in izhodi REST storitev so semantično označeni, kar ustvari relacijo med REST storitvami in preostalimi podatki, ki so na voljo v SPARQL končni točki. Sistem je na osnovi globalne ontologije sposoben narediti federacijo med SPARQL končnimi točkami, REST storitvami in relacijskimi viri, ovitimi v SPARQL končne točke. Federacijo opravijo z lastno komponentno Semantic Query Decomposition (SQD). SQD je del zaprte komercialne rešitve, zaradi tega ni na voljo podrobnih informacij, kako je dosežena federacija heterogenih virov, niti kakšne so natančne zmožnosti in omejitve sistema.

2.4.2.5 DQ5 - Buil-Aranda & Corcho

Pristop omogoča federacijo sparql končnih točk [12]. V ta namen so avtorji razširili specifikacijo SPARQL, da omogoča določitev več SPARQL podatkovnih virov v samem povpraševanju. Pristop dodaja k posamezni SPARQL spremenljivki atribut, ki določa, v kateri SPARQL končni točki naj pogon za izvajanje povpraševanja pridobi podatke. Izpis 2.6 prikazuje takšno razširjeno povpraševanje, kjer sta "iproclass"in "geneid"heterogena SPARQL podatkovna vira.

Pomanjkljivost tega pristopa je, da integracija ni avtomatizirana, temveč poteka na osnovi specifikacije, ki jo je s povpraševanjem podal uporabnik. Samo izvajanje takšne federacije je relativno trivialno, saj je sam potek integracije deklarativno podan s povpraševanjem samim.

```
PREFIX ipr: <http://bio2rdf.org/ns/iproclass>
PREFIX gn: <http://bio2rdf.org/ns/bio2rdf>
SELECT ?iproclass.protein ?geneid.taxon
```

```
FROM iproclass: <http://iproclass.bio2rdf.org/sparql>
FROM geneid: <http://geneid.bio2rdf.org/sparql>
WHERE{
?iproclass.protein ipr:xGeneid ?iproclass.gene .
OPTIONAL {?geneid.gene gn:xTaxon ?geneid.taxon}}
```

Izpis 2.6: Primer razširitve SPARQL za federacijo SPARQL končnih točk

2.4.2.6 DQ6 - Buil-Aranda et al.

Avtorji predstavijo formalni semantični model za federacijo, ki jo uvaja SPARQL verzija 1.1, ter uvedejo optimizacije za učinkovito izvajanje federacije, ki temeljijo na transformaciji povpraševanja [13]. Specifikacija SPARQL verzija 1.1 vključuje federacijo SPARQL končnih točk na način, podoben pristopu DQ5. Federacija je podana deklarativno v samem povpraševanju. V ta namen je dodan nov operator *service*, ki izvede podpovpraševanje na drugih SPARQL končnih točkah. Federacija med osnovnim povpraševanjem ter gnezdenemi podpovpraševanji je dosežena z uporabo standardnih SPARQL operatorjev. Izpis 2.7 prikazuje federacijo v SPARQL verziji 1.1, ki vsebuje pod-povpraševanje na dve heterogeni SPARQL končni točki.

```
SELECT ?pubmed ?gene1 ?mesh ?descriptor ?meshReference
WHERE
{
{SERVICE <http://127.0.0.1:2020/sparql-pubmed> {
?gene1 <http://bio2rdf.org/geneid_resource:pubmed_xref>
   ?pubmed .}}.
{SERVICE <http://pubmed.bio2rdf.org/sparql> {
?pubmed <http://bio2rdf.org/pubmed_resource:meshref> ?mesh .
?mesh <http://bio2rdf.org/pubmed_resource:descriptor>
   ?descriptor .}}.
OPTIONAL { SERVICE <http://127.0.0.1:2021/sparql-mesh> {
?meshReference <http://www.w3.org/2002/07/owl#sameAs>
   ?descriptor .}}.
}
```

Izpis 2.7: Primer federacije v SPARQL verziji 1.1

2.4.2.7 DQ7 - Hartig et al.

Avtorji so razvili pristop, ki omogoča povpraševanje po distribuiranih RDF virih, tako da dinamično, v času izvajanja, pridobiva podatke iz RDF virov, ki se nahajajo na svetovnem spletu [40]. Pristop izhaja iz predpostavke, da je možno posamezne vire znotraj RDF trojčkov dereferencirati in da se ti viri nahajajo na svetovnem spletu. Na ta način dobimo množico virov, ki so predstavljeni z URI naslovom. Za nekatere od teh virov se na URI naslovu, ki jih identificira, nahaja RDF datoteka, ki podrobneje opisuje ta vir.

Pristop DQ7 vsebuje algoritem, ki omogoča povpraševanje po RDF virih, ki so med seboj povezani na zgoraj opisan način. Algoritem vsebuje optimizacijo načrta izvajanja povpraševanj na način, ki minimizira število RDF virov, ki jih je potrebno pridobiti iz spleta, vendar pa je kljub temu mogoče odgovoriti na podano povpraševanje.

Poglavje 3

Model za integracijo podatkovnih virov na osnovi ontologij s semantično označenimi storitvami

3.1 Integracija heterogenih podatkovnih virov z uporabo semantično označenih storitev

3.1.1 OSII model za integracijo

V tem poglavju predstavljamo pristop OSII (ang. Ontology and service based information integration), ki smo ga razvili v okviru doktorske disertacije. Ključna značilnost pristopa je uporaba storitev kot podatkovnih nosilcev za integracijo informacij na osnovi ontologij. Storitve tako služijo kot dostopna točka do heterogenih podatkovnih virov, ki so lahko realizirani s poljubno tehnologijo (npr. relacijske podatkovne baze, dokumenti, preglednice, podatki, objavljeni na svetovnem spletu, ali celo podatki, ki so izolirani znotraj aplikacij).

Slika 3.1 prikazuje konceptualno zasnovo OSII modela za integracijo podatkovnih virov. OSII sestavljajo štiri ključne komponente:

- **Ontologija** predstavlja globalno shemo sistema za integracijo podatkovnih virov. Storitve so semantično označene na osnovi konceptov, ki so definirani v ontologiji. Na osnovi ontologije uporabniki podajajo povpraševanja. Tako

je ontologija edini sestavni del OSII, v katerega imajo vpogled uporabniki, ki izvajajo povpraševanja.

- **Storitve** nastopajo v vlogi podatkovnih nosilcev. S proženjem storitev sistem dostopa do podatkov, ki se nahajajo v heterogenih podatkovnih virih. OSII samodejno proži storitve glede na vhodna povpraševanja na osnovi semantičnih opisov storitev.

- **Materializirani podatki.** Sistem omogoča, da podatke, ki se ne spreminjajo pogosto, materializiramo (podatke izvozimo iz prvotnih podatkovnih virov ter jih uvozimo v OSII). Na tak način lahko optimiziramo delovanje sistema, saj lahko za vsako vrsto podatka izbiramo, ali bo pridobljen s proženjem storitve ali pa ga uvozimo in bo pridobljen iz lokalne hrambe podatkov.

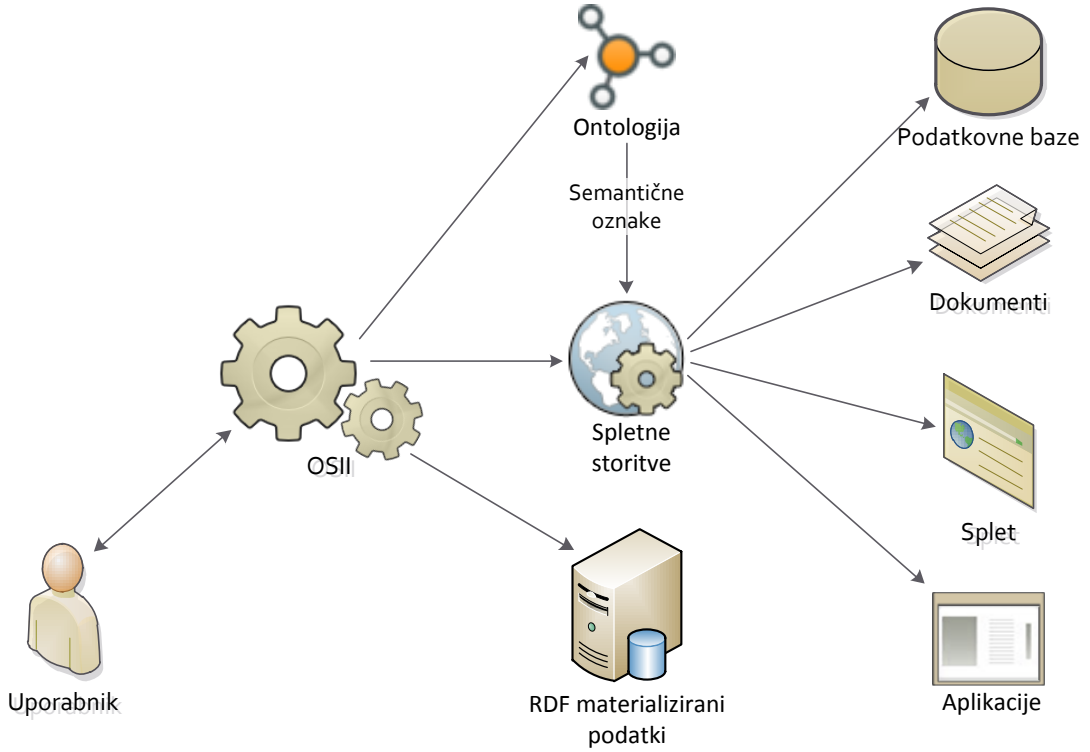
- **Sparql končna točka** predstavlja stično točko med uporabniki in OSII sistemom. Uporabniki pošljejo sparql povpraševanja, ki so skladna z ontologijo, sparql končni točki. Sistem nato samodejno izvede kompozicijo storitev, jih proži ter združi podatke.

Slika 3.2 predstavlja arhitekturo predlaganega sistema za integracijo heterogenih podatkovnih virov. Osnovni predpogoj pri razvoju pristopa za integracijo je bil, da mora biti uporaba takšnega sistema za uporabnika povsem transparenta. To pomeni, da

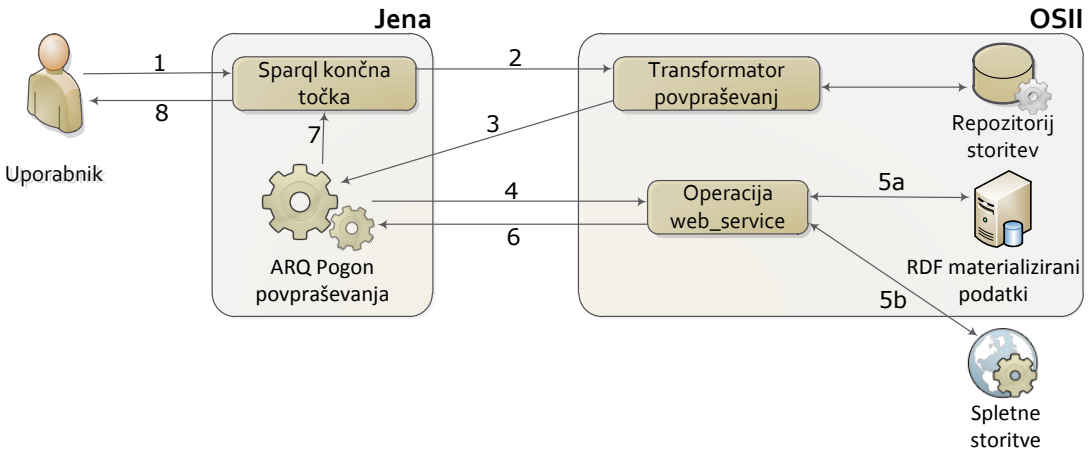
- uporabniku ni treba vedeti, katere storitve se bodo prožile, katere storitve so na voljo sistemu za integracijo, niti kako poteka integracija podatkov, ter

- uporabnik izvaja povpraševanja, kot je to običajno za sisteme, ki temeljijo na semantičnih tehnologijah.

Razvit pristop omogoča dinamično pridobivanje podatkov, ki jih zagotavljajo storitve. Sistem na osnovi globalne ontologije ter semantičnih opisov storitev v času izvajanja povpraševanja identificira storitve, ki lahko podajo odgovor na povpraševanje. Identificirane storitve sistem samodejno združi, pripravi vhodne



Slika 3.1: Konceptualna zasnova OSII za integracijo heterogenih podatkovnih virov na osnovi semantično označenih storitev



Slika 3.2: Arhitektura modela OSII

podatke ter jih tudi samodejno proži. Celoten proces se izvede v času izvajanja povpraševanja.

OSII temelji na razširitvi SPARQL algebre. Algebro smo razširili z novo operacijo *web_service*, ki proži storitev na določenem naslovu. Vsaka operacija *web_service* predstavlja natanko eden klic storitve. Vhodni podatki za posamezno storitev so lahko podani v povpraševanju samem, lahko jih zagotavljajo druge storitve ali pa jih sistem pridobi samodejno iz materializiranih RDF podatkov. Potek izvajanja povpraševanja v okviru OSII poteka v dveh fazah:

- **transformacija povpraševanj:** sistem transformira vhodno povpraševanje, ki ga je podal uporabnik, v povpraševanje, ki vsebuje klice spletnih storitev z operacijo *web_service*. Takšno povpraševanje lahko izvede standardni pogon sklepanja, ki ima implementirano operacijo *web_service*. Transformirano povpraševanje predstavlja načrt proženja storitev ter načrt integracije podatkov zapisan v razširjeni sparql algebri.

- **izvajanje operacije web_service:** komponenta, ki implementira operacijo, skrbi za proženje storitev ter odpravo heterogenosti med podatki. Implementacija operacije mora vsebovati:

  - transformacija vhodnih podatkov storitve v obliko, ki jo zahteva storitev,
  - proženje storitve,
  - transformacijo podatkov, ki jih je vrnila storitev v obliko, ki je skladna z globalno ontologijo.

Model smo zasnovali tako, da ga je mogoče implementirati kot razširitev obstoječih sparql pogonov povpraševanj. V okviru doktorske naloge smo razvili prototip, ki je realiziran kot razširitev odprtokodne pogona za izvajanje sparql povpraševanj (ARQ[1]). ARQ je del odprtokodne knjižnice Jena[2]. Ta knjižnica omogoča delo s tehnologijami semantičnega spleta (RDF, OWL, sparql). Prototip vsebuje implementacijo transformatorja povpraševanj ter implementacijo

[1] http://jena.sourceforge.net/ARQ/
[2] http://jena.sourceforge.net/

operacije *web_service*. Slika 3.2 prikazuje arhitekturo sistema ter zaporedje komunikacije med posameznimi komponentami.

3.1.2 Vloga storitev v OSII

OSII je LAV [1] sistem za integracijo podatkovnih virov, kjer je globalna shema predstavljena z ontologijo, zapisano v opisni logiki. Izvorni podatkovni viri so realizirani s spletnimi storitvami. Semantične oznake podajajo za vsako storitev naslednje informacije:

- **Izhod** storitve določa vrsto podatkov, ki jo storitev vrača. Tansformator povpraševanja a nosnovi izhoda storitve identificira storitve, ki vračajo podatke, zahtevane v povpraševanju.
- **Vhod** storitve določa vrsto podatkov, ki jih storitev zahteva za uspešno proženje storitve.
- **Omejitve** se uporabljajo za ločevanje med storitvami, kadar več storitev vrača isto vrsto podatkov [2]. Omejitve so podane s sparql povpraševanjem.
- **Odvisnosti** definirajo vrsto podatkov iz globalne sheme, od katerih je storitev neposredno odvisna [3]. Odvisnost storitve pogosto predstavlja eden izmed vhodnih podatkov.

Med globalno shemo (ontologijo) in storitvami so definirane LAV preslikave. Preslikave omogočajo transformacijo podatkov med ontologijo in vhodnimi ter izhodnimi sporočili storitev. Storitve prožimo tako, da storitvi pošljemo vhodno

[1] LAV - lokalno kot pogled (glej poglavje 2.1) je pristop, kjer so lokalni podatkovni viri definirani kot preslikave nad globalno shemo. LAV pristop je bolj fleksibilen od GAV, pomanjkljivost LAV pristopa je težja obdelava povpraševanj. OSII skrbi za avtomatično obdelavo povpraševanj na osnovi LAV preslikav.

[2] Za primer privzamimo, da imamo več storitev, ki vračajo podatke o vremenu. Posamezne storitve vračajo podatke samo za omejeno geografsko regijo (npr. Slovenija, Hrvaška...). Geografsko področje za katerega storitev vrača podatke o vremenu je podano z omejitvami storitve na osnovi katerih sistem v času izvajanja izbere ustrezno storitev.

[3] Privzamimo, da imamo storitve, ki vračajo podatke o vremenu. Takšne storitve so neposredno odvisne od neke geografske lokacije(kraja). Slednji predstavlja odvisnost storitve. Sistem za vsak primerek odvisnega razreda proži storitev - v podanem primeru sistem za vsak kraj proži storitev za pridobivanje podatkov vremena.

sporočilo, storitev pa vrne rezultat v obliki izhodnega sporočila. Za vsak izvorni podatkovni vir, torej storitev, imamo tako podani dve shemi - shemo vhodnega sporočila in shemo izhodnega sporočila. Pristop, ki smo ga razvili ima za vsak vir definirani natanko dve preslikavi, in sicer preslikavo med globalno shemo in vhodnim sporočilom ter preslikavo med globalno shemo in izhodnim sporočilom.

Pri pristopu, ki ga predstavljamo, je globalna shema definirana neodvisno od storitev. Le-te so označene s koncepti iz globalne sheme. Ključna prednost takšnega pristopa je, da je možno nove vire dodajati relativno enostavno. Ob dodajanju virov ni potrebno spreminjati globalne sheme, potrebno je zgolj označiti nov vir [54].

Ključen izziv, ki je skupen vsem LAV pristopom, je transformacija povpraševanj. Za razliko od GAV pristopa, kjer je za vsak element globalne sheme podan načrt pridobivanja podatkov iz izvornih podatkovnih virov, je potrebno pri LAV pristopu razviti algoritme, ki to opravijo samodejno v času izvajanja povpraševanja. Posplošeno lahko rečemo, da načrt za pridobivanje podatkov iz izvornih podatkovnih virov pri GAV določi snovalec sistema za integracijo v času načrtovanja, medtem ko pri LAV pristopu to delo opravijo algoritmi v času izvajanja.

Ključna prednost LAV sistema je večja fleksibilnost. LAV sistem je možno razširjati mnogo bolj enostavno kot GAV sistem. Poglavitna pomanjkljivost LAV pristopa je, da je transformacija povpraševanj precej bolj kompleksna. V kolikor primerjamo oba pristopa z vidika načrtovanja, lahko sklenemo, da je pri pristopu LAV osnovna naloga specifikacija vsebine izvornih virov na osnovi globalne sheme. GAV predstavlja večji izziv pri načrtovanju, saj je potrebno za vsak element globalne sheme določiti, kako pridobiti podatke iz izvornih virov na osnovi povpraševanj.

Ključna prednost OSII pristopa, v primerjavi s sorodnimi pristopi, je podpora sklepanju nad podatki, ki jih nudijo storitve. OSII podpira sklepanje na podmnožici **OWL2 QL** profila. Ta profil sklepanja je zasnovan na opisni logiki in se uporablja za sisteme, ki želijo ponuditi osnovni nivo sklepanja na osnovi opisne logike, vendar želijo hkrati ohraniti visoko zmogljivost izvajanja povpraševanj. Sklepanje na osnovi profila OWL2 QL ima logaritemsko časovno zahtevnost [41].

3.1.3 Formalni model

V tem odseku predstavljamo formalni model pristopa OSII, ki smo ga razvili v okviru doktorske disertacije. OSII formalni model je razširitev LAV formalnega modela za integracijo heterogenih podatkovnih virov, ki ga je definiral Lenzerini [54]. Osnovni LAV formalni model, ki smo ga razširili, je definiran kot (za podrobno definicijo modela glej poglavje 2.1.1):

$$I = \{G, S, M\} \tag{3.1}$$

kjer je:

- G - Globalna shema,
- S - Množica lokalnih shem,
- M - Množica preslikav, ki zagotavljajo preslikavo podatkov med globalno in lokalnimi shemami:

$$\begin{aligned} q_S &\rightarrow q_G \\ q_G &\rightarrow q_S \end{aligned} \tag{3.2}$$

  kjer je preslikava $q_s \rightarrow q_g$ definirana kot preslikava povpraševanja q, zapisanega v abecedi lokalne sheme, v enakovredno povpraševanje v abecedi globalne sheme.

Na osnovi obstoječega formalnega modela, ki smo ga povzeli zgoraj, definiramo model za integracijo podatkov iz heterogenih virov z uporabo semantičnih spletnih storitev I kot množico:

$$I = \{G, WS, A\} \tag{3.3}$$

kjer je:

- G - Globalna shema, ki je izražena z ontologijo,
- WS - Množica spletnih storitev,

- A - Množica semantičnih opisov spletnih storitev, ki omogočajo proženje storitev in preslikavo podatkov med globalno ontologijo G in spletnimi storitvami WS.

Vsaka storitev ima pripadajoče semantične opise, ki jih definiramo kot n-terico:

$$A = \{WS_A, WS_{in}, WS_{out}, WS_{dep}, WS_{res}, M_I, M_O\} \tag{3.4}$$

kjer je (vrednosti v zavitih oklepajih določajo kardinalnost posameznega elementa):

- WS_A {1} – dostopna točka spletne storitve omogoča proženje storitve. Podatki, ki so potrebni za proženje so odvisni od vrste storitve (REST ali SOAP). Ti podatki nimajo vpliva na model za integracijo, temveč nudijo zgolj informacijo kako fizično prožiti storitev.

- WS_{in} {0-n} – vhod storitve določa zahteve spletne storitve, ki so izražene skladno z ontologijo G,

- WS_{out} {1-n} – izhod storitve določa zmožnosti oz. funkcionalnosti spletne storitve, ki so izražene skladno z ontologijo G,

- WS_{dep} {0-1} – odvisnosti spletne storitve specificira elementov iz G, od katerih je storitev neposredno odvisna,

- WS_{res} {0-n} – omejitve spletne storitve so podane kot aksiomi k ontologiji G,

- M_I {0-n} – preslikava vhoda preslika podatke, ki so predstavljeni skladno z ontologijo G, v vhodna sporočila spletne storitve,

- M_O {1-n} – preslikava izhoda preslika izhodna sporočila spletne storitve v podatke, ki so predstavljeni skladno z ontologijo G.

Vsaka storitev ima natanko eno dostopno točko WS_A, nič ali več vhodov WS_{in}, enega ali več izhodov WS_{out}, eno ali nobeno odvisnost WS_{dep}, nič ali več

omejitev WS_{res} ter po eno preslikavo vhoda in izhoda za vsak vhod in izhod storitve:

$$\begin{aligned} \forall i \in WS_{in} : \exists M_I(i) \\ \forall o \in WS_{out} : \exists M_O(o) \end{aligned} \tag{3.5}$$

Globalna shema G je ontologija zapisana, v opisni logiki. Na osnovi definicije ontologije (glej 2.2) lahko definiramo globalno shemo G kot:

$$G = \{C, P, A\} \tag{3.6}$$

kjer je:

- C – množica razredov, ki jih definira ontologija,
- P – množica predikatov, ki jih definira ontologija,
- A – množica aksiomov, ki jih definira ontologija skladno z izbranim formalizmom (nadrazred, podrazred, nadlastnost, domena,...). V našem primeru je podprt formalizem OWL2 QL, zaradi tega vsebuje ta množica aksiome, ki jih definira OWL2 QL.

Skladno z definicijo globalne sheme definiramo posamezne elemente preslikave kot:

$$\begin{aligned} \forall i \in WS_{in}(i) &: i \in \{C \cup P\} \\ \forall o \in WS_{out}(o) &: o \in \{C \cup P\} \\ \forall d \in WS_{dep}(d) &: d \in C \end{aligned} \tag{3.7}$$

Standardni jezik za povpraševanje pri uporabi tehnologij semantičnega spleta je SPARQL, skladno s tem tudi v našem modelu uporabljamo SPARQL jezik. Omejitve storitev ter vhodne in izhodne preslikave so definirani z uporabo SPARQL povpraševanj. Naj bo q^{sparql} povpraševanje v SPARQL jeziku. Potem omejitve in preslikave spletne storitve definiramo kot:

$$WS_{res}(r) = q_r^{sparql} \tag{3.8}$$

Vhodne in izhodne preslikave so prav tako definirane s SPARQL povpraševanjem. Običajno SPARQL povpraševanja vsebujejo spremenljivke. Dodatno je k vsakemu povpraševanju dodana še preslikava med spremenljivkami iz SPARQL povpraševanja in potjo do enakovrednega podatka v vhodnem oz. izhodnem sporočilu storitve. Sistem za integracijo podatkovnih virov sestavi vhodno sporočilo tako, da na osnovi SPARQL povpraševanja pridobi podatke za oblikovanje vhodnega sporočila. Za tem generira vhodno sporočilo za storitev na osnovi preslikav med SPARQL spremenljivkami in potjo v vhodnem sporočilu. Pridobitev podatkov iz izhodnega sporočila potega analogno, le v obratnem vrstnem redu. Skladno z opisanim procesom sta vhodna in izhodna preslikava definirani kot:

$$M_I(i) = \{q^{sparql}, \{var^{sparql}, path^{imsg}\}\} \tag{3.9}$$
$$M_O(o) = \{q^{sparql}, \{var^{sparql}, path^{omsg}\}\}$$

kjer je:

- q^{sparql} – povpraševanje nad globalno shemo (ontologijo),
- var^{sparql} – spremenljivka v definiranem SPARQL povpraševanju, ki se uporablja za vzpostavitev relacije med elementom vhodnega oz. izhodnega sporočila in globalno shemo,
- $path^{imsg}$ – pot do posameznega elementa v vhodnem sporočilu (v primeru SOAP storitve je pot podana z XPath izrazom),
- $path^{omsg}$ – pot do posameznega elementa v izhodnem sporočilu (v primeru SOAP storitve je pot podana z XPath izrazom).

3.2 Ontologija za semantično označevanje storitev

Na osnovi formalnega modela, definiranega v prejšnjem poglavju, smo razvili ontologijo OSII-SO za opis storitev v jeziku OWL DL. Ontologija omogoča definicijo preslikave za posamezno storitev. Storitve, ki so opisane skladno s to ontologijo, je možno neposredno uporabiti v sistemu za integracijo. V kolikor želimo v sistem za integracijo podatkovnih virov dodati nov podatkovni vir, je

potrebno zgolj definirati preslikavo, ki skladna z razvito ontologijo. Ontologija, zapisana v OWL, je na voljo v prilogi A, primer opisa storitev pa v prilogi B.

Slika 3.3 prikazuje ontologijo OSII-SO, ki smo jo razvili v okviru doktorske disertacije. Na sliki so v ovalih predstavljeni razredi, *isA* relacije nakazujejo dedovanje med razredi, preostale relacije predstavljajo objektne in podatkovne lastnosti. Osrednji koncept ontologije je razred *Service*, ki predstavlja posamezno storitev. Ta razred ima dva podrazreda *SOAP_Service* in *REST_Service*. Vsak izmed njih predstavljajo storitve, ki so realizirane z drugačno tehnologijo (REST ali SOAP). Oba podrazreda storitve imata specifične podatkovne lastnosti, ki omogočajo proženje storitve. Vsaka storitev v OSII-SO ima podano:

- **shemo vhodnega in izhodnega sporočila** (podatkovne lastnosti *hasInputSchema* in *hasOutputSchema*)
- **omejitve**, ki so podane s podatkovno lastnostjo *hasRestriction*) (WS_res v formalnem modelu, izraz 3.4)
- **odvisnost**, ki je podana z objektno lastnostjo *hasMainConcept* in konceptom *Service_Main_Concept* (WS_dep v formalnem modelu, izraz 3.4)
- **vhod storitve**, ki je podan z objektno lastnostjo *hasInput* in konceptom *Service_Input* (WS_in v formalnem modelu, izraz 3.4)
- **izhod storitve**, ki je podan z objektno lastnostjo *hasOutput* in konceptom *Service_Output* (WS_out v formalnem modelu, izraz 3.4)

Preslikave so definirane z mediatorji (razred *Mediator* v ontologiji). Razred *Mediator* ima tri podrazrede:

- razred *LoweringMediator* definira preslikavo vhodnega sporočila storitve (M_I v formalnem modelu, izraz 3.4),
- razred *LiftingMediator* definira preslikavo izhodnega sporočila storitve (M_O v formalnem modelu, izraz 3.4),
- razred *TransformationMediator* definira preslikavo za poljubno preoblivanje podatkov.

Vsak mediator ima definirano povpraševanje nad globalno shemo, ki predstavlja preslikavo posameznega vhoda ali izhoda (podatkovna lastnost *hasCode* in q^{sparql} v formalnem modelu, izraz 3.9). Dodatno ima vsak mediator tudi določeno preslikavo (razred *Mapping*), ki omgoča pretvorbo sintaktičnega zapisa podatkov med povpraševanjem nad globalno shemo (q^{sparql}) ter vhodnimi in izhodnimi sporočili (glej izraz 3.9, kjer so definirane preslikave v formalnem modelu, koncepta *Mediator* in *Mapping* predstavljato realizacijo tega izraza).

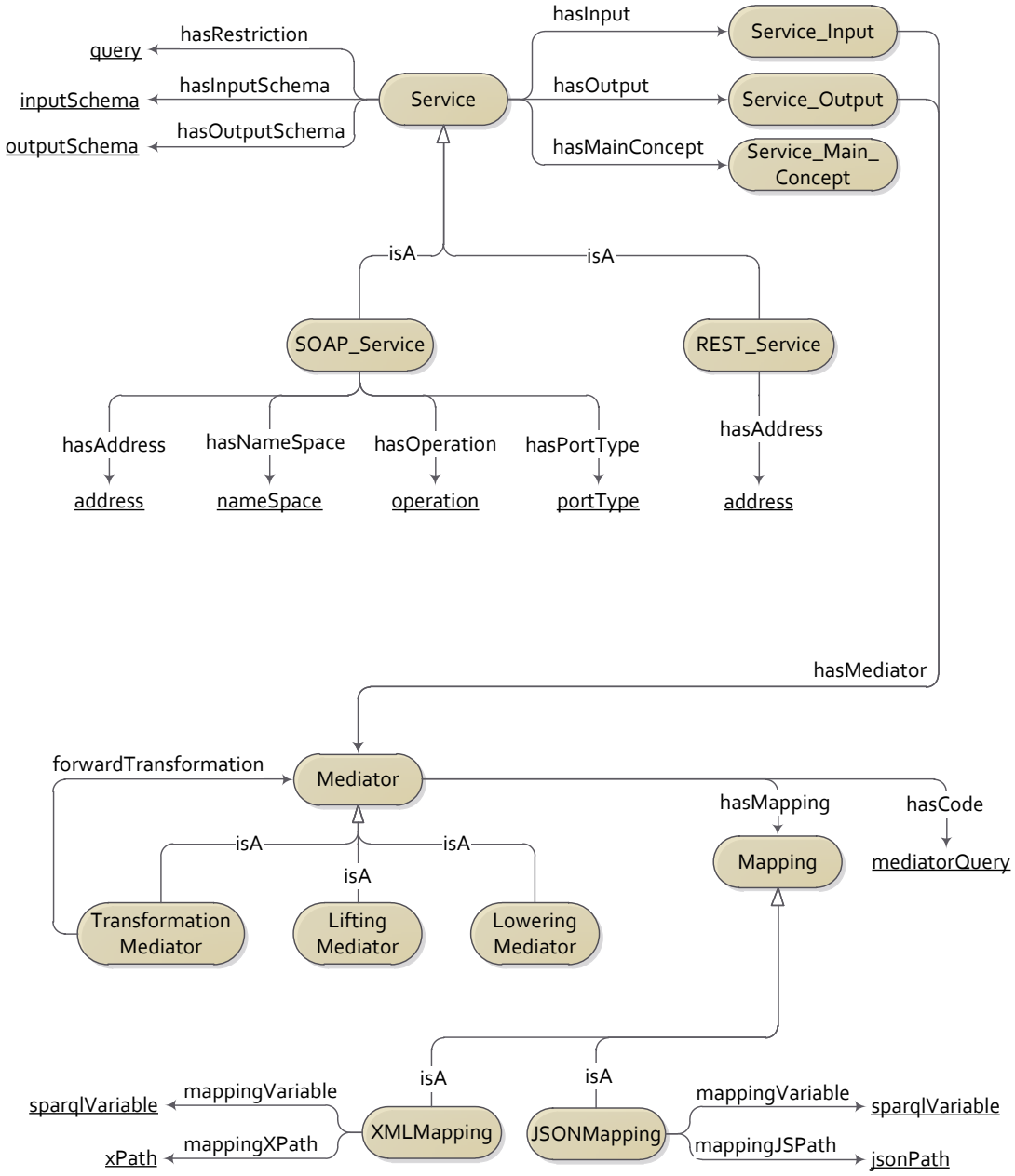
Opis SOAP in REST storitev poteka na enak način, razlikuje se le v specifikaciji podatkovnih lastnosti za proženje storitev ter v specifikaciji preslikav. SOAP storitve uporabljajo preslikave, ki so definirane s konceptom *XMLMapping*, REST storitve pa uporabljajo preslikave, ki so definirane s konceptom *JSONMapping*. Oba razreda imata enakovredne podatkovne lastnosti. Podatkovna lastnost *mappingVariable* je enakovredna var^{sparql} v formalnem modelu (izraz 3.9). Podatkovni lastnosti *mappingXPath* in *mappingJSPAth* sta enakovredni $path^{imsg}$ oz. $path^{omsg}$ v formalnem modelu (izraz 3.9).

3.3 Transformacija SPARQL povpraševanj

Uporabniki podajajo povpraševanja sistemu za integracijo podatkovnih virov v SPARQL povpraševalnem jeziku. Vhodno povpraševanje mora biti skladno z globalno ontologijo. Sistem za integracijo podatkovnih virov nato izvede transformacijo povpraševanja, ki v samo povpraševanje vključi klice k spletnim storitvam. Transformirano povpraševanje sistem preda naprej pogonu izvajanja. V tem poglavju se bomo osredotočili na ključni del pristopa, transformacijo SPARQL povpraševanj.

SPARQL povpraševanja se izvajajo na osnovi SPARQL algebre, ki je definirana v specifikaciji jezika SPARQL [66]. Proženje spletnih storitev smo vključili v SPARQL tako, da smo razširi SPARQL algebro z novim operatorjem *web_service*. Razširitev ne vpliva na povpraševalni jezik, ki ga uporabljajo uporabniki, ti podajo povpraševanja na osnovi SPARQL specifikacije. Razširitev vpliva zgolj na povpraševalni pogon, ki mora imeti implementiran operator *web_service*.

Ključni del našega modela za integracijo je transformacija, ki transformira



Slika 3.3: OSII-SO ontologija za semantično označevanje storitev

vhodno povpraševanje v novo povpraševanje, ki vsebuje klice spletnih storitev, načrt proženja storitev ter pridobivanje materializiranih podatkov. Model, ki smo ga razvili, omogoča, da je del podatkov materializiran v samem sistemu za integracijo, del podatkov pa se pridobi dinamično iz spletnih storitev.

Zaradi lažjega razumevanja bomo transformacijo podatkov opisali na primeru kalkulatorja. Sicer to ni tipičen primer s področja integracije podatkovnih virov, vendar je lahko razumljiv in na njem je mogoče najbolj nazorno prikazati zmožnosti in omejitve sistema za integracijo podatkov. V okviru doktorske disertacije smo naredili evalvacijo prototipnega sistema na realnem primeru iz domene integracije podatkovnih virov (glej poglavje 4.1.2).

3.3.1 Primer integracije na kalkulatorju

Globalna ontologija G vsebuje naslednje elemente:

$$
\begin{aligned}
G &= \{C, P, A\} \qquad (3.10)\\
C &= \{Racun, LeviDel, DesniDel, Vsota\}\\
P &= \{imaLeviDel, imaDesniDel, imaVsoto, imaVrednost, izracunaj\}\\
A &= \{domena(imaLeviDel, Racun), obseg(imaLeviDel, LeviDel),\\
&\quad domena(imaDesniDel, Racun), obseg(imaDesniDel, DesniDel),\\
&\quad domena(imaVsoto, Racun), obseg(imaVsoto, Vsota),\\
&\quad domena(imaVrednost, \{LeviDel, DesniDel, Vsota\}),\\
&\quad obseg(imaVrednost, xsd:int),\\
&\quad domena(izracunaj, Racun), obseg(izracunaj, xsd:boolean)\}
\end{aligned}
$$

Globalna ontologija vsebuje štiri razrede (*Racun, LeviDel, DesniDel* in *Vsota*) in štiri relacije (*imaLeviDel, imaDesniDel, imaRezultat* in *imaVrednost*). Relacija *imaLeviDel* poteka med konceptoma *Racun* in *LeviDel*, relacija *imaDesniDel* poteka med konceptoma *Racun* in *DesniDel*, relacija *imaVsoto* poteka med konceptoma *Racun* in *Vsota*, relacija *imaVrednost* je podatkovna relacija tipa *xsd:int* za razrede *LeviDel, DesniDel, Vsota*.

Predpostavimo, da obstaja storitev za izračun vsote $WS(vsota)$, ki ima defi-

nirano preslikavo $A(vsota)$:

$$
\begin{aligned}
&A(vsota) = \{WS_A, WS_{in}, WS_{out}, WS_{dep}, WS_{res}, M_I, M_O\} \\
&WS_{in}(vsota) = \{LeviDel, DesniDel\} \\
&WS_{out}(vsota) = \{Vsota\} \\
&WS_{dep}(vsota) = \{Racun\} \\
&WS_{res}(vsota) = \{\} \\
&M_I(leviDel) = \{q^{sparql}_{leviDel}, \{"?Y", "/vsota/leviDel"\}\} \\
&M_I(desniDel) = \{q^{sparql}_{desniDel}, \{"?Y", "/vsota/desniDel"\}\} \\
&M_O(Vsota) = \{q^{sparql}_{vsota}\}, \{"?vsota", "/ : vsotaResponse/return"\}
\end{aligned}
\tag{3.11}
$$

Povpraševanja $q^{sparql}_{leviDel}$, $q^{sparql}_{desniDel}$, q^{sparql}_{vsota}, ki so del zgornje preslikave, so podana v izpisih 3.1, 3.2 in 3.3

```
PREFIX calc: <http://ii.uni-mb.si/osii/calculator.owl#>
SELECT ?Y WHERE {
?mainConcept calc:imaLeviDel ?X .
?X calc:imaVrednost ?Y }
```

Izpis 3.1: Povpraševanje $q^{sparql}_{leviDel}$

```
PREFIX calc: <http://ii.uni-mb.si/osii/calculator.owl#>
SELECT ?Y WHERE {
?mainConcept calc:imaDesniDel ?X .
?X calc:imaVrednost ?Y }
```

Izpis 3.2: Povpraševanje $q^{sparql}_{desniDel}$

```
PREFIX calc: <http://ii.uni-mb.si/osii/calculator.owl#>
SELECT ?vsota WHERE {
?mainConcept calc:imaVsoto ?X .
?X calc:imaVrednost ?vsota }
```

Izpis 3.3: Povpraševanje q^{sparql}_{vsota}

Racun	*imaLeviDel→imaVrednost*	*imaDesniDel→imaVrednost*	*izracunaj*
Racun_3+5	3	5	true
Racun_1+2	1	2	false

Tabela 3.1: Vsebina podatkovne hrambe primera

Vsebina ontologije oz. podatkovne hrambe je definirana v tabeli 3.1. Privzamimo, da imamo povpraševanja q_1 (izpis 3.4) in q_2(izpis 3.5). Obe povpraševanji povprašujeta po vsoti nekega računa. Osnovna razlika je, da povpraševanje q_1 ne vsebuje operandov računa, te je potrebno pridobiti iz lokalne podatkovne hrambe (tabela 3.1). Pri povpraševanju q_2 so podatki za izračun podani v povpraševanju samem (prvi operand je 2, drugi pa 4). Sistem tako pri povpraševanju q_1 pridobi vhodne podatke storitve iz lokalne podatkovne hrambe, v povpraševanju q_2 pa iz povpraševanja samega.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX calc: <http://ii.uni-mb.si/osii/calculator.owl#>
SELECT ?racun ?vsota WHERE {
?racun calc:imaVsoto ?Y .
?Y calc:imaVrednost ?vsota.
?racun calc:izracunaj "true"^^xsd:boolean }
```

Izpis 3.4: Povpraševanje q_1

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX calc: <http://ii.uni-mb.si/osii/calculator.owl#>
SELECT ?vsota WHERE {
?racun calc:imaLeviDel ?LD .
?LD calc:imaVrednost 2^^xsd:int .
?racun calc:imaDesniDel ?DD .
?DD calc:imaVrednost 4^^xsd:int .
?racun calc:imaVsoto ?Y .
?Y calc:imaVrednost ?vsota }
```

Izpis 3.5: Povpraševanje q_2

3.3.2 Koraki transformacije SPARQL povpraševanja

V nadaljevanju podajamo korake transformacije SPARQL povpraševanja na osnovi podanega primera. Vhod transformacije predstavlja povpraševanje, zapisano v SPARQL algebri. Za zgoraj podan primer sta povpraševanji v SPARQL algebri naslednji.

```
(project (?racun ?vsota)
  (bgp
    (triple ?racun calc:imaVsoto ?Y)
    (triple ?Y calc:imaVrednost ?vsota)
    (triple ?racun calc:izracunaj true)
  ))
```

Izpis 3.6: Povpraševanje q_1, zapisano v SPARQL algebri

```
(project (?vsota)
  (bgp
    (triple ?racun calc:imaLeviDel ?LD)
    (triple ?LD calc:imaVrednost 2)
    (triple ?racun calc:imaDesniDel ?DD)
    (triple ?DD calc:imaVrednost 4)
    (triple ?racun calc:imaVsoto ?Y)
    (triple ?Y calc:imaVrednost ?vsota)
  ))
```

Izpis 3.7: Povpraševanje q_2, zapisano v SPARQL algebri

3.3.2.1 Izgradnja grafa povpraševanja

Transformacija se izvede za vsako operacijo osnovni graf povpraševanja (operator *bgp*). V kolikor vhodno povpraševanje vsebuje več operatorjev *bgp*, se transformacija izvede za vsakega izmed njih. Prvi korak transformacije je izgradnja grafa povpraševanja na podlagi osnovnega vzorca povpraševanja (operator *bgp*). Vsaka operacija *bgp* je sestavljena iz trojčkov (operator *triple*). Trojčki v sparql algebri so enakovredni trojčkom v RDF [1]. Seznam trojčkov (osnovni

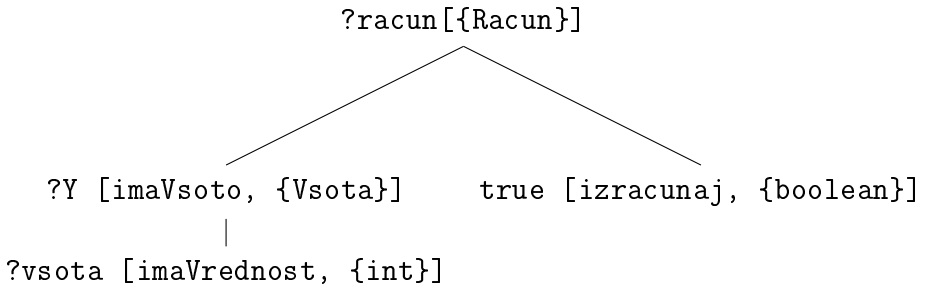
[1]Trojček predstavlja izjavo, ki je sestavljena iz subjekta, predikata in objekta. Subjekt identificira vir za kateri je podana izjava. Predikat določa lastnost, ki jo želimo podati za določen

vzorec povpraševanja v sparql algebri) tako tvori usmerjen graf, kjer subjekti in objekti tvorijo vozlišča grafa, predikati pa tvorijo povezave med vozlišči.

Algoritem, ki ga predstavljamo v nadaljevanju, ne podpira transformacije povpraševanj, kjer so v osnovnem grafu povpraševanja (*bgp*) cikli. Z nadaljnjim raziskovalnim delom je možno razširiti algoritem za transformacijo, tako da razširjeni algoritmi na osnovi hevrističnih lastnosti podatkovnega modela odpravijo cikle v grafu povpraševanja.

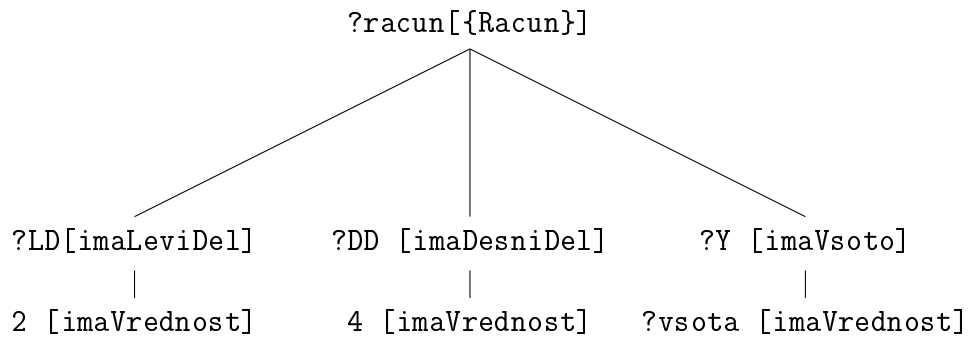
OSII predvideva gradnjo grafa povpraševanja na osnovi seznama trojčkov (osnovi vzorec povpraševanja). Za vsako vozlišče v grafu, sistem pridobi še tip vozlišča na osnovi podanega vprašanja ter na osnovi sklepanja nad globalno ontologijo. Slika 3.4 prikazuje graf povpraševanja za povpraševanje q_1. Podatki o predikatu, ki povezuje dve vozlišči, so podani v oglatih oklepajih ciljnega vozlišča. Podatki o tipu vozlišča pa so podani v zavitih oklepajih.

Iz slike 3.4 je tako razvidno, da sta vozlišči `?racun` in `?Y` povezani s predikatom `imaVsoto`. Tip vozlišča `?racun` je `Racun`, tip vozlisca `?Y` pa je `Vsota`. Na primeru vidimo, da je sistem samostojno pridobil podatke o tipu vseh vozlišč na osnovi aksiomov, podanih v globalni shemi. Na sliki 3.5 smo zaradi optimizacije prostora izpustili tipe vozlišč. Vozlišče *?LD* je tipa *LeviDel*, vozlišče *?DD* je tipa *DesniDel*, vozlišče *?Y* je tipa *Vsota*, vozlišče *?vsota* pa je tipa *xsd:int*.



Slika 3.4: Graf povpraševanja q_1

subjekt, objekt pa določa vrednost lastnosti. Izjavo "nebo je modre barve"lahko predstavimo z naslednjim trojčkom *("nebo", "ima barvo", "modra")*.



Slika 3.5: Graf povpraševanja q_2

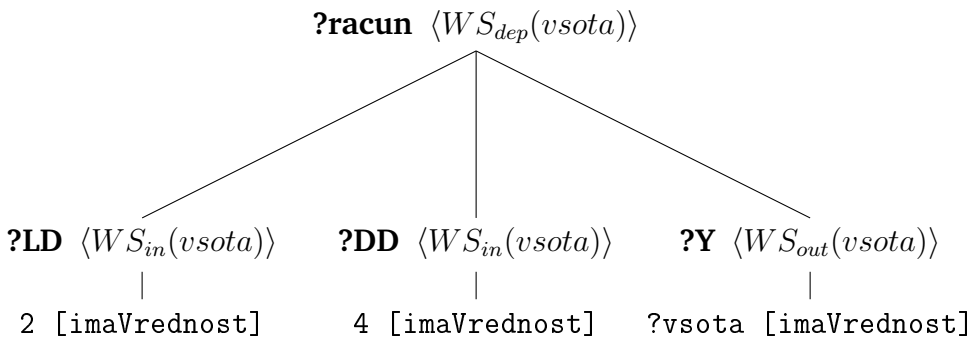
3.3.2.2 Identifikacija potencialnih storitev

Na osnovi grafa povpraševanja sistem za vsako spremenljivko v grafu preveri, ali obstaja storitev, ki nudi tako vrsto podatka. Za vsako vozlišče sistem preveri, ali obstaja storitev, ki vrača ali predikat vozlišča, ali pa tip podatkov, ki je enak tipu vozlišča.

V zgoraj podanem primeru sistem za vozlišče *?Y* odkrije potencialno storitev $WS(vsota)$, saj je vozlišče *?Y* tipa *Vsota*, storitev $WS(vsota)$ pa vrača podatke tipa *Vsota*.

3.3.2.3 Identifikacija vhodov, izhodov in odvisnosti potencialnih storitev v povpraševanju

Po izbiri potencialnih storitev sistem za vsako potencialno storitev v grafu poišče vozlišča, ki se nanašajo na vhod, izhod ter odvisnosti vsake storitve. Označevanje poteka na osnovi definicije preslikave posamezne storitve (glej izraze 3.4 in 3.11). Slika 3.6 prikazuje označen graf za povpraševanje q_2.



Slika 3.6: Označen graf povpraševanja q_2 po identifikaciji storitev

3.3.2.4 Pridobitev vhodnih podatkov na osnovi povpraševanja

Sistem za vsako potencialno storitev, ki ima vhod storitve podan že v samem povpraševanju, pripravi podatke, ki so potrebni za generiranje vhodnih sporočil. Sistem pripravi podatke na osnovi označenega grafa povpraševanja in preslikav M_I (glej izraz 3.4).

Za povpraševanje q_1 in potencialno storitev $WS(vsota)$ nobeden od vhodov ni vsebovan v povpraševanju samem. Zaradi tega mora sistem sam pridobiti vhodne podatke iz podatkov, ki so materializirani v podatkovnih shrambi.

V povpraševanju q_2 sta oba vhoda storitve $WS(vsota)$ podana že v samem povpraševanju. Za povpraševanje q_2 sistem na osnovi opisa storitve (glej izraz 3.11) pridobi naslednje preslikave:

- `/vsota/leviDel` $\rightarrow$ 2: levi del preslikave predstavlja XPath lokacijo podatka v vhodnem sporočilu storitve, medtem ko desni del preslikave predstavlja vrednost podatka. Vrednost sistem pridobi iz grafa na osnovi povpraševanja $q^{sparql}_{leviDel}$ (glej izpis 3.1), ki je del preslikave $M_I(leviDel)$. Na osnovi te iste preslikave sistem podatku, ki ga je pridobil s povpraševanjem $q^{sparql}_{leviDel}$, določi lokacijo v vhodnem sporočilu.

- `/vsota/desniDel` $\rightarrow$ 4: sistem analogno kot za zgornjo preslikavo pridobi tudi to preslikavo ob uporabi povpraševanja $q^{sparql}_{desniDel}$ (glej izpis 3.2), ki je del preslikave $M_I(desniDel)$.

3.3.2.5 Izbira storitev

Za vsako vozlišče v grafu povpraševanja, ki ima definirane potencialne storitve, sistem izbere eno ali več storitev. Sistem izbira storitve glede na prioriteto. Pri določitvi prioritete sistem upošteva naslednja merila:

- $vhodNiStoritev$: merilo ima vrednost `true`, v kolikor za pridobitev vhoda storitve ni potrebno prožiti nobene druge storitve, v nasprotnem primeru dobi merilo vrednost `false`.
- $odvisnostVPovprasevanju$: merilo ima vrednost `true`, v kolikor je razred, ki predstavlja odvisnost storitve, vsebovan v grafu povpraševanja, ali v kolikor storitev nima definirane odvisnosti. V takih primerih sklepamo, da je storitev semantično relevantna za podano povpraševanje, zaradi tega dobi višjo prioriteto.
- $vhodDelnoPokrit$: merilo ima vrednost `true`, v kolikor je vsaj en vhodni podatek storitve pridobljen z vhodnim povpraševanjem. Tudi za takšne primere sklepamo, da ima takšna storitev večjo relevantnost kot storitev, ki ima vrednost tega merila `false`.

Prioritete za izbiro storitev so sledeče:

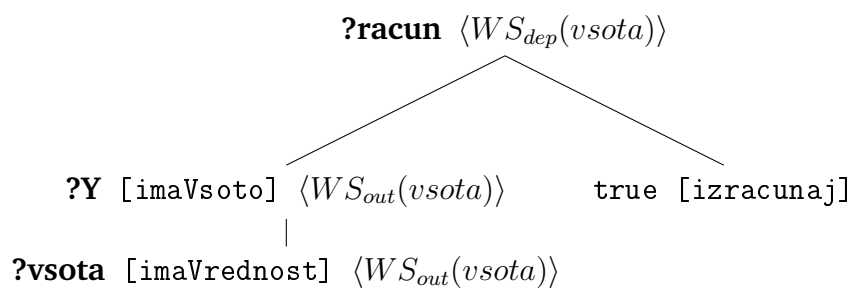
1. Vhod storitve je eksplicitno v celoti podan v vhodnem povpraševanju (primer q_2). To pomeni, da ni potrebno pridobivati podatkov za proženje storitev iz modela in je tako optimalna izbira.
2. Vhod storitve je podan implicitno s vhodnim povpraševanjem. Pridobljen je v času izvajanja povpraševanja, kot stranski rezultat povpraševanja, ki ga je podal uporabnik. Sistemu ni potrebno izvesti novih povpraševanj za pridobitev vhodnih podatkov storitve.
3. $\langle vhodNiStoritev \wedge odvisnostVPovprasevanju \wedge vhodDelnoPokrit \rangle$
4. $\langle vhodNiStoritev \wedge odvisnostVPovprasevanju \rangle$
5. $\langle vhodNiStoritev \wedge vhodDelnoPokrit \rangle$

6. $\langle vhodNiStoritev \rangle$
7. $\langle odvisnostVPovprasevanju \wedge vhodDelnoPokrit \rangle$
8. $\langle odvisnostVPovprasevanju \rangle$
9. $\langle vhodDelnoPokrit \rangle$
10. Vse ostale storitve, ki ne zadostujejo zgornjim pogojem

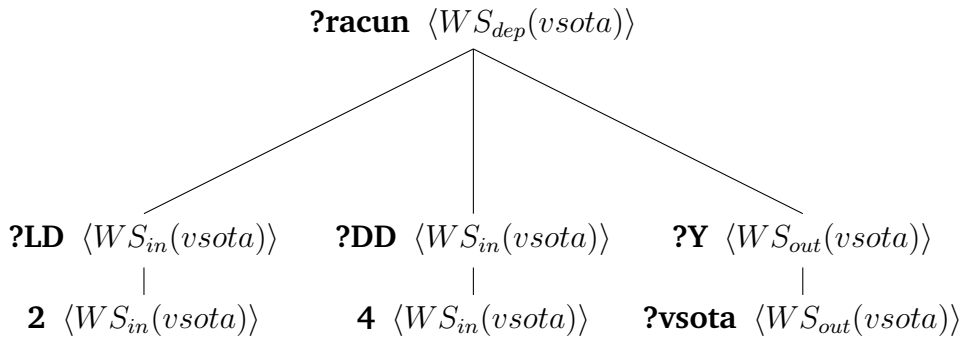
Sistem izbere storitev ali množico storitve z najvišjo prioriteto. Če za posamezno prioriteto obstaja več alternativnih storitev ali množic storitev z enako prioriteto, potem sistem izbere tisto, za katero je potrebno pridobiti najmanj vhodnih podatkov. Izjema je prioriteta 1, katera ima pridobljene že vse vhodne podatke. V takšnem primeru, sistem izbere prvo potencialno storitev.

3.3.2.6 Identifikacija vhodov in izhodov izbranih storitev v povpraševanju

Vhod oziroma izhod storitve lahko v grafu povpraševanja obsega več vozlišč. V 3. koraku (glej poglavje 3.3.2.3) so bila označena samo začetna vozlišča za posamezni vhod na osnovi potencialnih storitev. V tem koraku sistem na osnovi izbranih storitev označi vsa vozlišča, ki se nanašajo na vhod ali izhod. V podanem primeru so tako vključena tudi vozlišča s predikatom *imaVrednost*, ki niso bila označena v predhodnem koraku. Na osnovi teh oznak sistem vozlišča odstrani iz povpraševanja. Sliki 3.7 in 3.8 prikazujeta označena vozlišča za vhodni povpraševanji q_1 in q_2.



Slika 3.7: Označen graf povpraševanja q_1 po identifikaciji vhodov in izhodov



Slika 3.8: Označen graf povpraševanja q_2 po identifikaciji vhodov in izhodov

3.3.2.7 Izdelava delovnega toka povpraševanja

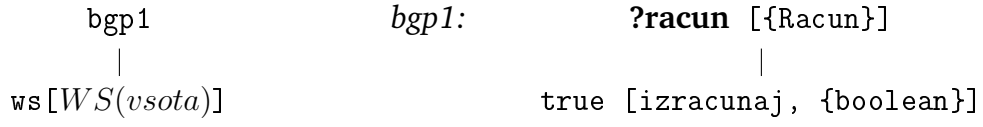
V tem koraku sistem izdela načrt izvajanja povpraševanja. Na osnovi označenega grafa povpraševanja sistem izdela drevo izvajanja. Drevo izvajanja predstavlja delovni tok proženja storitev. Sistem razbije graf povpraševanja na več manjših, med seboj odvisnih, podgrafov. Ti podgrafi predstavljajo osnovne gradnike drevesa izvajanja. Drevo izvajanja je sestavljeno iz dveh tipov vozlišč, in sicer iz osnovnega vzorca (BGP) in iz klicev storitev. Prvi tip vozlišč se kasneje pretvori v operacijo *bgp*, drugi pa v novo operacijo *web_service*. Skladno s tem prvi tip vozlišča predstavlja povpraševanje nad materializiranimi podatki v RDF podatkovni hrambi, drugi tip vozlišča pa predstavlja pridobivanje podatkov iz storitev. Osnovna naloga tega koraka je, da razbije osnovno povpraševanja na povpraševanja po RDF hrambi in na povpraševanja po storitvah ter da naredi načrt izvajanja teh povpraševanj. Vozlišča je možno med seboj poljubno gnezditi neodvisno od tipa. Gnezdenje poteka z binarnimi sparql operatorji *union* in *join*. Razbitje grafa povpraševanja poteka po sledečem postopku:

- Razbij grafa poteko tako, da sistem odcepi vsako vozlišče v grafu, ki predstavlja klic storitve.
- Za vsak graf, ki ima korensko vozlišče storitev, sistem odcepi vse podgrafe, ki niso vhod ali izhod storitve.
- Iz vseh podgrafov odstrani vsa vozlišča, ki predstavljajo vhod storitve, vendar samo v primeru, če ima ta storitev vhod v celoti pridobljen.

- Iz podgrafov premakni vsa preostala vozlišča, ki predstavljajo vhod storitve, h klicu storitve.

Ko je graf razbit na podgrafe tipa *bgp* ali *web_service*, sistem poišče odvisnosti med podgrafi na osnovi referenciranja spremenljivk v povpraševanju ter na osnovi definicij preslikav storitve (predvsem vhod, izhod ter odvisnost storitve). Na osnovi odvisnosti med podgrafi sistem zgradi drevo izvajanja, kjer so elementi posamezni podgrafi.

Na osnovi tega postopka pridobimo za povpraševanje q_1 naslednje drevo:



Slika 3.9: Drevo izvajanja za q_1 in pripadajoči podgraf *bgp1*

Vozlišče *ws[$WS(vsota)$]* predstavlja klic spletne storitve $WS(vsota)$, vozlišče *bgp1* pa predstavlja preostala vozlišča prvotnega povpraševanja, ki niso vhod ali izhod storitev. Storitev $WS(vsota)$ je odvisna od vozlišča *?racun* (določeno na osnovi grafa povpraševanja), zaradi tega je v drevesu izvajanja vozlišče *ws[$WS(vsota)$]* odvisno od vozlišča *bgp1*. Skladno s tem je vozlišče *bgp1* korensko vozlišče. Za izvajanje povpraševanja to pomeni, da bo pogon povpraševanja najprej izvedel povpraševanje na osnovi vozlišča *bgp1*. Rezultati tega povpraševanja pa bodo služili za pripravo vhodov storitve $WS(vsota)$.

3.3.2.8 Priprava pridobitve vhodnih podatkov iz podatkovnega modela

Vhodne podatke za storitve je mogoče pridobiti na dva načina: iz povpraševanja samega ali iz podatkov, ki so materializirani v RDF podatkovni hrambi. Sistem v tej fazi pridobi vhodne podatke za izbrane storitve, ki še nimajo pridobljenih vseh vhodov. Priprava vhodnih podatkov poteka na podlagi povpraševanj, definiranih v preslikavah M_I.

V kontekstu podanega primera ima povpraševanje q_2 že v celoti zagotovljen vhod storitve s povpraševanjem samim (operanda 2 in 4). Zaradi tega, v tem koraku za q_2 ne pride do obdelave. Pri povpraševanju q_1 je potrebno za oba

vhoda pridobiti vhodne podatke. Na osnovi preslikave storitve, sistem generira naslednji osnovni vzorec grafa, ki ga uporabi pogon povpraševanja v času izvajanja za pridobitev vhoda storitve:

```
(bgp
   (triple ?racun calc:imaLeviDel ?WS_LeviDel_X)
   (triple ?WS_LeviDel_X calc:imaVrednost ?WS_LeviDel)
   (triple ?racun calc:imaDesniDel ?WS_DesniDel_X)
   (triple ?WS_DesniDel_X calc:imaVrednost ?WS_DesniDel)
)
```

Izpis 3.8: Osnovni vzorec grafa (BGP) za pridobitev vhoda storitve $WS(vsota)$ povpraševanja q_1

Na osnovi zgornjega povpraševanja sistem pridobi operanda 3 in 5 (glej tabelo 3.1). Zgornje povpraševanje ne vrne operandov 1 in 2 zaradi tega, ker sistem predhodno izbere račune, ki imajo vrednost podatkovne lastnosti `izracunaj` enako `true` (na osnovi podgrafa *bgp1*, slika 3.9).

3.3.2.9 Transformacija povpraševanja

Sistem na osnovi drevesa izvajanja kreira novo povpraševanje v sparql algebri. Med seboj odvisna vozlišča, poveže z uporabo binarnega združevalnega operatorja *join*. Ta operator zagotavlja, da se bo najprej izvedla operacija podana kot prvi argument operatorja, nakar se bo z uporabo rezultatov prve operacije izvedla še druga operacija. Med seboj neodvisna vozlišča, sistem poveže z uporabo binarnega združevalnega operatorja *union*. Ta ločeno izvede obe operaciji in nato izvede agregacijo rezultatov. Operator *union* se tudi uporabi, ko ima posamezno vozlišče izbranih več storitev. Oba operatorja sta del SPARQL algebre in jih podpira vsak SPARQL pogon povpraševanja.

Rezultata transformacije za povpraševanji q_1 in q_2 sta prikazana v izpisih 3.9 in 3.10. S tem je proces transformacije povpraševanja končan. Sistem preda transformirano povpraševanje pogonu izvajanja povpraševanj. Povpraševanje q_2 ima vhod pridobljen v celoti, zato ni potrebno pridobivati podatkov iz RDF hrambe ali kakšne druge storitve. Tako transformirano povpraševanje q_2 vsebuje zgolj klic storitve.

```
(project (?racun ?vsota)
  (join
    (bgp (triple ?racun calc:izracunaj true))
    (web_service  CalculatorService{[?vsota]}
      Input:
      (bgp
        (triple ?racun calc:imaLeviDel ?WS_LeviDel_X)
        (triple ?WS_LeviDel_X calc:imaVrednost ?WS_LeviDel)
        (triple ?racun calc:imaDesniDel ?WS_DesniDel_X)
        (triple ?WS_DesniDel_X calc:imaVrednost ?WS_DesniDel)
      )
    )
  ))
```

Izpis 3.9: Transformirano povpraševanje q_1

```
(project (?vsota)
  (web_service  CalculatorService{[?vsota]})
    Input: 2, 4
)
```

Izpis 3.10: Transformirano povpraševanje q_2

3.3.3 Izvajanje transformiranih povpraševanj

Izvajanje transformiranih povpraševanj poteka na povsem enak način kot izvajanje klasičnih SPARQL povpraševanj. Edina zahteva je, da ima pogon za izvajanje povpraševanj implementiran operator *web_service*. Implementacija tega operatorja je precej enostavna. Operator dobi kot vhod:

- celotno preslikavo storitve (v podanem primeru dobi operator preslikavo $A(vsota)$, glej izrek 3.11),
- vhodne podatke, pridobljene iz vhodnega povpraševanja (glej poglavje 3.3.2.4), v kolikor so na voljo, in
- povpraševanje za pridobitev vhodnih podatkov.

Proženje povpraševanja q_2 poteka precej trivialno, saj vsebuje samo klic operacije *web_service*, ki že ima pridobljene vhodne podatke. Tako se dejansko izvede samo klice storitve $WS(vsota)$.

Proženje povpraševanja q_1 je bolj kompleksno. Glede na SPARQL algebro, se najprej izvede naslednja operacija:

```
(bgp (triple ?racun calc:izracunaj true)
```

Kot rezultat te operacije pogon izvajanja povpraševanj veže prosto spremenljivko *?racun* na vrednost *Racun_3+5* (glej definicijo vsebine RDF hrambe v tabeli 3.1). V kolikor bi povpraševanje vrnilo več odgovorov, bi pogon izvajanja na prosto spremenljivko *?racun* vezal seznam vrednosti.

Kot naslednja se izvede operacija *web_service*. Ker v tem primeru vsi podatki še niso pridobljeni, operacija najprej izvede povpraševanje za pridobitev vhoda:

```
(bgp
  (triple ?racun calc:imaLeviDel ?WS_LeviDel_X)
  (triple ?WS_LeviDel_X calc:imaVrednost ?WS_LeviDel)
  (triple ?racun calc:imaDesniDel ?WS_DesniDel_X)
  (triple ?WS_DesniDel_X calc:imaVrednost ?WS_DesniDel)
)
```

Zgoraj podano povpraševanje se izvede za vsako vrednost, ki je vezana na spremenljivko *?racun* (v našem primeru zgolj za podatek *Racun_3+5*). Izvajanje povpraševanja za pridobitev vhoda veže proste spremenljivke s pridobljenimi vrednostmi. Te vrednosti nato sistem uporabi pri klicanju storitev. Sistem za vsak rezultat, ki ga pridobi s tem povpraševanjem, izvede proženje spletne storitve. V našem primeru se to zgodi samo enkrat za vhodne podatke storitve: *LeviDel=3, DesniDel=5*. Operacija *web_service* veže rezultate storitev na spremenljivko *?vsota* (glej drevo povpraševanja, slika 3.7), kar se na koncu uporabi pri projekciji rezultatov (operacija *project*).

3.4 Sklepanje

Klasični pogoni za sklepanje opravljajo sklepanje nad izjavnim okvirjem ontologije. Model, ki smo ga predstavili v tej doktorski disertaciji omogoča integracijo

podatkov, ki jih ponujajo storitve. Izjavni okvir v našem modelu tako predstavljajo podatki, ki jih vračajo storitve. Klasični pogoni sklepanja zaradi tega niso primerni, saj bi, v kolikor bi hoteli izvesti sklepanje, morali predhodno pridobiti vse podatke, ki jih ponujajo storitve.

Pristop, ki smo ga razvili, opravi sklepanje na podlagi semantičnih opisov storitev (preslikav) v času izvajanja transformacije povpraševanja. Sklepanje se v našem modelu izvaja v fazi identifikacije potencialnih storitev (poglavje 3.3.2.2). V tej fazi sistem za vsako spremenljivko v vzorcu povpraševanja poišče storitve, ki nudijo takšno vrsto podatkov. V okviru odkrivanja storitev izvedemo sklepanje na osnovi ontologije. Za primer privzemimo, da imamo podano sledečo OWL 2 QL ontologijo:

$$\begin{aligned} Profesor &\sqsubseteq OsebjeFakultete \\ Docent &\sqsubseteq Profesor \\ IzredniProfesor &\sqsubseteq Profesor \\ RedniProfesor &\sqsubseteq Profesor \end{aligned}$$

ter naslednje definicije izhodov storitev:

$$\begin{aligned} WS_{out}(DocentService) &= \{Docent\} \\ WS_{out}(IzredniProfesorService) &= \{IzredniProfesor\} \\ WS_{out}(RedniProfesorService) &= \{RedniProfesor\} \end{aligned}$$

V kolikor sistem prejme povpraševanje, ki vsebuje povpraševanje po podatkih tipa *Profesor*, potem bo sistem na osnovi razredne hierarhije sklepal, da so potencialne storitve *DocentService, IzredniProfesorService* in *RedniProfesorService*. OSII omogoča sklepanje na podmnožici OWL 2 QL profila [41]. Podpira vse aksiome tega profila, razen naslednjih aksiomov:

- **Inverzna objektna lastnost** se uporablja za določitev inverzne relacije med dvemi različnimi objektnimi lastnostmi, ki imata diametralno nasprotni pomen (npr. relacija *imaDiplomoFakultete* je inverzna relaciji *imaDiplomanta*). Proces sklepanja na osnovi podane relacije vstavi novo izjavo, ki

vsebuje inverzno relacijo.

- **Eksistenčni kvantifikator** se uporablja za specifikacijo razreda na osnovi obstoja določene podatkovne lastnosti. Za primer privzemimo, da obstaja razred *Dekan*, ki ima definirano eksistenčno omejitev za lastnost *jeVodja-Fakultete*. V kolikor ima posamezna oseba podano vrednost za lastnost *jeVodjaFakultete*, potem proces sklepanja določi to osebo kot pripadnika razreda *Dekan*.

- **Presek** se uporablja za specifikacijo razreda kot preseka dveh drugih razredov.

- **Negacija** se uporablja za specifikacijo razreda kot komplement nekega razreda (npr. razred *DodiplomskiStudent* je komplementaren razredu *Podpiplomski-Student*).

OSII model v svoji zasnovi omogoča podporo za celotni OWL 2 QL profil. Pri razvoju algoritmov in prototipa, ki so predstavljeni v tej doktorski disertaciji, smo se omejili zgolj na podmnožico, ki ne vsebuje zgoraj podanih aksiomov. Tako smo se odločili ker smo mnenja, da zgornji aksiomi niso ključni za področje integracije heterogenih podatkovnih virov. Hkrati ima lahko realizacija omenjenih aksiomov znaten negativni učinek na delovanje sistema v kolikor je globalna ontologija slabo zasnovana. Predstavljene algoritme in razvit prototip je možno sorazmerno enostavno razširiti tako, da podpirajo zgornje aksiome. Izločitev zgornjih aksiomov nima pozitivnega vpliva na učinkovitost prototipa v evalvaciji učinkovitosti, ki je predstavljena v nadaljevanju (poglavje 4).

3.5 Algoritmi

V tem odseku podajamo algoritme ključnih komponent modela za integracijo podatkovnih virov s semantično označenimi storitvami. Ključna komponenta modela je transformator povpraševanj, ki na osnovi preslikav generira načrt izvajanja povpraševanja v sparql algebri. Algoritem 3.1 podaja algoritem za transformacijo povpraševanj zapisan v psevdokodu. Algoritem 3.2 definira

funkcijo *generirajPovpraševanje*, ki je del algoritma 3.1. Algoritem 3.2 na osnovi drevesa izvajanja generira končno sparql povpraševanje. Algoritem 3.3 podaja psevdokod za operacijo web_service.

Algoritem 3.1 TransformirajSparql($vhodnoPovprasevanje$)
Algoritem

$graf :=$ dobiGraf($vhodnoPovprasevanje$);
$vozliscaStoritev := null$;
comment: Pridobi potencialne storitve;
for $vozlisce$ in $graf$ do
    $ps :=$ dobiPotencialneStoritve($vozlisce.tip$);
    $vozlisce.potencialneStoritve := ps$;
    if $ps! = null$ then
        $vozliscaStoritev := vozliscaStoritev + vozlisce$;
    fi
od
označiGraf($graf$);
pridobiVhodeIzPovpraševanja($vozliscaStoritev$);
comment: Izbira storitev;
for $vozlisce$ in $vozliscaStoritev$ do
    izračunajPrioritete($vozlisce$);
    izberiStoritve($vozlisce$);
od
označiGrafNaIzbranihStoritvah($graf$);
comment: Izdelava delovnega toka;
$seznamGrafov := graf$;
for $vozlisce$ in $vozliscaStoritev$ do
    $podgraf :=$ odcepiGraf($vozlisce$);
    $seznamPodgrafov :=$ odcepiVozlišcaNeodvisnaOdStoritve($podgraf$);
    $seznamGrafov := seznamGrafov + podgraf + seznamPodgrafov$;
od
for $podgraf$ in $seznamGrafov$ do
    for $vozlisce$ in $podgraf$ do
        $storitev :=$ poišciStoritevZVhodom($vozlisce$);
        if $storitev$.vhodPridobljenVCeloti() then odstrani($vozlisce$); fi
        if $storitev$.vhodPridobljenDelno() then premakni($vozlisce, storitev.graf$); fi
    od
od
$drevoIzvajanja =$ dolociOdvisnostiPodgrafov($seznamGrafov$);
generirajPovpraševanje($drevoIzvajanja.korenskoVozlisce$);

Algoritem 3.2 generirajPovpraševanje($vozlisce$)

```
proc generirajPovpraševanje(v) ≡
  p := null;
  for v_otrok in v.otroci do
      np = generirajPovpraševanje(v_otrok)
      if np ∧ ¬p then p = np; fi
      if np ∧ p then p = ustvariOp(join, p, np); fi
  od
  if jeStoritev(v) then
                          np = ustvariOp(web_service, v);
                          if np ∧ ¬p then p = np; fi
                          if np ∧ p ∧ jeNeodvisen(v) then p = ustvariOp(union, p, np); fi
                          if np ∧ p ∧ jeOdvisen(v) then p = ustvariOp(join, p, np); fi
  fi
  if niStoritev(v) then
                          np = ustvariOp(bgp, v);
                          if np ∧ ¬p = null then p = np; fi
                          if np ∧ p! = null then p = ustvariOp(join, p, np); fi
  fi
  return p;
```

Algoritem 3.3 web_service($preslikava, vhod, povprasevanje$)

```
proc web_service(p, v, q) ≡
  if v = null then
                  v = izvediPovpraševanje(q)
  fi
  vhod = transformiraj(v, p.M_I);
  i = prožiStoritev(p.WS_A, vhod);
  izhod = transformiraj(i, p.M_O);
  return izhod;
```

Poglavje 4

Ovrednotenje učinkovitosti sistemov za integracijo podatkov na osnovi ontologij

4.1 Model za ocenjevanje učinkovitosti

Področje integracije podatkovnih virov na osnovi ontologij je sorazmerno nov pristop. Pri pregledu literature nismo našli pristopa, ki bi omogočal celovito primerjavo sistemov za integracijo podatkovnih virov na osnovi ontologij. Iz tega razloga smo na osnovi obstoječih znanstvenih spoznanj razvili dvonivojski model za ocenjevanje učinkovitosti, ki temelji na kvalitativni in kvantitativni evalvaciji.

Izbira posameznega sistema oziroma pristopa za integracijo podatkovnih virov je odvisna od specifičnih zahtev posameznega projekta. Zaradi tega ni možno posplošiti ugotovitev kvalitativne analize za poljuben problem integracije heterogenih podatkovnih virov. Vsak projekt integracije heterogenih podatkovnih virov predstavlja kompromis med funkcionalnostmi sistema (kvalitativna evalvacija) in učinkovitostjo sistema (kvantitativna evalvacija). V tem okviru služi kvalitativna evalvacija kot ogrodje za izvedbo primerjalne analize pristopov in sistemov za integracijo heterogenih podatkovnih virov na podlagi kvalitativnih meril. Kvantitativna evalvacija, kot jo predstavimo v nadaljevanju, služi za

kvantitativno ovrednotenje učinkovitosti posameznih sistemov za integracijo heterogenih podatkovnih virov na osnovi ontologij.

4.1.1 Kvalitativna evalvacija

Seng in Kong sta v [74] predstavila primerjalno analizo pristopov za integracijo podatkovnih virov. V ta namen sta na osnovi obstoječih pristopov iz področja integracije heterogenih podatkovnih virov definirala devet kriterijev za primerjavo sistemov. Dodali smo dva kriterija, ki sta specifična za sisteme za integracijo podatkovnih virov na osnovi ontologij. Model za kvalitativno analizo sestavljajo naslednja merila:

- **Leto objave:** leto, v katerem je bil predstavljen posamezen pristop.
- **Raziskovalna ekipa:** raziskovalna ekipa, ki je razvila pristop.
- **Heterogenost informacijskih virov:** nekateri sistemi omogočajo integracijo iz raznolikih/heterogenih informacijskih virov neodvisno od spodaj ležeče tehnologije. Ta kriterij opisuje stopnjo heterogenosti podatkovnih virov, ki jih je možno integrirati s pristopom.
- **Podatkovni model:** model, ki se uporablja za specifikacijo sheme.
- **Strategija integracije:** strategija, ki jo uporablja pristop za integracijo podatkovnih virov (LAV oz. GAV - glej poglavje 2.1).
- **Povpraševalni jezik:** jezik, ki se uporablja za povpraševanje po integriranem podatkovnem modelu.
- **Pristop povpraševanja:** pristop obdelave in izvajanja povpraševanja.
- **Dodajanje/odstranjevanje virov:** kriterij opisuje, na kakšen način je možno dodati nov ali odstraniti obstoječ vir.
- **Nivo sklepanja:** nivo sklepanja, ki ga zagotavlja pristop.
- **Nivo federacije:** kriterij opisuje način, s katerim pristop obravnava problematiko federacije podatkov iz različnih virov.

Tabeli 4.1 in 4.2 vsebujeta primerjavo obstoječih pristopov, ki smo jih povzeli v poglavju 2.4, vključno z našim pristopom (OSII), ki smo ga predstavili v prejšnjem poglavju.

Na osnovi kvalitativne primerjave lahko ugotovimo, da samo pristop, ki smo ga razvili, omogoča sklepanje nad heterogenimi podatkovnimi viri. Pristopa OI2 in OI3 prav tako omogočata sklepanje vendar ne iz heterogenih podatkovnih virov. Pri pristopu OI2 je potrebno podatke predhodno združiti na nivoju relacijske baze, pri čemer avtorji uporabijo komercialna orodja za federacijo relacijskih baz. Pristop OI3 predvideva uporabo podatkovnega skladišča, tako je potrebno vse podatke izvoziti iz lokalnih podatkovnih virov ter jih uvoziti v podatkovno skladišče.

Izmed primerjanih pristopov samo dva pristopa (OI1 in OSII) uporabljata LAV pristop k integraciji podatkovnih virov. LAV pristop k integraciji je fleksibilnejši od pristopov GAV ali podatkovnega skladišča. Slabost LAV pristopa je kompleksnejše obdelovanje povpraševanj. Zaradi tega je potrebno razviti kompleksnejše algoritme. OSII podpira naprednejšo federacijo podatkovnih virov kot drugi pristopi. Federacija pri OSII poteka na osnovi zmožnosti in omejitev posameznih podatkovnih virov.

Glede na rezultate kvalitativne primerjave lahko sklenemo, da OSII predstavlja napreden LAV pristop k integraciji heterogenih podatkovnih virov na osnovi ontologij, ki omogoča sklepanje in federacijo. Nobeden od primerjanih pristopov ne podpira vseh lastnosti, ki jih podpira OSII. OSII je prav tako edini izmed pristopov, ki ima podano kvantitativno ovrednotenje učinkovitosti.

4.1.2 Kvantitativna evalvacija

Pri pregledu literature nismo odkrili pristopa, ki bi omogočal kvantitativno analizo sistemov za integracijo podatkov na nivoju ontologij. Zaradi tega smo analizirali pristope za kvantitativno evalvacijo podatkovnih shramb, ki temeljijo na tehnologijah semantičnega spleta. Izbrali smo pristop, ki je najustreznejši za naše raziskovalno področje, ter ga razširili glede na specifične značilnosti sistemov za integracijo podatkovnih virov na osnovi ontologij.

Področje kvantitativne analize podatkovnih shramb na osnovi tehnologij

Kriterij	OI1 - Levy	OI2 - Calvanese	OI3 - Sahoo	OI4 - Nachouki	OI5 - Verstichtel
Leto objave	1996	2008	2008	2011	2011
Raziskovalna ekipa	AT&T, Bell Lab., Univerza Stanford	Free University of Bolzano, Sapienza Universita di Roma	Wright State University, National Library of Medicine, National Institute on Drug Abuse	Nantes University, Aix-Marseille II University	Ghent University, DeltaRail, Televic
Heterogenost virov	Poljubni vir ovit s programskim vmesnikom	Relacijske podatkovne baze	XML/RDF	XML	RDF
Podatkovni model	Lasten objektno-relacijski model	OWL	OWL	XML	OWL
Strategija integracije	LAV	GAV	Podatkovno skladišče	GAV	GAV
Povpraševalni jezik	Lasten povpraševalni jezik	Epistemic Query Language (EQL)	Sparql	Xquery	Sparql
Pristop povpraševanja	Na osnovi LAV preslikav, ki definirajo zmožnosti podatkovnih virov, naredijo plan izvajanja povpraševanj	V času obdelave povpraševanja izvede sklepanje, nato povpraševanja transformira v SQL povpraševanja	Vsi podatki so v skupni podatkovni hrambi, povpraševanje poteka kot vsako standardno sparql povpraševanje	Uporabnik poda načrt integracije s povpraševanjem	Enostavna agregacija povpraševanj
Dodajanje in odstranjevanje virov	Potrebno je podati opis novega podatkovnega vira	Potrebno spremeniti obstoječe GAV preslikave	Potrebno je uvoziti podatke v globalno podatkovno shrambo	Potrebno opisati vire in preslikave	Potrebno je ustvariti owl:sameAs relacije z globalno ontologijo
Sklepanje	Ne podpira	Podmnožica opisne logike $DL-Lite_A$	Delno OWL-DL sklepanje	Ne podpira	Ne podpira
Federacija	Podpira, ne obravnava podrobno	Ne podpira	Ne podpira	Ne podpira avtomatizirane federacije, temveč jo uporabnik poda deklarativno	Podpira federacijo na osnovi relacij owl:sameAs v ontologiji

Tabela 4.1: Prvi del kvalitativne primerjave pristopov za integracijo podatkovnih virov na osnovi ontologij

Kriterij	OI6 - Cruz in Xiao	OI7 - Seng in Kong	OI8 - Lam	WSII – Grašič
Leto objave	2009	2009	2006	2011
Raziskovalna ekipa	University of Illinois	National Chengchi University, UMC Semiconductor Corp.	Yale University	Univerza v Mariboru
Heterogenost virov	Relacijske baze, XML, RDF	Poljuben vir, ki je ovit s specifičnim vmesnikom, ki ga uvaja pristop	RDF	Poljuben podatkovni vir, do katerega je moč dostopati s spletnimi storitvami
Podatkovni model	RDFS	XML	OWL	OWL
Strategija integracije	GAV	GAV	GAV	LAV
Povpraševalni jezik	RQL, mogoča tudi uporaba Sparql	Xquery	Sparql	Sparql
Pristop povpraševanja	Prevajanje in dekompozicija povpraševanj na osnovi GAV preslikav in nato agregacija rezultatov	Dekompozicija povpraševanj na osnovi GAV preslikav in nato agregacija rezultatov.	Enostavna agregacija rezultatov	Sklepanje v času obdelave povpraševanje in transformacija povpraševanj v sparql algebri na osnovi LAV preslikav
Dodajanje in odstranjevanje virov	Potrebno je kreirati ontologijo novega vira ter narediti preslikavo z globalno ontologijo	Potrebno je razviti ovojnico vira in definirati preslikave	Potrebno je ustvariti owl:sameAs relacije z globalno ontologijo	Potrebno opisati samo nov vir
Sklepanje	Ne podpira	Ne podpira	Ne podpira	Model podpira OWL 2 QL, prototip podpira podmnožico OWL 2 QL
Federacija		Federacija je podana deklarativno z GAV preslikavami	Podpira federacijo na osnovi relacij owl:sameAs v ontologiji	Podpira federacijo, potek federacije je določen deklarativno z zmožnostmi in omejitvami storitev

Tabela 4.2: Drugi del kvalitativne primerjave pristopov za integracijo podatkovnih virov na osnovi ontologij

semantičnega spleta je precej aktivno raziskovalno področje. Najbolj razširjeni in sprejeti pristopi so LUBM, UOBM, BSBM in SP^{2}Bench.

LUBM - Lehigh University Benchmark

LUBM pristop [37] vsebuje ontologijo, orodja za generiranje podatkov ter množico povpraševanj za izvajanje primerjalnih testov. Pristop je namenjen vrednotenju rešitev, ki imajo majhen terminološki okvir (TOkvir) in velik izjavni okvir (IOkvir). Zaradi te lastnosti je primeren za vrednotenje sistemov za integracijo podatkov na nivoju ontologij.

UOBM - University Ontology Benchmark

UOBM predstavlja razširitev LUBM pristopa [57]. Avtorji so LUBM razširili z novo kompleksnejšo ontologijo ter povpraševanji, ki zahtevajo kompleksnejše sklepanje. Na povpraševanja v LUBM ontologiji je mogoče odgovoriti s sklepanjem nad terminološkim okvirjem, medtem ko povpraševanja UOBM zahtevajo tudi sklepanje nad izjavnim okvirjem. Bistvena razlika je, da je povpraševanje nad izjavnim okvirjem bistveno bolj kompleksno in časovno zahtevno, saj je izjavni okvir tipično precej večji od terminološkega okvirja. Sklepanje nad izjavnim okvirjem je koristno za sisteme za upravljanje z znanjem, medtem ko je za sisteme za integracijo heterogenih informacijskih virov manj koristno. Profila OWL2 EL in OWL QL sta bila razvita z namenom zagotovitve učinkovitosti sklepanja pri izvajanju povpraševanj nad OWL ontologijami. Razširitve, ki jih uvaja UOBM, niso kompatibilne z OWL2 EL ali OWL QL.

BSBM - Berlin Sparql Benchmark

Namen pristopa je merjenje učinkovitosti podatkovnih shramb, ki temeljijo na semantičnih tehnologijah in, ki za fizično hrambo podatkov uporabljajo različne tehnologije. Pristop se ne osredotoča toliko na sklepanje, kot se ostali pristopi, temveč bolj na večjo količino med seboj tesno povezanih podatkov. V ta namen so razvili shemo in podatkovno množico, ki bi naj predstavljala čimbolj realen

primer [8]. Podatkovna množica ponazarja podatke, ki jih srečamo v tipični spletni trgovini.

SP^{2}Bench

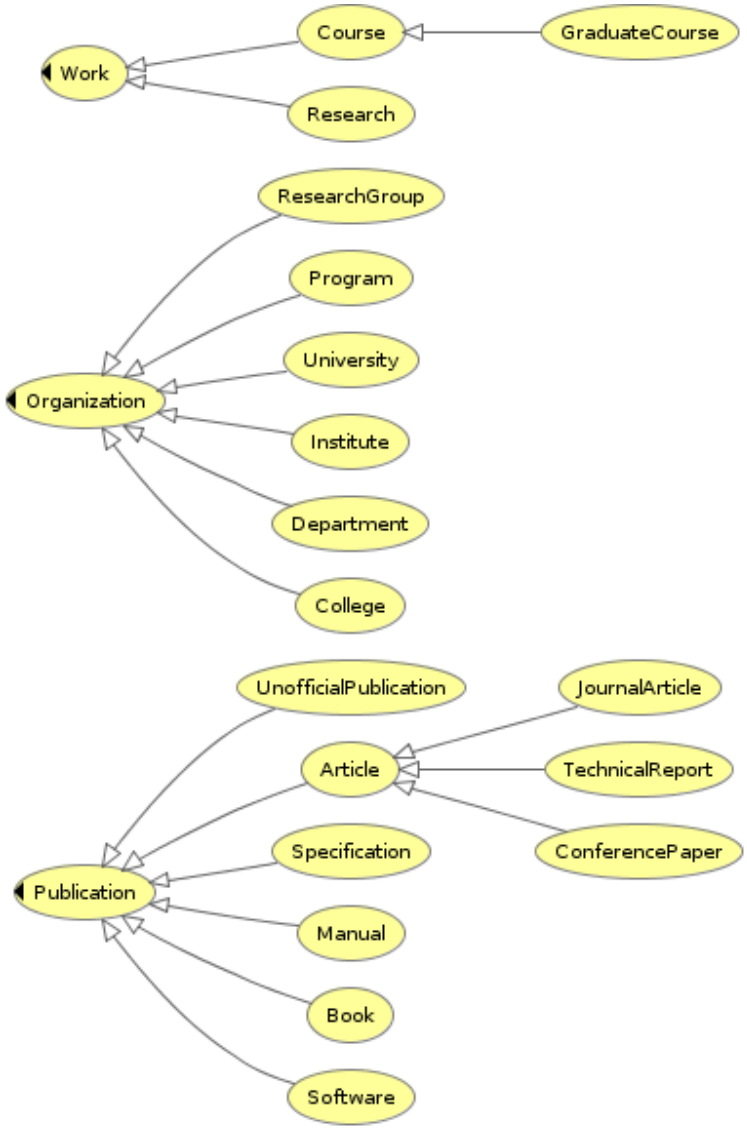
SP^{2}Bench prav tako kot ostali pristopi vsebuje ontologijo, generatorje podatkov ter množico povpraševanj. Avtorji so zasnovali ontologijo skladno z obstoječo bazo podatkov DBLP. Na osnovi tipičnih primerov uporabe te baze podatkov so podali tudi povpraševanja [73].

Glede na ugotovljene karakteristike predstavljenih pristopov, smo za kvantitativno evalvacijo uporabili LUBM pristop, ki je de facto standard [80][76][62][26] za merjenje učinkovitosti podatkovnih shramb, ki temeljijo na tehnologijah semantičnega spleta. V nadaljevanju podajamo podrobnejši opis LUBM ter razširitev, ki omogoča boljšo kvantitativno analizo pristopov, ki ne podpirajo sklepanja.

4.1.2.1 LUBM

LUBM pristop vsebuje ontologijo zapisano v opisni logiki (v jezikih DAML in OWL-DL), generatorje podatkov ter 14 povpraševanj za merjenje učinkovitosti RDF podatkovnih hramb, ki uporabljajo sparql kot povpraševalni jezik[37]. Ontologija je iz domene upravljanja podatkov univerze in vsebuje podatke o fakultetah, študentih, profesorjih, predmetih in publikacijah.

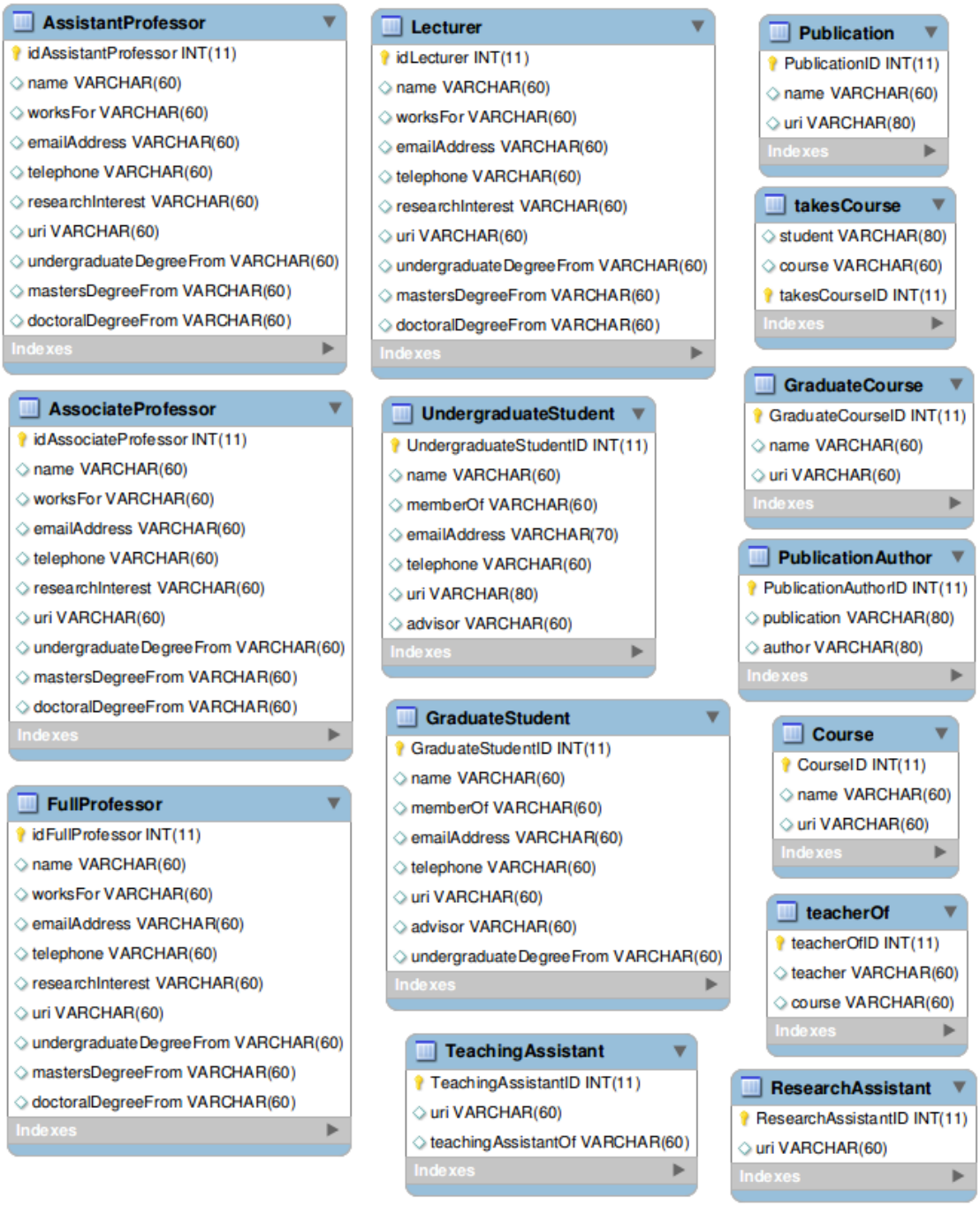
LUBM ontologija je sorazmerno enostavna, vsebuje 43 razredov in 15 lastnosti. 37 razredov je primitivnih, 6 pa jih je definiranih na osnovi aksiomov. Razredi so organizirani v precej kompleksno razredno hierarhijo. Razredno hierarhijo smo zaradi kompleksnosti predstavili na dveh ločenih slikah (sliki 4.1 in 4.2).



Slika 4.1: Prvi del hierarhije LUBM ontologije

Slika 4.2: Drugi del hierarhije LUBM ontologije

LUBM omogoča sintetično generiranje podatkov za poljubno število univerz. V ta namen vsebuje generator podatkov, ki vrne podatke v RDF/XML formatu. Kvantitativno evalvacijo smo izvedli na podatkovni množici za 1, 5, 10 in 20 univerz (označeno kot LUBM1, LUBM5, LUBM10 in LUBM20). Tabela 4.3 prikazuje podatke o podatkovnih množicah, ki smo jih pridobili z generatorjem. Prototip, ki smo ga razvili, kot tudi nekateri sorodni pristopi za integracijo podatkovnih virov uporabljajo za fizično hrambo podatkov relacijske baze. Zaradi tega smo na podlagi RDF grafa, ki ga vrne generator, razvili entitetno-relacijski model (ER model).



Slika 4.3: Stuktura relacijske podatkovne baze, ki vsebuje podatke, pridobljene z LUBM generatorjem

Oznaka	Št. datotek	Velikost (MB)	Št. trojčkov
LUBM1	15	8,6	103.397
LUBM5	93	54,2	646.128
LUBM10	189	110,6	1.316.993
LUBM20	402	234,8	2.782.419

Tabela 4.3: Vsebina podatkovne hrambe primera

ER model smo razvili na osnovi LUBM ontologije in generiranih podatkov tako, da smo za vsak razred, ki ima podatke neposredno podane v RDF datotekah, ustvarili svojo entiteto (npr. *FullProfessor, UndergraduateStudent, Course*). Za vsako objektno relacijo v ontologiji, ki vsebuje izjave v RDF datotekah, pa smo ustvarili relacijo med entitetami. V kolikor gre za relacijo ena proti mnogo, smo relacijo dodali k entiteti v obliki tujega ključa (npr. *FullProfessor.worksFor, FullProfessor.doctoralDegreeFrom*). V primeru, da gre za relacijo mnogo proti mnogo pa smo dodali novo entiteto (npr. *teacherOf, takesCourse*).

Na osnovi pridobljenega modela smo kreirali relacijsko podatkovno bazo, v katero smo uvozili podatke iz RDF datotek. Podatkov nismo transformirali, uvozili smo jih natančno takšne, kot so eksplicitno zapisani v generiranih RDF datotekah. Ontologijo smo uporabili zgolj za kreiranje podatkovne strukture, ne pa za transformacijo podatkov. Podatki v RDF modelu so identificirani na osnovi URI naslovov, medtem ko so v relacijskih bazah v večini primerov identificirani na osnovi avtomatično generiranih celih števil, ki so enostavnejša za procesiranje.

Zaradi zagotavljanja primerljivosti smo v relacijski bazi ohranili identifikatorje v URI obliki, kljub temu, da iz vidika modeliranja relacijskih baz to ni najbolj učinkovito. Vsaka entiteta relacijskega modela kljub temu naključni celoštevilčni identifikator, ki ima vlogo primarnega ključa, vendar se tega ne uporablja. Za določanje identite se v vsaki entiteti uporablja atribut *uri*, ki je enak identifikatorju v RDF datotekah. Tuji ključi entite so uri identifikatorji drugih tabel. Za vsak *uri* atribut ter za vsak tuj ključ smo kreirali indekse. Slika 4.3 prikazuje strukturo tabel v relacijski bazi, ki smo jo uporabili za evalvacijo pristopov, ki za fizično hrambo podatkov uporabljajo relacijske baze. Koda za kreiranje relacijske baze je v prilogi C.

4.1.2.2 LUBM povpraševanja

Avtorji LUBM so v okviru specifikacije definirali 14 SPARQL povpraševanj, od katerih vsako naslavlja svoj vidik kompleksnosti povpraševanja ter sklepanja nad OWL-DL ontologijami. Povpraševanja Q1, Q2, Q3 in Q14 ne zahtevajo sklepanja, medtem ko vsa ostala povpraševanja sklepanje zahtevajo. V nadaljevanju povzemamo povpraševanja [37].

Povpraševanje Q1

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:GraduateStudent .
?X ub:takesCourse
  <http://www.Department0.University0.edu/GraduateCourse0>}
```

Izpis 4.1: Sparql povpraševanje Q1

Povpraševanje vrača vse dodiplomske študente, ki poslušajo določen predmet. Povpraševanje ima velik vhod in visoko selektivnost. Nanaša se zgolj na en razred in eno lastnost. Povpraševanje ne predpostavlja nobenih informacij o razredni hierarhiji ali sklepanja.

Povpraševanje Q2

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
 SELECT * WHERE
 {
       ?x rdf:type ub:GraduateStudent .
       ?y rdf:type ub:University .
       ?z rdf:type ub:Department .
       ?x ub:memberOf ?z .
       ?z ub:subOrganizationOf ?y .
```

```
      ?x ub:undergraduateDegreeFrom ?y .
}
```

Izpis 4.2: Sparql povpraševanje Q2

Povpraševanje vrne vse dodiplomske študente, ki so končali študij, vključno s fakulteto in inštitutom na katerem so diplomirali. Povpraševanje poveča kompleksnost, saj vključuje 3 razrede in 3 spremenljivke. Med razredi je t.i. trikotni vzorec, saj so vsi razredi v povpraševanju odvisni drug od drugega.

Povpraševanje Q3

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:Publication .
?X ub:publicationAuthor
<http://www.Department0.University0.edu/AssistantProfessor0>}
```

Izpis 4.3: Sparql povpraševanje Q3

Povpraševanje vrne vse publikacije določenega docenta. Povpraševanje je podobno povpraševanju Q1, s to razliko, da ima razred *Publication* široko razredno hierarhijo.

Povpraševanje Q4

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X ?Y1 ?Y2 ?Y3
WHERE
{?X rdf:type ub:Professor .
?X ub:worksFor <http://www.Department0.University0.edu> .
?X ub:name ?Y1 .
?X ub:emailAddress ?Y2 .
```

```
?X ub:telephone ?Y3}
```

Izpis 4.4: Sparql povpraševanje Q4

Povpraševanje vrne vse profesorje, njihova imena, telefonske številke ter naslove elektronske pošte za določen inštitut. Povpraševanje ima majhen vhod in visoko selektivnost. Povpraševanje predvideva sklepanje na osnovi razredne hierarhije. V generiranih podatkih ni noben vir eksplicitno definiran kot primerek razreda *Professor*, temveč kot njegovi podrazredi (npr. *FullProfessor*). Razred *Professor* ima izredno široko hierarhijo, dodatno Q4 povprašuje po več lastnostih posameznega razreda.

Povpraševanje Q5

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:Person .
?X ub:memberOf <http://www.Department0.University0.edu>}
```

Izpis 4.5: Sparql povpraševanje Q5

Povpraševanje vrne vse osebe, ki pripadajo k določenemu inštitutu. Povpraševanje predpostavlja sklepanje na osnovi razredne hierarhije med razredom *Person* in njegovimi podrazredi. Hkrati povpraševanje predpostavlje sklepanje na osnovi hierarhije lastnosti med lastnostjo *memberOf* in njenimi podlastnostmi. Razred *Person* ima široko in globoko hierarhijo.

Povpraševanje Q6

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X WHERE {?X rdf:type ub:Student}
```

Izpis 4.6: Sparql povpraševanje Q6

Povpraševanje vrne vse študente. Q6 povprašuje zgolj po enem razredu, za katerega predpostavlja eksplicitno sklepanje na osnovi razredne hierarhije med razredi *Student* in *UndergraduateStudent* ter implictno sklepanje med razredi *Student* in *GraduateStudent*. Povpraševanje ima velik vhod in nizko selektivnost.

Povpraševanje Q7

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X ?Y
WHERE
{?X rdf:type ub:Student .
?Y rdf:type ub:Course .
?X ub:takesCourse ?Y .
<http://www.Department0.University0.edu/AssociateProfessor0>
     ub:teacherOf ?Y}
```

Izpis 4.7: Sparql povpraševanje Q7

Povpraševanje vrne vse študente, ki poslušajo predavanja določenega profesorja. Povpraševanje je z vidika sklepanja podobno povpraševanju Q6, vendar vpeljuje dodatno kompleksnost tako, da poveča število razredov in lastnosti v povpraševanju. Q6 poviša tudi selektivnost povpraševanja.

Povpraševanje Q8

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X ?Y ?Z
WHERE
{?X rdf:type ub:Student .
?Y rdf:type ub:Department .
?X ub:memberOf ?Y .
?Y ub:subOrganizationOf <http://www.University0.edu> .
?X ub:emailAddress ?Z}
```

Izpis 4.8: Sparql povpraševanje Q8

Povpraševanje vrne vse študente vključno z naslovi njihove elektronske pošte za določeno univerzo. Povpraševanje Q8 povečuje kompleksnost v primerjavi z Q7 tako, da doda še eno lastnost.

Povpraševanje Q9

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X ?Y ?Z
WHERE
{?X rdf:type ub:Student .
?Y rdf:type ub:Faculty .
?Z rdf:type ub:Course .
?X ub:advisor ?Y .
?Y ub:teacherOf ?Z .
?X ub:takesCourse ?Z}
```

Izpis 4.9: Sparql povpraševanje Q9

Povpraševanje vrne vse študente, kateri imajo za mentorja osebo, pri kateri so poslušali vsaj en predmet. Gre za najkompleksnejše povpraševanje, saj v povpraševanju samem vključuje največje število lastnosti in razredov, ima zahtevno sklepanje, dodatno pa imajo koncepti trikotno odvisnost podobno kot pri povpraševanju Q2.

Povpraševanje Q10

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:Student .
?X ub:takesCourse
<http://www.Department0.University0.edu/GraduateCourse0>}
```

Izpis 4.10: Sparql povpraševanje Q10

Povpraševanje vrne vse študente, ki poslušajo določen dodiplomski predmet. Kompleksnost sklepanja povpraševanja je podobna povpraševanjem Q6 – Q9, struktura povpraševanja je enostavna. Povpraševanje ima velik vhod in visoko selektivnost.

Povpraševanje Q11

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:ResearchGroup .
?X ub:subOrganizationOf <http://www.University0.edu>}
```

Izpis 4.11: Sparql povpraševanje Q11

Povpraševanje vrne vse raziskovalne skupine, ki delujejo v sklopu določene univerze. Povpraševanja Q11, Q12 in Q13 zahtevajo kompleksnejše OWL-DL sklepanje. Relacija *subOrganizationOf* je definirana kot tranzitivna relacija.

Povpraševanje Q12

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X ?Y
WHERE
{?X rdf:type ub:Chair .
?Y rdf:type ub:Department .
?X ub:worksFor ?Y .
?Y ub:subOrganizationOf <http://www.University0.edu>}
```

Izpis 4.12: Sparql povpraševanje Q12

Povpraševanje vrne vse predstojnike inštitutov za določeno univerzo. Razred *Chair* je definiran z eksistenčno omejitvijo na relaciji *headOf*. Vsi primerki razreda *Professor*, ki imajo definirano relacijo *headOf*, so tudi primerki razreda *Chair*. Vhod tega povpraševanja je majhen, selektivnost pa visoka.

Povpraševanje Q13

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:Person .
<http://www.University0.edu> ub:hasAlumnus ?X}
```

Izpis 4.13: Sparql povpraševanje Q13

Povpraševanje vrne vse osebe, ki so zaključile študij na določeni univerzi. Lastnost *hasAlumnus* je definirana kot inverzna lastnost *degreeFrom*, ki ima tri podlastnosti (*undergraduateDegreeFrom, mastersDegreeFrom, doctoralDegreeFrom*). Povpraševanje tako predvideva sklepanje na osnovi inverzne lastnosti kot tudi hierarhije lastnosti.

Povpraševanje Q14

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X
WHERE {?X rdf:type ub:UndergraduateStudent}
```

Izpis 4.14: Sparql povpraševanje Q14

Povpraševanje vrne vse dodiplomske študente. Gre za najenostavnejše povpraševanje med vsemi. Povpraševanje vsebuje zgolj en razred, ima velik vhod in nizko selektivnost.

4.1.2.3 Razširitev LUBM

Večina povpraševanj v LUBM pristopu predvideva OWL-DL sklepanje. Za pristope, ki ne podpirajo sklepanja, LUBM tako nudi dokaj slabo oceno učinkovitosti. Za nas sta zanimiva predvsem naslednja kriterija, za katera obstoječa množica ne da odgovorov, v kolikor pristop ne podpira sklepanja:

- **skalabilnost** - kako vpliva količina vrnjenih podatkov na hitrost izvajanja povpraševanj za enakovredna povpraševanja,
- **stičnost** - kako vpliva združevanje lastnosti iz več razredov na učinkovitost izvajanja povpraševanj.

Da bi ponudili odgovor na zgoraj podana kriterija tudi za pristope, ki ne podpirajo sklepanja, smo definirali dve novi povpraševanji Q1a in Q3a, ki sta izpeljani iz povpraševanj Q1 in Q3.

Povpraševanje Q1a

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:UndergraduateStudent .
?X ub:memberOf
  <http://www.Department0.University0.edu}
```

Izpis 4.15: Sparql povpraševanje Q1a

Povpraševanje vrne vse dodiplomske študente na določenem inštitutu. Povpraševanje je podobno povpraševanju Q1, povprašuje po razredu z enakovredno hierarhijo. Razlika je, da je selektivnost povpraševanja Q1a manjša, zaradi česar povpraševanje vrne več rezultatov. Na osnovi povpraševanj Q1 in Q1a lahko sklepamo, kakšna je skalabilnost pristopa glede na količino odgovorov.

Povpraševanje Q3a

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://ii.uni-mb.si/OSII/lubm_1.owl#>
SELECT ?X ?Yname
WHERE
{?X rdf:type ub:Publication .
?X ub:name ?Yname.
?X ub:publicationAuthor
```

```
<http://www.Department0.University0.edu/AssistantProfessor0>}
```

Izpis 4.16: Sparql povpraševanje Q3a

Povpraševanje vrne vse publikacije določenega docenta vključno z naslovom publikacije. Povpraševanje Q3a je podobno povpraševanju Q3, Q3a povprašuje zgolj po dodatni lastnosti. S tem povpraševanjem dobimo informacijo, kakšen vpliv ima povpraševanje po dodatnih parametrih na čas izvajanja. V primeru, ko se za fizično hrambo podatkov uporabi relacijska baza, pride pri tem povpraševanju do stika dveh tabel (*Publication* in *PublicationAuthor*).

4.2 Eksperiment: primerjava učinkovitosti

V okviru doktorske disertacije smo razvili prototip, ki implementira pristop, ki smo ga opisali v poglavju 3. Učinkovitost našega pristopa smo kvantitativno in kvalitativno ovrednotili v primerjavi s klasičnimi pristopi. Kvalitativna primerjava je podana v tabelah 4.1 in 4.2. Pristop, ki smo ga razvili smo kvalitativno primerjali s pristopi za integracijo informacijskih virov na osnovi ontologij, ki smo jih odkrili pri pregledu literature.

Nobeden od pristopov, ki so opisani v poglavju 2.4.1 nima implementacije ali pa le-ta ni prosto dostopna. Zaradi tega ni bilo možno primerjati pristopa, ki smo ga razvili v okviru doktorske disertacije, z obstoječimi pristopi. Namesto tega smo razvit pristop primerjali s temeljnimi tehnologijami za delo s semantičnimi tehnologijami. Nekateri od opisanih sorodnih pristopov uporabljajo za temeljne tehnologije rešitve, ki smo jih primerjali z našim pristopov. Prav tako obstaja precejšnja verjetnost, da bodo potencialni novorazviti pristopi prav tako uporabili rešitve, ki smo jih primerjali, za temeljne tehnologije. Na osnovi podanih dejstev je primerjava razvitega pristopa s temeljnimi tehnologijami za delo s semantičnimi tehnologijami smiselna. V eksperimentu smo primerjali naslednje rešitve:

- **Jena:** Jena je odprtokodna knjižnica za delo s tehnologijami semantičnega spleta[1]. Je ena izmed najbolj uporabljanih rešitev za delo z RDF in OWL.

[1] http://jena.sourceforge.net/

Knjižnica ima vgrajenih več pogonov sklepanja, omogoča izvajanja sparql povpraševanj ter vključitev zunanjih pogonov sklepanja. V nadaljevanju se, kadar uporabljamo termin Jena, sklicujemo na Jena pogon povpraševanja brez sklepanja, ki hrani vse podatke v pomnilniku.

- **Jena in Pellet pogon sklepanja (Jena_P):** Pellet je eden izmed najbolj razširjenih OWL-DL pogonov sklepanja, ki omogoča popolno OWL-DL sklepanje [76]. V tem testnem primeru smo ga uporabili za pogon sklepanja v okviru Jena-e. Vsi podatki so se nahajali v pomnilniku.

- **Jena in RDFS Simple pogon sklepanja (Jena_R):** Uporabili smo knjižnico Jena, za pogon sklepanja pa RDFS Simple pogon sklepanja[1]. Ta pogon sklepanja omogoča osnovno sklepanje na osnovi hierarhije razredov in lastnosti.

- **Jena in OWL Micro pogon sklepanja (Jena_O):** Uporabili smo knjižnico Jena v kombinaciji z OWL Micro pogonom sklepanja[2]. Gre za pogon, ki le delno podpira OWL-DL sklepanje. Gre za kompromis med izraznostjo ontologije in hitrostjo izvajanja povpraševanja.

- **TDB:** TDB je razširitev Jene, ki omogoča trajno hranjenje podatkov[3]. Model tako ni več v pomnilniku, temveč je trajno shranjen na trdem disku. Za fizično hrambo podatkov uporablja lasten format zapisa podatkov. TDB je visokozmogljiv, netransakcijski pogon, ki je namenjen uporabi na enem računalniku. Sklepanje z uporabo TDB ni mogoče. V kolikor uporabimo pogon sklepanja, sistem celoten podatkovni model naloži v pomnilnik, nakar poteka izvajanje na enak način kot brez trajne hrambe podatkov.

- **SDB:** SDB je razširitev Jene, ki podobno kot TDB omogoča trajno hranjenje podatkov[4]. Podatkovni model je fizično shranjen v relacijski podatkovni bazi. Analogno kot TDB tudi SDB ne omogoča sklepanja nad podatki. V

[1] http://jena.sourceforge.net/inference/
[2] http://jena.sourceforge.net/inference/
[3] http://jena.sourceforge.net/TDB/
[4] http://jena.sourceforge.net/SDB/

kolikor uporabimo pogon sklepanja, sistem naloži celoten model v pomnilnik.

- **D2R:** D2R pristop omogoča transformacijo podatkov, ki se nahajajo v relacijskih bazah v RDF[7][1]. Transformacija poteka v času izvajanja povpraševanj, dostop do podatkov pa je zagotovljen s sparql končno točko. Transformacija povpraševanj poteka na osnovi preslikav med entitetno relacijskim modelom in RDF modelom. Gre za enega od v splošnem najširše uporabljanih pristopov za transformacijo podatkov iz relacijskih baz v RDF (med drugim ga uporabljata tudi pristopa za integracijo podatkovnih virov na osnovi ontologij OI5 [78] in OI8 [48]).

- **D2R in RDFS Simple pogon sklepanja (D2R_R):** Grez za D2R pogon, ki je ovit z RDFS Simple pogonom sklepanja. Pri uporabi te kombinacije se nekateri podatki shranijo in prenesejo v pomnilnik, nekatere podatke pa sistem še zmeraj črpa iz relacijske podatkovne baze.

- **OSII_P:** To je pristop, ki smo ga razvili v okviru doktorske disertacije. Storitve služijo kot podatkovni viri, sistem samodejno izvede kompozicijo storitev v času izvajanja, jih proži ter združi rezultate. Predstavlja pesimističen scenarij modeliranja storitev. OSII_P vsebuje neoptimizirane storitve, kot jih lahko pričakujemo brez optimizacij glede na globalno ontologijo.

- **OSII_O:** Enak pristop kot OSII_P, s to razliko, da ta testni primer vsebuje dodatne storitve, ki so optimalno modelirane za LUBM primer.

4.2.1 Prototip

Meritve smo izvajali na prototipu, ki smo ga razvili na osnovi modela, ki je predstavljen v poglavju 3. Prototip predstavlja modularno razširitev odprtokodne knjižnice Jena. Razširitev je zasnovana tako, da je pri izvajanju povpraševanja možno izbrati ali naš pristop ali standardno implementacijo pogona za izvajanje povpraševanj. Razširitev ne vpliva na izvajanje standardnih povpraševanj v okviru Jene. Osnovni komponenti prototipa sta:

[1] http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/

- **transformator povpraševanj:** komponenta transformira prvotno povpraševanje v povpraševanje, ki vsebuje klice storitev preko operacije *web_service*. Ta komponenta na osnovi vhodnega povpraševanja identificira storitve, ki jih je potrebno prožiti, sestavi kompozicijo storitev na osnovi semantičnih opisov storitev, globalne ontologije ter vhodnega povpraševanja, in pridobi vhodne podatke storitev. Izhod komponente je transformirano povpraševanje, ki vsebuje načrt proženja storitev zapisan v sparql algebri. Takšen načrt izvajanja prevzame standardni Jena pogon povpraševanja.

- **implementacija operacije** *web_service***:** operacijo proži nespremenjen Jena pogon povpraševanja, kadar se v transformiranem povpraševanju pojavi naziv operacije *web_service*. Operacija v času povpraševanja transformira vhodne podatke, proži storitev ter na osnovi preddefiniranih preslikav transformira izhodne podatke. Integracija podatkov poteka na nivoju sparql algebre na osnovi transformacije povpraševanja. Samo integracijo izvede standardni Jena pogon povpraševanja.

Prototip smo implementirali v programskem jeziku Java. Razširitev smo razvili na osnovi Jene verzije 2.6.4, ki je v času izdelave doktorske disertacije bila zadnja stabilna različica knjižnice. Jena uporablja za pogon povpraševanja knjižnico ARQ. Pri razvoju prototipa smo uporabili predzadnjo stabilno različico (verzija 2.8.7). Predzadnjo različico smo uporabili zaradi tega, ker pristop SDB ni kompatibilen z zadnjo različico ARQ pogona povpraševanja. Da bi zagotovili primerljivost rezultatov, smo uporabili predhodno različico. Kljub temu smo naš prototip preizkusili tudi na zadnji različici ARQ (2.8.8), s katero je kompatibilen.

Izpis 4.17 vsebuje izsek kode, ki izvede OSII povpraševanje in vse rezultate izpiše na standarden izhod. Samo povpraševanje in obdelava rezultatov poteka na povsem enak način kot pri uporabi standardne Jena knjižnice. V primerjavi s klasičnimi Jena povpraševanjem, je edina stvar, ki jo mora uporabnik dodatno narediti, izvedba sparql transformacije, ki je v spodnjem primeru podana v vrsticah 4 in 5.

```
1 Query query = QueryFactory.create(queryString);
2 Op op = Algebra.compile(query) ;
3
```

```
4  op = Transformer.transform(
5          new OSII_SparqlTransformer_Graph(rdfModel), op);
6  op = Algebra.optimize(op) ;
7
8  QueryIterator qIter = Algebra.exec(op, rdfModel) ;
9  ResultSet results =
10     ResultSetFactory.create(qIter, query.getResultVars());
11 while (results.hasNext()) {
12         System.out.println(results.next());
13 }
```

Izpis 4.17: Primer uporabe OSII za izvajanje povpraševanj

4.2.2 Izvajanje LUBM povpraševanj na prototipu OSII

Prototip, ki smo ga razvili, je generičen in ga je mogoče uporabiti na poljubni domeni, v kolikor so podatki izpostavljeni v obliki spletnih storitev. Da bi zagotovili izvajanje LUBM povpraševanj na prototipu OSII, smo razvili SOAP spletne storitve, ki dostopajo do LUBM podatkovne množice. Storitve smo semantično označili skladno z OSII-S ontologijo. Kot globalno shemo smo uporabili nespremenjeno LUBM OWL ontologijo.

Spletne storitve smo zasnovali na osnovi LUBM ontologije. Vsak razred, ki ima podatke materializirane v RDF datotekah, je predstavljen z eno storitvijo. Vsaka storitev lahko ima več operacij. Operacije so dveh vrst, nekatere na osnovi vhodnih kriterijev vračajo zgolj primerke razreda, medtem ko druge vračajo podrobnosti posameznega primerka. V kolikor bi želeli pridobiti imena vseh študentov, ki so vpisani na posamezno fakulteto, bi tako morali prožiti dve različni operaciji spletne storitve. Najprej bi morali pridobiti URI identifikatorje vseh študentov, ki so vpisani na posamezno fakulteto, nato pa bi morali za vsakega študenta pridobiti še njegovo ime. Število klicev storitev je v tem primeru odvisno od števila študentov. Opisan scenarij je pesimističen. Lahko bi seveda zasnovali storitve tako, da bi operacija vračala kar imena študentov, ki so vpisani na posamezni fakulteti (optimističen scenarij). V takšnem primeru bi bil potreben zgolj en klic storitve.

Za namene eksperimenta smo razvili dva testna primera, OSII_P in OSII_O. Prvi testni primer vsebuje storitve, razvite po pesimističnem scenariju, medtem ko drug primer vsebuje dodatne storitve, ki so modelirane po optimističnem scenariju. Storitve dostopajo do LUBM podatkov, ki smo jih izvozili v MySql relacijsko podatkovno bazo[1]. Podatkov nismo spreminjali in so povsem identični kot v generiranih RDF datotekah (glej poglavje 4.1.2.1). Storitve smo razvili z uporabo knjižnice JAX-B[2] in jih namestili na Glassfish aplikacijski strežnik[3] različice 3.1.

4.2.3 Okolje eksperimenta

Eksperiment smo izvedli na namiznem računalniku s procesorjem Intel i5 2500K in 8 GB pomnilnika, na katerem je bila nameščena 64-bitna različica operacijskega sistema Ubuntu 11.04. Vse testne primere razen D2R smo izvajali z naslednjimi komponentami: Java (različica Java SE 1.6.0.24), Jena (različica 2.6.4), ARQ (različica 2.8.7), Pellet pogon sklepanja (različica 2.2.2), TDB (različica 0.8.10), SDB (različica 1.3.4) ter Apache Axis2 (različica 1.5.5). Eksperimente za D2R smo izvedli na različici 0.7. D2R ni kompatibilen z novejšimi različicami knjižnic Jena in ARQ, zato smo uporabili knjižnice, ki jih je vseboval paket D2R različice 0.7. Za izvajanje testov smo rezervirali 4 GB pomnilnika.

Teste smo izvajali sekvenčno. Najprej smo izvedli teste za podatkovne množice LUBM1, nato za LUBM5 ter nato še za LUBM10 in LUBM20. Za vsako podatkovno množico smo zaporedno izvedli teste za vsak pristop. Za vsak pristop smo najprej naložili podatkovno množico. Pred izvajanjem meritev smo izvedli povpraševanje Q1. To smo storili zato, da bi se izognili zakasnitvam zaradi dinamičnega nalaganja razredov. Po inicializaciji smo za vsak pristop sekvenčno izvedli razširjeno množico LUBM povpraševanj. Vsako povpraševanje smo izvedli 10-krat ter nato izračunali povprečni čas izvajanja povpraševanja.

Pri pristopih Jena_M, Jena_P, Jena_R, Jena_O, TDB in SDB smo za podatkovne množice uporabili RDF datoteke, ki smo jih dobili neposredno od LUBM generatorja. Za pristopa D2R in OSII smo podatke iz generiranih RDF datotek

[1] http://www.mysql.com/
[2] http://jaxb.java.net/
[3] http://glassfish.java.net/

uvozili v MySQL podatkovno bazo (različice 5.1.54). Podatkov nismo transformirali, za uvoz smo uporabili pristop, ki smo ga opisali v poglavju 4.1.2.1. Pristopa OSII in D2R sta za vsako množico podatkov dostopala do iste podatkovne baze.

4.3 Analiza in razlaga rezultatov eksperimenta

V nadaljevanju podajamo rezultate eksperimenta ter njihovo analizo. Najprej podajamo analizo izvajanja LUBM povpraševanj na razvitem prototipu, nato podajamo analizo časovne zahtevnosti. Nadalje podajamo analizo kvantitativne primerjave razvitega in sorodnih pristopov ter na koncu še statistično analizo rezultatov.

4.3.1 Analiza LUBM povpraševanj na pristopu OSII

V nadaljevanju podajamo analizo LUBM povpraševanj za pristop, ki smo ga razvili v okviru doktorske disertacije na osnovi pesimističnega testnega primera. Za vsako povpraševanje podajamo vhodno povpraševanje v sparql algebri ter transformirano povpraševanje v sparql algebri. Transformiran graf povpraševanja predstavlja načrt proženja storitev in je ključen element pristopa, ki smo ga razvili.

Povpraševanje Q1

Povpraševanje Q1 je relativno enostavno, po transformaciji sistem proži storitev *GraduateStudent_course*, za katero pridobi vhodni parameter iz povpraševanja.

```
(project (?X)
  (bgp
    (triple ?X <rdf:type> <lubm:GraduateStudent>)
    (triple ?X <lubm:takesCourse>
        <http://www.Department0.University0.edu/GraduateCourse0>)
  ))
```

Izpis 4.18: Vhodno povpraševanje Q1

```
(project (?X)
  (web_service  GraduateStudent_course{[?X]}
    Input: <http://www.Department0.University0.edu/GraduateCourse0>
  ))
```

Izpis 4.19: Transformirano povpraševanje Q1

Povpraševanje Q1a

Povpraševanje Q1a je podobno povpraševanju Q1. Edina razlika je, da sistem v povpraševanju Q1a proži drugo storitev.

```
(project (?X)
  (bgp
    (triple ?X <rdf:type> <lubm:UndergraduateStudent>)
    (triple ?X <lubm:memberOf> <http://www.Department0.University0.edu>)
  ))
```

Izpis 4.20: Vhodno povpraševanje Q1a

```
(project (?X)
  (web_service  UndergraduateStudent_memberOf{[?X]}
    Input: <http://www.Department0.University0.edu>
  ))
```

Izpis 4.21: Transformirano povpraševanje Q1a

Povpraševanje Q2

Povpraševanje vsebuje t.i. trikotni vzorec med spremenljivkami, definiranimi v povpraševanju, kar pomeni, da so 3 spremenljivke odvisne druga od druge. Kot smo že izpostavili, OSII ne podpira povpraševanj, ki vključujejo trikotni vzorec med parametri, zaradi tega vrne transformacija kot rešitev prazno povpraševanje.

```
(project (?X)
  (bgp
    (triple ?x <rdf:type> <lubm:GraduateStudent>)
    (triple ?y <rdf:type> <lubm:University>)
```

```
    (triple ?z <rdf:type> <lubm:Department>)
    (triple ?x <lubm:memberOf> ?z)
    (triple ?z <lubm:subOrganizationOf> ?y)
    (triple ?x <lubm:undergraduateDegreeFrom> ?y)
  ))
```

Izpis 4.22: Vhodno povpraševanje Q2

```
(project (?X)
  (bgp
  ))
```

Izpis 4.23: Transformirano povpraševanje Q2

Povpraševanje Q3

Vzorec povpraševanja je podoben povpraševanjema Q1 in Q1a. Sistem proži storitev *Publication_author*, katere vhod dobi v povpraševanju.

```
(project (?X)
  (bgp
    (triple ?X <rdf:type> <lubm:Publication>)
    (triple ?X <lubm:publicationAuthor>
      <http://www.Department0.University0.edu/AssistantProfessor0>)
  ))
```

Izpis 4.24: Vhodno povpraševanje Q3

```
(project (?X)
  (web_service  Publication_author{[?X]}
   Input:
   <http://www.Department0.University0.edu/AssistantProfessor0>
  ))
```

Izpis 4.25: Transformirano povpraševanje Q3

Povpraševanje Q3a

To povpraševanje razširja povpraševanje Q3. Sistem izvede kompozicijo storitev. Najprej proži storitev *Publication_author*, ki vrne vse publikacije za posameznega avtorja. Sistem dobi vhod storitve iz vhodnega povpraševanja, rezultat storitve pa veže na spremenljivko *?X*. Druga storitev *Publication_name* je odvisna od prve storitve, saj vhod storitve predstavlja spremenljivka *?X*. Za vsako publikacijo, ki je rezultat klica prve storitve, sistem proži drugo storitev, katere rezultat je naziv publikacije, ki ga sistem veže na spremenljivko *?Y*.

```
(project (?X ?Yname)
  (bgp
    (triple ?X <rdf:type> <lubm:Publication>)
    (triple ?X <lubm:name> ?Yname)
    (triple ?X <lubm:publicationAuthor>
      <http://www.Department0.University0.edu/AssistantProfessor0>)
  ))
```

Izpis 4.26: Vhodno povpraševanje Q3a

```
(project (?X ?Y)
  (leftjoin
    (web_service  Publication_author{[?X]}
     Input:
     <http://www.Department0.University0.edu/AssistantProfessor0> )
    (web_service  Publication_name{[?Y]}
     Input: ?X )
  ))
```

Izpis 4.27: Transformirano povpraševanje Q3a

Povpraševanje Q4

Povpraševanje zahteva sklepanje na osnovi ontologije ter kompozicijo storitev. Skladno s hierarhijo razredov v globalni ontologiji sistem proži storitve *AssistantProfessor_worksFor, FullProfessor_worksFor, AssociateProfessor_worksFor*, ki vračajo primerke razredov *AssistantProfessor, FullProfessor* in *AssociateProfessor*.

Ti razredi so podrazredi razreda *Professor*. Vhod teh storitev je oddelek, ki je podan z vhodnim povpraševanjem. Sistem izvede unijo rezultatov teh treh storitev ter rezultate veže na spremenljivko *?X*.

Sistem nato proži še storitev *Professor_personalData*, ki vrača podatke o imenu, telefonski številki ter naslovu elektronske pošte za posameznega profesorja. Storitev se proži za vsak rezultat, ki je vezan na spremenljivko *?X* (za vsak primerek tipa *Professor*).

```
(project (?X ?Y1 ?Y2 ?Y3)
  (bgp
    (triple ?X <rdf:type> <lubm:Professor>)
    (triple ?X <lubm:worksFor>
      <http://www.Department0.University0.edu>)
    (triple ?X <lubm:name> ?Y1)
    (triple ?X <lubm:emailAddress> ?Y2)
    (triple ?X <lubm:telephone> ?Y3)
  ))
```

Izpis 4.28: Vhodno povpraševanje Q4

```
(project (?X ?Y1 ?Y2 ?Y3)
  (leftjoin
    (union
      (union
        (web_service  AssistantProfessor_worksFor{[?X]}
         Input: <http://www.Department0.University0.edu> )
        (web_service  FullProfessor_worksFor{[?X]}
         Input: <http://www.Department0.University0.edu> ))
      (web_service  AssociateProfessor_worksFor{[?X]}
         Input: <http://www.Department0.University0.edu> ))
    (web_service  Professor_personalData{[?Y2, ?Y3, ?Y1]}
         Input: ?X )
  ))
```

Izpis 4.29: Transformirano povpraševanje Q4

Povpraševanje Q5

Povpraševanje je enostavnejše od predhodnega saj zahteva samo sklepanje na osnovi razredne strukture. Na osnovi podrazredov razreda *Person* sistem proži storitve *AssistantProfessor_worksFor, FullProfessor_worksFor, AssociateProfessor_worksFor, GraduateStudent_memberOf* in *UndergraduateStudent_memberOf*. Vhod vseh storitev sistem pridobi iz vhodnega povpraševanja.

```
(project (?X)
  (bgp
    (triple ?X <rdf:type> <lubm:Person>)
    (triple ?X <lubm:memberOf>
     <http://www.Department0.University0.edu>)
  ))
```

Izpis 4.30: Vhodno povpraševanje Q5

```
(project (?X)
  (union
    (union
      (union
        (union
          (union
            (web_service  Lecturer_worksFor{[?X]}
             Input: <http://www.Department0.University0.edu>)
            (web_service  AssistantProfessor_worksFor{[?X]}
             Input: <http://www.Department0.University0.edu>))
          (web_service  FullProfessor_worksFor{[?X]}
           Input: <http://www.Department0.University0.edu>))
        (web_service  AssociateProfessor_worksFor{[?X]}
         Input: <http://www.Department0.University0.edu>))
      (web_service  GraduateStudent_memberOf{[?X]}
       Input: <http://www.Department0.University0.edu>))
    (web_service  UndergraduateStudent_memberOf{[?X]}
     Input: <http://www.Department0.University0.edu>)
  ))
```

Izpis 4.31: Transformirano povpraševanje Q5

Povpraševanje Q6

Q6 povprašuje po vseh primerkih tipa *Student*. Na osnovi razredne hierarhije sistem izbere storitvi *GraduateStudent_memberOf* in *UndergraduateStudent_memberOf*. Obe storitvi zahtevata vhodni podatek tipa *Department*. V vhodnem povpraševanju ni podan noben podatek, ki bi ga bilo možno uporabiti za vhod katerekoli storitve, zaradi tega sistem izvede povpraševanje po lokalnem modelu, iz katerega pridobi seznam primerkov tipa *Department*. Za vsak takšen primerek sistem proži obe storitvi.

```
(project (?X)
  (bgp (triple ?X <rdf:type> <lubm:Student>)))
```

Izpis 4.32: Vhodno povpraševanje Q6

```
(project (?X)
  (union
    (web_service  GraduateStudent_memberOf{[?X]}
      Input:
      (bgp (triple ?X <rdf:type> <lubm:Department>))
    )
    (web_service  UndergraduateStudent_memberOf{[?X]}
      Input:
      (bgp (triple ?X <rdf:type> <lubm:Department>))
    )
  ))
```

Izpis 4.33: Transformirano povpraševanje Q6

Povpraševanje Q7

Q7 povprašuje po vseh študentih, ki poslušajo predavanja določenega predavatelja. Pri tem povpraševanju gre za kompozicijo storitev. Sistem najprej proži storitev *Faculty_teacherOf* na osnovi podatka, podanega v vhodnem povpraševanju. Storitev vrne vse predmete, ki jih predava določen predavatelj. Za vsak predmet sistem proži storitvi *GraduateStudent_course* in *UndergraduateStudent_course* na osnovi rezultatov prve storitve.

```
(project (?X ?Y)
  (bgp
    (triple ?X <rdf:type> <lubm:Student>)
    (triple ?Y <rdf:type> <lubm:Course>)
    (triple ?X <lubm:takesCourse> ?Y)
    (triple <http://www.Department0.University0.edu/Associate
Professor0> <lubm:teacherOf> ?Y)
  ))
```

Izpis 4.34: Vhodno povpraševanje Q7

```
(project (?X ?Y)
  (leftjoin
    (web_service  Faculty_teacherOf{[?Y]}
     Input: <http://www.Department0.University0.edu/Associate
Professor0> )
    (union
      (web_service  GraduateStudent_course{[?X]}
       Input: ?Y )
      (web_service  UndergraduateStudent_course{[?X]}
       Input: ?Y ))
  ))
```

Izpis 4.35: Transformirano povpraševanje Q7

Povpraševanje Q8

Povpraševanje predstavlja kombinacijo podatkov pridobljenih v RDF modelu, na osnovi katerih sistem proži storitve. Podatkov tipa *Department* ne zagotavljajo storitve, temveč se nahajajo v RDF modelu. Zaradi tega sistem zadrži dele povpraševanja, ki se nanašajo na podatke, ki jih ne zagotavljajo storitve (oddelki univerze *http://www.University0.edu*). Na osnovi teh podatkov sistem proži dve kompoziciji storitev. Prva kompozicija vsebuje storitvi *GraduateStudent_memberOf* in *GraduateStudent_email*. Vhod prve storitve predstavlja oddelek univerze, ki ga sistem pridobi iz RDF modela, medtem ko vhod druge storitve predstavlja izhod prve storitve. Analogno tej kompoziciji sistem proži tudi

kompozicijo storitev *UndergraduateStudent_memberOf* in *UndergraduateStudent_email*. Po proženju obeh kompozicij sistem združi podatke, ki sta jih vrnili kompoziciji.

```
(project (?X ?Y ?Z)
  (bgp
    (triple ?X <rdf:type> <lubm:Student>)
    (triple ?Y <rdf:type> <lubm:Department>)
    (triple ?X <lubm:memberOf> ?Y)
    (triple ?Y <lubm:subOrganizationOf>
       <http://www.University0.edu>)
    (triple ?X <lubm:emailAddress> ?Z)
  ))
```

Izpis 4.36: Vhodno povpraševanje Q8

```
(project (?X ?Y ?Z)
  (leftjoin
    (bgp
      (triple ?Y <rdf:type> <lubm:Department>)
      (triple ?Y <lubm:subOrganizationOf>
 <http://www.University0.edu>)
    )
    (union
      (leftjoin
        (web_service  GraduateStudent_memberOf{[?X]}
         Input: ?Y )
        (web_service  GraduateStudent_email{[?Z]}
         Input: ?X ))
      (leftjoin
        (web_service  UndergraduateStudent_memberOf{[?X]}
         Input: ?Y )
        (web_service  UndergraduateStudent_email{[?Z]}
         Input: ?X )))
     ))
```

Izpis 4.37: Transformirano povpraševanje Q8

Povpraševanje Q9

To povpraševanje tvori trikotni vzorec spremenljivk, podobno kot pri povpraševanju Q2. Sistem ne podpira trikotnih vzorcev v povpraševanjih, zaradi tega vrne za rešitev prazno povpraševanje.

```
(project (?X ?Y ?Z)
  (bgp
    (triple ?X <rdf:type> <lubm:Student>)
    (triple ?Y <rdf:type> <lubm:Faculty>)
    (triple ?Z <rdf:type> <lubm:Course>)
    (triple ?X <lubm:advisor> ?Y)
    (triple ?Y <lubm:teacherOf> ?Z)
    (triple ?X <lubm:takesCourse> ?Z)
  ))
```

Izpis 4.38: Vhodno povpraševanje Q9

```
(project (?X ?Y ?Z)
  (bgp
  ))
```

Izpis 4.39: Transformirano povpraševanje Q9

Povpraševanje Q10

Pri tem povpraševanju gre za precej enostaven vzorec. Na osnovi sklepanja nad ontologijo sistem proži storitvi *GraduateStudent_course* in *UndergraduateStudent_course*, katerih vhod dobi v povpraševanju samem.

```
(project (?X)
  (bgp
    (triple ?X <rdf:type> <lubm:Student>)
    (triple ?X <lubm:takesCourse>
<http://www.Department0.University0.edu/GraduateCourse0>)
  ))
```

Izpis 4.40: Vhodno povpraševanje Q10

```
(project (?X)
  (union
    (web_service  GraduateStudent_course{[?X]}
     Input:
<http://www.Department0.University0.edu/GraduateCourse0> )
    (web_service  UndergraduateStudent_course{[?X]}
     Input:
<http://www.Department0.University0.edu/GraduateCourse0> )
   ))
```

Izpis 4.41: Transformirano povpraševanje Q10

Povpraševanje Q11

Q11 povprašuje samo po podatkih, ki se nahajo v RDF modelu, zato tega OSII ne transformira vhodnega povpraševanja.

```
(project (?X)
  (bgp
    (triple ?X <rdf:type> <lubm:ResearchGroup>)
    (triple ?X <lubm:subOrganizationOf>
       <http://www.University0.edu>)
  ))
```

Izpis 4.42: Vhodno povpraševanje Q11

```
(project (?X)
  (bgp
    (triple ?X <rdf:type> <lubm:ResearchGroup>)
    (triple ?X <lubm:subOrganizationOf>
       <http://www.University0.edu>)
  ))
```

Izpis 4.43: Transformirano povpraševanje Q11

Povpraševanje Q12

Povpraševanje temelji na sklepanju na osnovi eksistenčnih kvantifikatorjev (razred *Chair*). Sistem ne podpira sklepanja na osnovi eksistenčnih kvantifikatorjev, zato ne odkrije nobene od storitev, ki bi lahko podale odgovor na povpraševanje. Transformator tako ohrani izvorno povpraševanje.

```
(project (?X ?Y)
  (bgp
    (triple ?X <rdf:type> <lubm:Chair>)
    (triple ?Y <rdf:type> <lubm:Department>)
    (triple ?X <lubm:worksFor> ?Y)
    (triple ?Y <lubm:subOrganizationOf>
       <http://www.University0.edu>)
  ))
```

Izpis 4.44: Vhodno povpraševanje Q12

```
(project (?X ?Y)
  (bgp
    (triple ?X <rdf:type> <lubm:Chair>)
    (triple ?Y <rdf:type> <lubm:Department>)
    (triple ?X <lubm:worksFor> ?Y)
    (triple ?Y <lubm:subOrganizationOf>
       <http://www.University0.edu>)
  ))
```

Izpis 4.45: Transformirano povpraševanje Q12

Povpraševanje Q13

Povpraševanje Q13 zahteva sklepanje na osnovi inverzne lastnosti. Sistem ne podpira sklepanja na osnovi inverznih lastnosti, zaradi tega ne izbere pravilnih storitev za proženje. Kljub temu, da so v transformiranem povpraševanju predvideni klici storitev, do teh klicev na pride, saj prvotno povpraševanje po RDF modelu vrne kot rezultat prazno množico.

```
(project (?X)
  (bgp
    (triple ?X <rdf:type> <lubm:Person>)
    (triple <http://www.University0.edu> <lubm:hasAlumnus> ?X)
  ))
```

Izpis 4.46: Vhodno povpraševanje Q13

```
(project (?X)
  (join
    (bgp
      (triple <http://www.University0.edu> <lubm:hasAlumnus> ?X)
    )
    (union
      (union
        (union
          (union
            (union
              (union
                (web_service  Lecturer_worksFor{[?X]}
                  Input:
                  (bgp (triple ?Y <rdf:type> <lubm:Department>))
                )
                (web_service  AssistantProfessor_worksFor{[?X]}
                  Input:
                  (bgp (triple ?Y <rdf:type> <lubm:Department>))
                ))
              (web_service  FullProfessor_worksFor{[?X]}
                Input:
                (bgp (triple ?Y <rdf:type> <lubm:Department>))
              ))
            (web_service  AssociateProfessor_worksFor{[?X]}
              Input:
              (bgp (triple ?Y <rdf:type> <lubm:Department>))
            ))
          (web_service  GraduateStudent_memberOf{[?X]}
            Input:
```

```
        (bgp (triple ?Y <rdf:type> <lubm:Department>))
      ))
    (web_service  UndergraduateStudent_memberOf{[?X]}
      Input:
      (bgp (triple ?Y <rdf:type> <lubm:Department>))
    ))
))
```

Izpis 4.47: Transformirano povpraševanje Q13

Povpraševanje Q14

Sistem proži storitev *UndergraduateStudent_memberOf*, katere vhod pridobi iz RDF modela.

```
(project (?X)
  (bgp (triple ?X <rdf:type> <lubm:UndergraduateStudent>)))
```

Izpis 4.48: Vhodno povpraševanje Q14

```
(project (?X)
  (web_service  UndergraduateStudent_memberOf{[?X]}
    Input:
    (bgp (?Y <rdf:type> <lubm:Department>))
  ))
```

Izpis 4.49: Transformirano povpraševanje Q14

4.3.2 Analiza časovne zahtevnosti pristopa OSII

V nadaljevanju so podani rezultati meritev časovne zahtevnosti predlaganega pristopa na osnovi razvitega prototipa. Meritve smo izvajali skladno s postopkom, opisanim v poglavju 4.2. Vsako povpraševanje je bilo, skladno s pogoji eksperimenta, izvedeno desetkrat.

Tabeli 4.4 in 4.5 podajata povzetek rezultatov meritev časovne zahtevnosti. Tabeli vsebujeta naslednje podatke:

- **podatkovna množica:** podatkovna množica, nad katero je bilo izvedeno povpraševanje,
- **povpraševanje:** povpraševanje, ki je bilo izvedeno,
- **skupen čas izvajanja:** skupen čas, ki je bil potreben za izvedbo povpraševanja,
- **število odgovorov:** število odgovorov, ki jih je povpraševanje vrnilo,
- **delež čakanja na odgovor:** čas, ki je bil potreben za pridobitev odgovora storitve (čas med poslanim vhodnim in prejetim izhodnim sporočilom), izražen v odstotkih glede na skupni čas izvajanja povpraševanja,
- **število klicev storitev:** število klicev storitev, ki je bilo opravljeno pri izvedbi povpraševanja,
- **povprečni čas izvajanja storitve:** povprečni čas izvajanja ene storitve je skupen čas izvajanja povpraševanja deljen s številom klicev storitev.

Iz rezultatov v tabelah 4.4 in 4.5 lahko opazimo, da ima sistem izredno dobre lastnosti skalabilnosti. Povpraševanja, ki vsebujejo večje število klicev storitev, imajo glede na rezultate manjši povprečni čas izvajanja ene storitve. Le-ta se na celotni množici rezultatov giblje med 2 in 5 milisekundami. Ta čas vsebuje celotni proces proženja storitev, kar vključuje pripravo vhodnih podatkov, transformacijo vhodnih in izhodnih podatkov ter samo proženje storitve, vključno s čakanjem na odgovor.

Iz rezultatov opazimo tudi, da velik delež celotnega časa izvajanja povpraševanja predstavlja čakanje na odgovor storitve. Za vsa povpraševanja, ki vsebujejo klice storitev, predstavlja čakanje na odgovor vsaj 50 % celotnega časa izvajanja povpraševanja. Iz rezultatov je možno tudi razbrati trend naraščanja učinkovitosti glede na količino rezultatov in število klicev. Odstotek čakanja glede na celotni čas izvajanja se veča z večanjem količine rezultatov ali števila klicev storitev. Ta podatek nakazuje na dobro skalabilnost prototipa, saj se z večanjem števila storitev manjša sorazmerni del časa procesiranja znotraj OSII.

Podatkovna množica	Povpra-ševanje	Skupen čas izvajanja (ms)	Število odgovorov	Delež čakanja na odgovor	Število klicev storitev	Povprečni čas izvajanja storitve (ms)
LUBM1	Q1	4,9	4	51,02%	1	4,9
	Q1a	7,2	532	72,22%	1	7,2
	Q2	3,2	0	0,00%	0	0,0
	Q3	3,3	6	63,64%	1	3,3
	Q3a	8,3	6	65,06%	2	4,2
	Q4	90,2	34	95,68%	37	2,4
	Q5	21,8	719	86,24%	6	3,6
	Q6	119,3	7.790	96,40%	30	4,0
	Q7	24,9	67	87,55%	9	2,8
	Q8	17.218,3	7.790	99,49%	7820	2,2
	Q9	3,1	0	0,00%	0	0,0
	Q10	6,3	4	77,78%	2	3,2
	Q11	3,0	0	0,00%	0	0,0
	Q12	1,5	0	0,00%	0	0,0
	Q14	64,7	5.916	95,98%	15	4,3
LUBM5	Q1	4,2	4	61,90%	1	4,2
	Q1a	7,9	532	73,42%	1	7,9
	Q2	3,8	0	0,00%	0	0,0
	Q3	4,5	6	64,44%	1	4,5
	Q3a	8,1	6	70,37%	2	4,1
	Q4	90,2	34	97,01%	37	2,4
	Q5	20,3	719	87,68%	6	3,4
	Q6	653,6	48.582	97,57%	186	3,5
	Q7	23,0	67	87,83%	9	2,6
	Q8	17.318,6	7.790	99,48%	7820	2,2
	Q9	3,2	0	0,00%	0	0,0
	Q10	5,8	4	75,86%	2	2,9
	Q11	5,1	0	0,00%	0	0,0
	Q12	1,8	0	0,00%	0	0,0
	Q14	381,4	36.682	97,82%	93	4,1
LUBM10	Q1	3,4	4	67,65%	1	3,4
	Q1a	6,3	532	74,60%	1	6,3
	Q2	1,6	0	0,00%	0	0,0
	Q3	3,4	6	67,65%	1	3,4
	Q3a	6,6	6	71,21%	2	3,3
	Q4	90,4	34	96,79%	37	2,4
	Q5	18,5	719	90,81%	6	3,1
	Q6	1.392,3	99.566	97,72%	378	3,7
	Q7	23,6	67	92,80%	9	2,6
	Q8	17.632,8	7.790	99,46%	7820	2,3
	Q9	3,1	0	0,00%	0	0,0
	Q10	5,7	4	73,68%	2	2,9
	Q11	5,0	0	0,00%	0	0,0
	Q12	1,2	0	0,00%	0	0,0
	Q14	793,8	75.547	98,00%	189	4,2
	Q1	4,3	4	62,79%	1	4,3
	Q1a	7,7	532	75,32%	1	7,7

Podatkovna množica	Povpra-ševanje	Skupen čas izvajanja (ms)	Število odgovorov	Delež čakanja na odgovor	Število klicev storitev	Povprečni čas izvajanja storitve (ms)
LUBM20	Q2	3,2	0	0,00%	0	0,0
	Q3	4,1	6	63,41%	1	4,1
	Q3a	6,8	6	72,06%	2	3,4
	Q4	82,7	34	96,74%	37	2,2
	Q5	19,9	719	86,93%	6	3,3
	Q6	3.045,6	210.603	98,14%	804	3,8
	Q7	20,9	67	92,82%	9	2,3
	Q8	17.850,9	7.790	99,44%	7820	2,3
	Q9	3,8	0	0,00%	0	0,0
	Q10	5,9	4	72,88%	2	3,0
	Q11	9,9	0	0,00%	0	0,0
	Q12	1,4	0	0,00%	0	0,0
	Q14	1.831,1	160.120	98,44%	402	4,6

Tabela 4.4: Povzetek rezultatov meritve časovne zahtevnosti za pesimističen primer

Podatkovna množica	Povpra-ševanje	Skupen čas izvajanja (ms)	Število odgovorov	Delež čakanja na odgovor	Število klicev storitev	Povprečni čas izvajanja storitve (ms)
LUBM1	Q1	6,3	4	59%	1	6,3
	Q1a	8,3	532	77%	1	8,3
	Q2	4,7	0	0%	0	0,0
	Q3	5,2	6	65%	1	5,2
	Q3a	5,5	6	60%	1	5,5
	Q4	119,1	34	97%	37	3,2
	Q5	26,3	719	87%	6	4,4
	Q6	40,9	7.790	92%	2	20,5
	Q7	31,2	67	92%	9	3,5
	Q8	217,9	7.790	96%	30	7,3
	Q9	3,2	0	0%	0	0,0
	Q10	8,3	4	78%	2	4,2
	Q11	2,7	0	0%	0	0,0
	Q12	1,4	0	0%	0	0,0
	Q14	26,6	5.916	93%	1	26,6
LUBM5	Q1	4,7	4	70%	1	4,7
	Q1a	6,9	532	80%	1	6,9
	Q2	2,5	0	0%	0	0,0
	Q3	5,9	6	78%	1	5,9
	Q3a	5,5	6	55%	1	5,5
	Q4	109,3	34	97%	37	3,0
	Q5	25,3	719	89%	6	4,2
	Q6	396,3	48.582	97%	2	198,2
	Q7	28,9	67	90%	9	3,2

Podatkovna množica	Povpra-ševanje	Skupen čas izvajanja (ms)	Število odgovorov	Delež čakanja na odgovor	Število klicev storitev	Povprečni čas izvajanja storitve (ms)
LUBM5	Q8	254,2	7.790	96%	30	8,5
	Q9	3,0	0	0%	0	0,0
	Q10	7,7	4	78%	2	3,9
	Q11	6,5	0	0%	0	0,0
	Q12	2,5	0	0%	0	0,0
	Q14	348,2	36.682	98%	1	348,2
LUBM10	Q1	4,6	4	67%	1	4,6
	Q1a	7,9	532	75%	1	7,9
	Q2	2,7	0	0%	0	0,0
	Q3	4,6	6	70%	1	4,6
	Q3a	4,5	6	60%	1	4,5
	Q4	107,6	34	96%	37	2,9
	Q5	21,6	719	90%	6	3,6
	Q6	710,7	99.566	97%	2	355,4
	Q7	25,9	67	91%	9	2,9
	Q8	200,1	7.790	95%	30	6,7
	Q9	6,1	0	0%	0	0,0
	Q10	7,4	4	81%	2	3,7
	Q11	5,4	0	0%	0	0,0
	Q12	1,1	0	0%	0	0,0
	Q14	579,7	75.547	98%	1	579,7
LUBM20	Q1	3,6	4	69%	1	3,6
	Q1a	7,6	532	76%	1	7,6
	Q2	2,7	0	0%	0	0,0
	Q3	4,2	6	74%	1	4,2
	Q3a	5,4	6	61%	1	5,4
	Q4	109,8	34	97%	37	3,0
	Q5	22,9	719	91%	6	3,8
	Q6	1.158,8	210.603	96%	2	579,4
	Q7	30,2	67	76%	9	3,4
	Q8	195,2	7.790	95%	30	6,5
	Q9	1,9	0	0%	0	0,0
	Q10	6,8	4	82%	2	3,4
	Q11	7,7	0	0%	0	0,0
	Q12	0,6	0	0%	0	0,0
	Q14	1.074,6	160.120	97%	1	1.074,6

Tabela 4.5: Povzetek rezultatov meritve časovne zahtevnosti za optimističen primer

V tabelah 4.6 in 4.7 podajamo podrobne rezultate meritev časovne zahtevnosti za razširjeno LUBM množico povpraševanj. Tabela 4.6 vsebuje podrobne rezultate za pesimističen primer, tabela 4.7 pa vsebuje podrobne rezultate za

optimističen primer. Za vsako povpraševanje so navedena naslednja merila:

- **skupni čas izvajanja:** celoten čas izvajanja povpraševanja,
- **procesiranje povpraševanja:** čas transformacije povpraševanja,
- **proženje storitev:** celoten čas proženja storitev, ki vključuje pripravo vhodnih podatkov, transformacijo vhodnih in izhodnih podatkov ter proženje same storitve,
- **priprava vhoda:** trajanje transformacije vhodnih podatkov v obliko, zahtevano od posamezne storitve,
- **čakanje na odgovor:** čas čakanja na odgovor spletne storitve (čas med poslanim vhodnim in prejetim izhodnim sporočilom),
- **procesiranje izhoda:** trajanje transformacije izhodnih podatkov v obliko, primerno za procesiranje v sparql povpraševalnem pogonu,
- **število klicev:** število klicev storitev za posamezno povpraševanje.

Za vsako merilo podajamo povprečno vrednost ter standardno deviacijo. Vsi podatki, ki so vezani na čas, so podani v milisekundah. Iz rezultatov lahko opazimo, da je procesiranje povpraševanja izredno hitro. Kljub dejstvu, da v samem procesiranju povpraševanja opravimo sklepanje nad ontologijo, traja procesiranje povpraševanja povprečno manj kot 1 milisekundo.

	Povpra-ševanje	Skupni čas izvajanja		Procesiranje povpr.		Proženje storitev		Priprava vhoda		Čakanje na odgovor		Procesiranje izhoda		Št. klicev
		Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	
LUBM1	Q1	4,9	2,4	1,1	2	2,5	0,5	0,1	0,3	2,4	0,5	0	0	1
	Q1a	7,2	1,2	0,4	0,5	5,2	0,9	0	0	3,1	0,5	2,1	0,5	1
	Q2	3,2	1	1	0,4	0	0	0	0	0	0	0	0	0
	Q3	3,3	0,5	0,2	0,4	2,1	0,5	0	0	2,1	0,5	0	0	1
	Q3a	8,3	1,6	0,8	0,4	5,4	1	0,1	0,3	5	1,3	0,1	0,3	2
	Q4	90,2	7,1	1,1	0,3	86,3	5,8	0,8	1	82,7	5,5	2,6	1,4	37
	Q5	21,8	2,1	0,6	0,5	18,8	1,9	0	0	15,3	1,6	3,5	1	6
	Q6	119,3	6	0,3	0,5	115	5,5	0,2	0,4	81,8	5,8	32,3	5,6	30
	Q7	24,9	2,2	0,7	0,5	21,8	2	0,1	0,3	21	2,2	0,3	0,5	9
	Q8	17218	119	1,3	0,9	17131	114	39,7	6,6	16901	103	162,3	11,3	7820
	Q9	3,1	2	1,1	0,6	0	0	0	0	0	0	0	0	0

	Povpra-ševanje	Skupni čas izvajanja		Procesiranje povpr.		Proženje storitev		Priprava vhoda		Čakanje na odgovor		Procesiranje izhoda		Št. klicev
		Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	
LUBM1	Q10	6,3	1	0,6	0,5	4,9	1,1	0,2	0,4	4,6	1	0,1	0,3	2
	Q11	3	1,5	0,1	0,3	0	0	0	0	0	0	0	0	0
	Q12	1,5	0,5	0,2	0,4	0	0	0	0	0	0	0	0	0
	Q14	64,7	9,7	0,1	0,3	62,1	8,9	0	0	40,6	5,6	21,2	4,3	15
LUBM5	Q1	4,2	0,7	0,3	0,5	2,6	0,5	0	0	2,5	0,5	0,1	0,3	1
	Q1a	7,9	0,3	0,2	0,4	5,8	0,4	0	0	3,3	0,5	2,5	0,5	1
	Q2	3,8	0,6	1,7	0,5	0	0	0	0	0	0	0	0	0
	Q3	4,5	1,7	0,5	0,5	2,9	1,2	0	0	2,9	1,2	0	0	1
	Q3a	8,1	1,8	0,4	0,5	5,7	1,2	0	0	5,6	1,2	0,1	0,3	2
	Q4	90,2	7,8	1,2	0,6	87,5	7,6	0,3	0,5	84,9	7,2	2,2	1,6	37
	Q5	20,3	1,4	1	0	17,8	1,8	0	0	15	1,6	2,8	0,6	6
	Q6	653,6	25,1	0,2	0,4	637,7	24,9	0,9	1,3	461	14,8	173,4	20,9	186
	Q7	23	1,3	0,4	0,5	20,2	1,5	0	0	19,9	1,4	0,3	0,5	9
	Q8	17319	239	1,9	2,7	17228	234	43,7	6,1	16997	223	165,9	13,6	7820
	Q9	3,2	1,9	1,2	0,7	0	0	0	0	0	0	0	0	0
	Q10	5,8	1,3	0,3	0,5	4,4	1,2	0	0	4,4	1,2	0	0	2
	Q11	5,1	2,5	0,2	0,4	0	0	0	0	0	0	0	0	0
	Q12	1,8	1,7	0,2	0,4	0	0	0	0	0	0	0	0	0
	Q14	381,4	28	0	0	373,1	28,3	0,4	0,7	254,9	14	115,9	18,2	93
LUBM10	Q1	3,4	0,5	0,2	0,4	2,3	0,5	0	0	2,2	0,4	0	0	1
	Q1a	6,3	0,5	0,4	0,5	4,7	0,5	0	0	3	0,4	1,7	0,6	1
	Q2	1,6	0,5	0,7	0,5	0	0	0	0	0	0	0	0	0
	Q3	3,4	0,5	0,3	0,5	2,3	0,5	0,1	0,3	2,2	0,4	0	0	1
	Q3a	6,6	0,7	0,6	0,5	4,7	0,5	0	0	4,5	0,7	0,2	0,4	2
	Q4	90,4	8,1	0,8	0,4	87,5	8,3	0,1	0,3	85,2	8,3	2,2	1	37
	Q5	18,5	1,6	0,5	0,5	16,8	1,8	0	0	15,3	2	1,5	0,7	6
	Q6	1392,3	75	0,2	0,4	1360,6	73,1	2,1	1,4	982,9	38,5	370,6	39,2	378
	Q7	23,6	2	0,8	0,4	21,9	1,8	0	0	21,7	2	0,2	0,4	9
	Q8	17633	382	0,9	0,3	17537	375	46	5,8	17293	367	175,1	12,7	7820
	Q9	3,1	0,7	1,1	0,6	0	0	0	0	0	0	0	0	0
	Q10	5,7	1	0,4	0,5	4,2	0,6	0	0	4,1	0,5	0,1	0,3	2
	Q11	5	1,3	0,1	0,3	0	0	0	0	0	0	0	0	0
	Q12	1,2	0,6	0,1	0,3	0	0	0	0	0	0	0	0	0
	Q14	793,8	36,1	0,1	0,3	777,9	36,6	1,4	0,9	518,2	16,7	255,3	26,1	189
LUBM20	Q1	4,3	0,5	0,4	0,5	2,7	0,5	0,1	0,3	2,6	0,5	0	0	1
	Q1a	7,7	0,5	0,5	0,5	5,8	0,4	0	0	3,5	0,5	2,3	0,5	1
	Q2	3,2	0,4	1,2	0,4	0	0	0	0	0	0	0	0	0
	Q3	4,1	0,3	0,4	0,5	2,6	0,5	0	0	2,6	0,5	0	0	1
	Q3a	6,8	1,5	0,6	0,5	4,9	0,9	0	0	4,6	1,1	0,3	0,5	2
	Q4	82,7	6,8	0,9	0,3	80	6,4	0,1	0,3	77,4	6,1	2,5	1,5	37
	Q5	19,9	1,7	0,6	0,5	17,3	1,3	0	0	14,9	0,9	2,4	1,1	6
	Q6	3045,6	113,4	0,3	0,6	2988,8	113,4	3,8	2,5	2135,3	66,1	842,1	61	804
	Q7	20,9	3	0,4	0,5	19,4	2,4	0	0	19,2	2,4	0,2	0,4	9
	Q8	17851	342	1,2	1	17751	331	44,4	4,2	17499	326	179,8	9,2	7820
	Q9	3,8	2,4	1,3	0,6	0	0	0	0	0	0	0	0	0
	Q10	5,9	1,1	0,3	0,5	4,3	0,9	0	0	4,2	0,9	0,1	0,3	2
	Q11	9,9	3	0,1	0,3	0	0	0	0	0	0	0	0	0

	Povpra-ševanje	Skupni čas izvajanja		Procesiranje povpr.		Proženje storitev		Priprava vhoda		Čakanje na odgovor		Procesiranje izhoda		Št. klicev
		Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	
	Q12	1,4	1,6	0,1	0,3	0	0	0	0	0	0	0	0	0
	Q14	1831,1	44	0,2	0,4	1802,6	45,3	2,4	1,4	1167,3	23,2	627	32,3	402

Tabela 4.6: Podroben prikaz izvajalnih časov povpraševanj za pesimističen testni primer

	Povpra-ševanje	Skupni čas izvajanja		Procesiranje povpr.		Proženje storitev		Priprava vhoda		Čakanje na odgovor		Procesiranje izhoda		Št. klicev
		Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	
LUBM1	Q1	6,3	2,1	1,2	2,3	3,7	0,5	0	0	3,7	0,5	0	0	1
	Q1a	8,3	1,4	0,4	0,7	6,4	1	0,1	0,3	3,8	0,6	2,5	0,7	1
	Q2	4,7	0,6	1,7	0,5	0	0	0	0	0	0	0	0	0
	Q3	5,2	1,1	0,4	0,5	3,4	0,5	0	0	3,3	0,5	0,1	0,3	1
	Q3a	5,5	1,2	0,6	0,5	3,3	0,5	0	0	3,1	0,3	0,1	0,3	1
	Q4	119,1	8,1	1,1	0,5	115,2	7,8	1,1	0,9	110,8	7,4	2,6	1,7	37
	Q5	26,3	2,4	1	0	22,9	2,1	0,2	0,4	18,9	1,3	3,6	0,7	6
	Q6	40,9	5,8	0,2	0,4	37,8	5,6	0,2	0,4	11,4	2,4	25,6	3,5	2
	Q7	31,2	2,1	0,4	0,5	28,8	2,4	0,4	0,5	27,6	2,1	0,6	0,7	9
	Q8	217,9	16,6	1,5	0,5	209,4	16,4	1,1	1,3	117	9,3	90,8	8,8	30
	Q9	3,2	1,1	1,1	0,6	0	0	0	0	0	0	0	0	0
	Q10	8,3	1	0,7	0,5	6,5	0,8	0,1	0,3	6,2	0,6	0,2	0,4	2
	Q11	2,7	0,6	0,1	0,3	0	0	0	0	0	0	0	0	0
	Q12	1,4	0,5	0,2	0,4	0	0	0	0	0	0	0	0	0
	Q14	26,6	3,3	0,3	0,5	24,7	3,2	0,1	0,3	7,5	1,5	17	2,4	1
LUBM5	Q1	4,7	1,4	0,5	0,9	3,3	0,9	0	0	3,3	0,9	0	0	1
	Q1a	6,9	1,4	0,1	0,3	5,5	1,1	0,2	0,4	3,4	0,5	1,7	0,8	1
	Q2	2,5	0,7	1	0,4	0	0	0	0	0	0	0	0	0
	Q3	5,9	3,1	0,5	0,5	4,6	2,9	0	0	4,6	2,9	0	0	1
	Q3a	5,5	2,5	0,2	0,4	3	0,4	0	0	2,8	0,6	0,2	0,4	1
	Q4	109,3	9,1	1,3	0,5	106,1	9,1	0,7	0,6	102,6	8,3	2,3	0,9	37
	Q5	25,3	1,4	0,4	0,5	22,6	1,5	0,3	0,5	19	0,9	3,3	0,8	6
	Q6	396,3	5,6	0,2	0,4	384,3	5,6	0,9	0,7	274,4	6	108,6	6,3	2
	Q7	28,9	2	0,8	0,4	26	1,9	0,2	0,4	25,6	2,1	0,1	0,3	9
	Q8	254,2	149,3	1,4	0,7	245,1	149	0,9	0,7	159,6	144,7	83,3	7,6	30
	Q9	3	3	1,2	0,6	0	0	0	0	0	0	0	0	0
	Q10	7,7	1,1	0,5	0,5	6	1,2	0,1	0,3	5,9	1,1	0	0	2
	Q11	6,5	2,9	0,1	0,3	0	0	0	0	0	0	0	0	0
	Q12	2,5	5,2	0,2	0,4	0	0	0	0	0	0	0	0	0
	Q14	348,2	7,1	0,4	0,5	341,5	7	0,6	0,5	264,3	5,1	76,5	5,1	1
LUBM10	Q1	4,6	0,7	0,1	0,3	3,1	0,3	0	0	3,1	0,3	0	0	1
	Q1a	7,9	0,5	0,4	0,5	5,9	0,5	0	0	3,5	0,5	2,4	0,5	1
	Q2	2,7	1,1	0,8	0,4	0	0	0	0	0	0	0	0	0
	Q3	4,6	1	0,2	0,4	3,2	0,7	0,1	0,3	3	0,8	0,1	0,3	1
	Q3a	4,5	1	0,2	0,4	2,7	0,6	0,1	0,3	2,6	0,7	0	0	1

	Povpra-ševanje	Skupni čas izvajanja		Procesiranje povpr.		Proženje storitev		Priprava vhoda		Čakanje na odgovor		Procesiranje izhoda		Št. klicev
		Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	Pvp	Dev	
LUBM10	Q4	107,6	8,9	0,9	0,3	103,8	8,6	1,2	0,9	100,2	8,7	2,3	1,9	37
	Q5	21,6	1,9	0,7	0,5	19,5	1,7	0	0	17,4	1,6	2	0,9	6
	Q6	710,7	140,9	0,2	0,4	688,5	140,7	0,8	0,4	476,4	142,7	210,7	25,5	2
	Q7	25,9	3,3	0,6	0,5	23,5	3	0,2	0,4	23,1	3,2	0,2	0,4	9
	Q8	200,1	13,3	1,5	0,5	190,4	11,8	1,1	0,7	105,5	6	83,2	8	30
	Q9	6,1	13,6	1,2	0,5	0	0	0	0	0	0	0	0	0
	Q10	7,4	0,5	0,3	0,5	6	0,4	0,1	0,3	5,8	0,6	0,1	0,3	2
	Q11	5,4	1,2	0,1	0,3	0	0	0	0	0	0	0	0	0
	Q12	1,1	0,3	0	0	0	0	0	0	0	0	0	0	0
	Q14	579,7	226,9	0,1	0,3	567,1	227,2	0,3	0,5	414,4	221,8	152,3	9,7	1
LUBM20	Q1	3,6	0,5	0,1	0,3	2,5	0,5	0	0	2,4	0,5	0,1	0,3	1
	Q1a	7,6	0,7	0,3	0,5	5,8	0,7	0	0	3,4	0,7	2,4	0,5	1
	Q2	2,7	1	1,2	0,6	0	0	0	0	0	0	0	0	0
	Q3	4,2	0,6	0	0	3,1	0,5	0	0	3,1	0,5	0	0	1
	Q3a	5,4	1,7	0,6	0,5	3,3	0,5	0	0	3,1	0,3	0,2	0,4	1
	Q4	109,8	9,1	0,9	0,3	106,4	8,8	0,5	0,7	102,8	8,6	3,1	2,4	37
	Q5	22,9	2,1	0,4	0,5	20,8	1,9	0	0	17,8	1,9	3	0,4	6
	Q6	1158,8	237,2	0,1	0,3	1113	237,1	0,6	0,7	680	215,1	431,8	35,1	2
	Q7	30,2	15,3	0,5	0,5	23	3,3	0	0	22,9	3,3	0,1	0,3	9
	Q8	195,2	15,6	1	0,4	185,3	14,4	0,6	1	104,9	5,7	79,4	10,1	30
	Q9	1,9	0,7	1,1	0,6	0	0	0	0	0	0	0	0	0
	Q10	6,8	0,4	0,4	0,5	5,6	0,5	0,1	0,3	5,3	0,8	0,2	0,4	2
	Q11	7,7	2,5	0,1	0,3	0	0	0	0	0	0	0	0	0
	Q12	0,6	0,5	0,1	0,3	0	0	0	0	0	0	0	0	0
	Q14	1074,6	215,5	0	0	1042,1	212,8	0,5	0,5	722,3	214,1	318,9	6,5	1

Tabela 4.7: Podroben prikaz izvajalnih časov povpraševanj za optimističen testni primer

4.3.3 Primerjalna analiza

Tabele 4.8, 4.9, 4.10 in 4.11 vsebujejo rezultate primerjave predlaganega pristopa z drugimi pristopi, ki so opisani v poglavju 4.2. Skladno z definiranimi pogoji eksperimenta smo vsako povpraševanje izvedli 10-krat. Tabele vsebujejo povprečni čas izvajanja povpraševanj v milisekundah.

Na osnovi rezultatov je lahko opazimo, da sta pristopa D2R in OSII po performansah precej podobna in se bistveno razlikujeta od preostalih pristopov. Pri preostalih pristopih je opaziti trend, da ima večanje selektivnosti povpraševanj negativen vpliv na hitrost izvajanja povpraševanj. To lahko opazimo

pri primerjavi povpraševanj Q1a in Q14. Obe povpraševanji povprašujeta po enaki vrsti podatkov (dodiplomskih študentih). Povpraševanje Q1a povprašuje po dodiplomskih študentih samo za en oddelek, medtem ko Q14 povprašuje po vseh dodiplomskih študentih. Opazimo lahko zanimivo karakteristiko, da sta D2R in OSII bistveno hitrejša za povpraševanje Q1a, medtem ko pri ostalih pristopih opazimo nasprotno situacijo. Čas izvajanja povpraševanja Q1a pri teh pristopih je v povprečju večji za faktor 3.

V kolikor primerjamo polnost odgovorov ugotovimo, da v večini primerov pristopi odgovorijo na povpraševanja v celoti ali pa sploh ne. To je zaradi tega, ker posamezna povpraševanja preizkušajo le določen aksiom sklepanja. V kolikor pristop podpira ta aksiom vrne poln rezultat, v nasprotnem primeru pa ne vrne ničesar. Izjema sta povpraševanji Q8 in Q9, za katera nekateri pristopi vrnejo delne rezultate.

Iz rezultatov je opaziti, da imajo obstoječi pristopi, z izjemo D2R, precej slabo skalabilnost. Povpraševanja Q1, Q1a, Q3, Q3a, Q4, Q5, Q7, Q8 in Q10 vračajo enake rezultate ne glede na velikost podatkovne množice. Medtem ko OSII in D2R ohranjata učinkovitost izvajanja teh povpraševanj ne glede na velikost podatkovne množice, pri ostalih pristopih temu ni tako. Pri teh pristopih lahko za ta povpraševanja opazimo linearno naraščanje časa izvajanja z večanjem podatkovne množice, kar govori o slabi skalabilnosti teh pristopov.

Rezultati meritev nakazujejo, da je pristop, ki smo ga razvili, pri določenih povpraševanjih hitrejši od klasičnih pristopov, ki ne podpirajo sklepanja. Pristop OSII je pogosto tudi hitrejši od pristopov, ki celotno podatkovno množico hranijo v pomnilniku. V kolikor primerjamo razvit pristop s pristopi, ki podpirajo sklepanje, lahko na osnovi rezultatov vidimo, da je naš pristop bistveno učinkovitejši. To je zaradi tega, ker pristop OSII opravlja sklepanje na nivoju sheme in ga opravi v času procesiranja povpraševanja, medtem ko ostali pristopi opravljajo sklepanju na nivoju podatkov.

Na osnovi povpraševanj Q1, Q1a in Q14 lahko opazimo, da ima pristop OSII, glede na velikost rezultata, boljšo skalabilnost kot D2R. Število odgovorov, ki jih vrne povpraševanje, ima za pristop D2R večji negativni vpliv na skupen čas izvajanja povpraševanja kot za OSII. Pri povpraševanjih, ki vračajo veliko število odgovorov, so najhitrejši klasični pristopi, ki temeljijo na Jeni. To lastnost

opazimo tako pri pristopih, ki imajo podatkovno množico v pomnilniku (Jena_M, Jena_P, Jena_R, Jena_O), kot pri tistih, kjer se podatkovna množica nahaja na trdem disku (TDB in SDB).

Pri primerjavi karakteristik pogosto izstopa pristop D2R_R, ki ga ne moremo uvrstiti ne med klasične pristope, ki temeljijo na Jeni, kot tudi ne med OSII in D2R. D2R_R pristop je klasičen D2R pogon, ki je ovit z RDFS pogonom sklepanja. Zaradi tega pogona sklepanja sistem določene podatke hrani v pomnilniku, do določenih pa dostopa neposredno preko pogona D2R v času izvajanja povpraševanja. Zaradi kombinacije uporabe pomnilnika in neposrednega dostopa do podatkov prihaja do odstopanj v primerjavi z drugimi pristopi.

		Jena	Jena_P	Jena_R	Jena_O	TDB	SDB	D2R	D2R_R	OSII_P	OSII_O
Q1	Čas(ms)	7	18	11	11	22	303	4	770	5	6
	Št. odg	4	4	4	4	4	4	4	4	4	4
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q1a	Čas(ms)	13	49	80	41	33	958	18	6.729	23	24
	Št. odg	532	532	532	532	532	532	532	532	532	532
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q2	Čas(ms)	15	>10 m	134.277	140.545	38	>10 m	75	>10 m	4	5
	Št. odg	0		0	0	0		0		0	0
	Polnost	100%		100%	100%	100%		100%		100%	100%
Q3	Čas(ms)	14	37	28	27	11	681	2	2.959	5	7
	Št. odg	6	6	6	6	6	6	6	6	6	6
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q3a	Čas(ms)	11	64	94	50	16	2.144	4	2.434	10	8
	Št. odg	6	6	6	6	6	6	6	6	6	6
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q4	Čas(ms)	1	15	7	1.133	1	1	1	604	130	144
	Št. odg	0	34	34	34	0	0	0	34	34	34
	Polnost	0%	100%	100%	100%	0%	0%	0%	100%	100%	100%
Q5	Čas(ms)	0	47	53	52	1	1	1	6.808	27	26
	Št. odg	0	719	719	719	0	0	0	719	719	719
	Polnost	0%	100%	100%	100%	0%	0%	0%	100%	100%	100%
Q6	Čas(ms)	0	54	8	19	0	1	1	5	138	39
	Št. odg	0	7.790	6.463	7.790	0	0	0	6.463	7.790	7.790
	Polnost										
Q7	Čas(ms)	1	78.741	29.100	37.892	2	1	1	>10 m	29	31
	Št. odg	0	67	61	67	0	0	0		67	67
	Polnost	0%	100%	91%	100%	0%	0%	0%		100%	100%
Q8	Čas(ms)	1	32.545	596	800	1	1	1	224.744	17.935	213
	Št. odg	0	7.790	6.463	7.790	0	0	0	5.916	7.790	7.790
	Polnost	0%	100%	83%	100%	0%	0%	0%	76%	100%	100%
Q9	Čas(ms)	1	>10 m	>10 m	>10 m	1	1	1	>10 m	4	3

		Jena	Jena_P	Jena_R	Jena_O	TDB	SDB	D2R	D2R_R	OSII_P	OSII_O
	Št. odg	0				0	0	0		0	0
	Polnost	0%				0%	0%	0%		0%	0%
Q10	Čas(ms)	0	77	28	33	1	1	1	3.130	6	9
	Št. odg	0	4	0	4	0	0	0	0	4	4
	Polnost	0%	100%	0%	100%	0%	0%	0%	0%	100%	100%
Q11	Čas(ms)	1	12	1	28	2	25	1	1	2	6
	Št. odg	0	224	0	224	0	0	0	0	0	0
	Polnost	0%	100%	0%	100%	0%	0%	0%	0%	0%	0%
Q12	Čas(ms)	1	103	1	2	1	1	1	0	1	2
	Št. odg	0	15	0	15	0	0	0	0	0	0
	Polnost	0%	100%	0%	100%	0%	0%	0%	0%	0%	0%
Q13	Čas(ms)	1	36	25	45	3	8	1	1.625	4	4
	Št. odg	0	1	0	1	0	0	0	0	0	0
	Polnost	0%	100%	0%	100%	0%	0%	0%	0%	0%	0%
Q14	Čas(ms)	3	17	11	11	3	14	27	36	69	23
	Št. odg	5.916	5.916	5.916	5.916	5.916	5.916	5.916	5.916	5.916	5.916
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Tabela 4.8: Kvantitativna primerjava obstoječih pristopov za podatkovno množico LUBM1

		Jena	Jena_P	Jena_R	Jena_O	TDB	SDB	D2R	D2R_R	OSII_P	OSII_O
Q1	Čas(ms)	35	115	65	68	20	1.561	1	5.193	6	5
	Št. odg	4	4	4	4	4	4	4	4	4	4
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q1a	Čas(ms)	69	268	237	154	87	11.635	4	74.926	22	8
	Št. odg	532	532	532	532	532	532	532	532	532	532
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q2	Čas(ms)	80	>10 m	>10 m	>10 m	89	>10 m	451	>10 m	4	4
	Št. odg	9				9		9		0	0
	Polnost	100%				100%		100%		0%	0%
Q3	Čas(ms)	58	237	149	146	102	11.463	4	43.096	5	5
	Št. odg	6	6	6	6	6	6	6	6	6	6
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q3a	Čas(ms)	58	389	252	244	59	29.740	7	42.909	10	7
	Št. odg	6	6	6	6	6	6	6	6	6	6
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q4	Čas(ms)	1	77	23	26	3	4	1	3.042	128	119
	Št. odg	0	34	34	34	0	0	0	34	34	34
	Polnost	0%	100%	100%	100%	0%	0%	0%	100%	100%	100%
Q5	Čas(ms)	8	336	336	556	3	4	1	90.320	28	26
	Št. odg	0	719	719	146	0	0	0	719	719	719
	Polnost	0%	100%	100%	20%	0%	0%	0%	100%	100%	100%
Q6	Čas(ms)	0	294	43	43	5	4	1	32	777	400

		Jena	Jena_P	Jena_R	Jena_O	TDB	SDB	D2R	D2R_R	OSII_P	OSII_O
	Št. odg	0	48.582	40.087	40.087	0	0	0	40.087	48.582	48.582
	Polnost	0%	100%	83%	83%	0%	0%	0%	83%	100%	100%
Q7	Čas(ms)	1	>10 m	>10 m	>10 m	3	3	1	>10 m	31	29
	Št. odg	0				0	0	0		67	67
	Polnost	0%				0%	0%	0%		100%	100%
Q8	Čas(ms)	4	>10 m	82.394	11.955	1	1	1	>10 m	17.415	192
	Št. odg	0		6.463	6.463	0	0	0		7.790	7.790
	Polnost	0%		83%	83%	0%	0%	0%		100%	100%
Q9	Čas(ms)	1	>10 m	>10 m	>10 m	1	1	0	>10 m	3	2
	Št. odg	0				0	0	0		0	0
	Polnost	0%				0%	0%	0%		0%	0%
Q10	Čas(ms)	1	542	158	162	1	1	1	42.266	9	8
	Št. odg	0	4	0	0	0	0	0	0	4	4
	Polnost	0%	100%	0%	0%	0%	0%	0%	0%	100%	100%
Q11	Čas(ms)	3	69	5	6	8	175	1	2	7	4
	Št. odg	0	224	0	0	0	0	0	0	0	0
	Polnost	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%
Q12	Čas(ms)	1	3.218	1	0	4	3	0	1	2	1
	Št. odg	0	15	0	0	0	0	0	0	0	0
	Polnost	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%
Q13	Čas(ms)	0	348	816	335	2	4	0	10.009	4	4
	Št. odg	0	21	0	0	0	0	0	0	0	0
	Polnost	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%
Q14	Čas(ms)	25	131	50	51	37	368	848	116	434	350
	Št. odg	36.682	36.682	36.682	36.682	36.682	36.682	36.682	36.682	36.682	36.682
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Tabela 4.9: Kvantitativna primerjava obstoječih pristopov za podatkovno množico LUBM5

		Jena	Jena_P	Jena_R	Jena_O	TDB	SDB	D2R	D2R_R	OSII_P	OSII_O
Q1	Čas(ms)	67	296	129	134	34	7615	1	26936	5	5
	Št. odg	4	4	4	4	4	4	4	4	4	4
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q1a	Čas(ms)	138	557	513	334	213	23934	35	144861	6	7
	Št. odg	532	532	532	532	532	532	532	532	532	532
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q2	Čas(ms)	173	>10 m	>10 m	>10 m	168	>10 m	1031	>10 m	3	4
	Št. odg	28				28		28		0	0
	Polnost	100%				100%		100%		0%	0%
Q3	Čas(ms)	121	632	353	317	239	23542	5	86399	4	4
	Št. odg	6	6	6	6	6	6	6	6	6	6
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q3a	Čas(ms)	110	848	575	573	182	60622	5	85864	8	5

		Jena	Jena_P	Jena_R	Jena_O	TDB	SDB	D2R	D2R_R	OSII_P	OSII_O
	Št. odg	6	6	6	6	6	6	6	6	6	6
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q4	Čas(ms)	18	119	45	34	2	4	1	11626	101	116
	Št. odg	0	34	34	34	0	0	0	34	34	34
	Polnost	0%	100%	100%	100%	0%	0%	0%	100%	100%	100%
Q5	Čas(ms)	0	798	752	1903	1	5	1	175105	24	26
	Št. odg	0	719	719	146	0	0	0	719	719	719
	Polnost	0%	100%	100%	20%	0%	0%	0%	100%	100%	100%
Q6	Čas(ms)	0	654	94	94	1	2	1	57	1513	663
	Št. odg	0	99566	82507	82507	0	0	0	82507	99566	99566
	Polnost	0%	100%	83%	83%	0%	0%	0%	83%	100%	100%
Q7	Čas(ms)	1	>10 m	>10 m	>10 m	7	1	1	>10 m	26	30
	Št. odg	0				0	0	0		67	67
	Polnost	0%				0%	0%	0%		100%	100%
Q8	Čas(ms)	1	>10 m	>10 m	50300	1	1	1	>10 m	16652	188
	Št. odg	0			6463	0	0	0		7790	7790
	Polnost	0%			83%	0%	0%	0%		100%	100%
Q9	Čas(ms)	10	>10 m	>10 m	>10 m	1	1	1	>10 m	3	5
	Št. odg	0				0	0	0		0	0
	Polnost	0%				0%	0%	0%		0%	0%
Q10	Čas(ms)	1	1093	365	376	1	1	1	88753	8	7
	Št. odg	0	4	0	0	0	0	0	0	4	4
	Polnost	0%	100%	0%	0%	0%	0%	0%	0%	100%	100%
Q11	Čas(ms)	5	161	10	11	12	400	0	2	6	5
	Št. odg	0	224	0	0	0	0	0	0	0	0
	Polnost	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%
Q12	Čas(ms)	13	13922	1	0	4	3	0	0	1	1
	Št. odg	0	15	0	0	0	0	0	0	0	0
	Polnost	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%
Q13	Čas(ms)	0	860	>10 m	976	4	8	1	20706	4	4
	Št. odg	0	33		0	0	0	0	0	0	0
	Polnost	0%	100%		0%	0%	0%	0%	0%	0%	0%
Q14	Čas(ms)	37	243	110	122	95	609	1789	231	887	491
	Št. odg	75547	75547	75547	75547	75547	75547	75547	75547	75547	75547
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Tabela 4.10: Kvantitativna primerjava obstoječih pristopov za podatkovno množico LUBM10

		Jena	Jena_P	Jena_R	Jena_O	TDB	SDB	D2R	D2R_R	OSII_P	OSII_O
Q1	Čas(ms)	136	593	291	369	68	7990	1	57611	4	4
	Št. odg	4	4	4	4	4	4	4	4	4	4
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q1a	Čas(ms)	317	1471	1237	857	369	49967	64	>10 m	7	7

		Jena	Jena_P	Jena_R	Jena_O	TDB	SDB	D2R	D2R_R	OSII_P	OSII_O
	Št. odg	532	532	532	532	532	532	532		532	532
	Polnost	100%	100%	100%	100%	100%	100%	100%		100%	100%
Q2	Čas(ms)	374	>10 m	>10 m	>10 m	360	>10 m	2276	>10 m	3	2
	Št. odg	59				59		59		0	0
	Polnost	100%				100%		100%		0%	0%
Q3	Čas(ms)	274	1138	731	765	423	24584	8	183217	5	4
	Št. odg	6	6	6	6	6	6	6	6	6	6
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q3a	Čas(ms)	273	1728	1198	1153	347	126555	6	178565	8	4
	Št. odg	6	6	6	6	6	6	6	6	6	6
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Q4	Čas(ms)	10	275	110	76	2	1	1	24250	98	108
	Št. odg	0	34	34	34	0	0	0	34	34	34
	Polnost	0%	100%	100%	100%	0%	0%	0%	100%	100%	100%
Q5	Čas(ms)	1	1517	1516	7149	4	1	0	>10 m	27	23
	Št. odg	0	719	719	146	0	0	0		719	719
	Polnost	0%	100%	100%	20%	0%	0%	0%		100%	100%
Q6	Čas(ms)	1	1414	199	207	3	1	0	123	3023	1035
	Št. odg	0	210603	174750	174750	0	0	0	174750	210603	210603
	Polnost	0%	100%	83%	83%	0%	0%	0%	83%	100%	100%
Q7	Čas(ms)	1	>10 m	>10 m	>10 m	5	1	0	>10 m	18	27
	Št. odg	0				0	0	0		67	67
	Polnost	0%				0%	0%	0%		100%	100%
Q8	Čas(ms)	1	>10 m	>10 m	222239	1	1	0	>10 m	16683	195
	Št. odg	0			6463	0	0	0		7790	7790
	Polnost	0%			83%	0%	0%	0%		100%	100%
Q9	Čas(ms)	21	>10 m	>10 m	>10 m	1	1	0	>10 m	3	2
	Št. odg	0				0	0	0		0	0
	Polnost	0%				0%	0%	0%		0%	0%
Q10	Čas(ms)	1	2487	734	729	0	1	12	190496	5	7
	Št. odg	0	4	0	0	0	0	0	0	4	4
	Polnost	0%	100%	0%	0%	0%	0%	0%	0%	100%	100%
Q11	Čas(ms)	10	412	22	24	18	411	1	4	8	7
	Št. odg	0	224	0	0	0	0	0	0	0	0
	Polnost	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%
Q12	Čas(ms)	8	65979	1	1	2	1	1	1	1	1
	Št. odg	0	15	0	0	0	0	0	0	0	0
	Polnost	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%
Q13	Čas(ms)	0	3207	>10 m	3417	2	5	1	43665	5	4
	Št. odg	0	86		0	0	0	0	0	0	0
	Polnost	0%	100%		0%	0%	0%	0%	0%	0%	0%
Q14	Čas(ms)	76	503	230	229	195	485	3913	561	1776	907
	Št. odg	160120	160120	160120	160120	160120	160120	160120	160120	160120	160120
	Polnost	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Tabela 4.11: Kvantitativna primerjava obstoječih pristopov za podatkovno množico LUBM20

4.3.4 Statistična analiza

Izvedli smo statistično analizo rezultatov meritev, ki smo jih predstavili v prejšnjem poglavju. Izvedli smo Friedmanov, Wilcoxonov in nato še Bonferroni-Dunnov test. Friedmanov test je neparametrični test za ugotavljanje signifikantnosti razlik med različnimi pristopi. Za preverjanje signifikantnosti razlik med pristopi se pogosto uporabljata tudi ANOVA in T-test, ki pa oba predvidevata normalno porazdelitev podatkov. Ker v našem primer podatki niso normalno porazdeljeni je ustreznejši Friedmanov test [21].

Friedmanov test vsakemu pristopu za vsak testni primer določi rang tako, da ima najučinkovitejši algoritem za posamezni primer rang 1, drugi najučinkovitejši algoritem rang 2,... Na osnovi porazdelitve rangov lahko izračunamo, ali obstaja signifikantna razlika med pristopi. Friedmanov test je neparametrični test, rezultati naših meritev pa vsebujejo dva parametra, namreč čas izvajanja in število odgovorov. V kolikor za Friedmanov test uporabimo samo čas izvajanja, ne dobimo ustreznih rezultatov, saj takšen test ne upošteva pravilnosti odgovora povpraševanja. Npr. pristop Jena bi za povpraševanje Q5 na LUBM1 testni množici dobil najvišji rang, saj se najhitreje izvede, čeprav ne vrne nobenega rezultata. Da bi upoštevali pravilnost odgovorov, smo obtežili podatke na naslednji način:

- v kolikor povpraševanje vrne popolni rezultat(vse možne odgovore), ohranimo čas nespremenjen
- v kolikor povpraševanje ne vrne popolnega rezultata, mu prištejemo najvišji čas izvajanja tega povpraševanja na isti podatkovni množici,
- v kolikor je prišlo do prekoračitve desetminutne časovne omejitve, ki smo jo določili v okviru eksperimenta, mu določimo čas omejitve 600000.

S to obtežitvijo smo uspeli doseči, da neparametrični Friedmanov test upošteva tudi pravilnost odgovora. Na osnovi uteži, dobijo višji rang pristopi, ki so uspeli v celoti odgovoriti na povpraševanje, sledijo pristopi, ki so podali odgovor, a ta ni bil popoln ter nato pristopi, pri katerih je prišlo do časovne omejitve. Znotraj posameznih kategorij so pristopi še zmeraj rangirani glede na čas izvaja-

nja. V tabeli 4.12 so prikazani združeni uteženi rezultati, ki smo jih uporabili za statistično analizo (vse vrednosti v tabeli so podane v ms).

		Jena	Jena_P	Jena_R	Jena_O	TDB	SDB	D2R	D2R_R	OSII_P	OSII_O
LUBM1	Q1	7	18	11	11	22	303	4	770	5	6
	Q1a	13	49	80	41	33	958	18	6729	23	24
	Q2	15	600000	134277	140545	38	600000	75	600000	4	5
	Q3	14	37	28	27	11	681	2	2959	5	7
	Q3a	11	64	94	50	16	2144	4	2434	10	8
	Q4	1134	15	7	1133	1135	1134	1134	604	130	144
	Q5	6808	47	53	52	6809	6809	6809	6808	27	26
	Q6	139	192	146	157	138	139	139	143	276	177
	Q7	78741	78741	107841	37892	78743	78742	78741	600000	29	31
	Q8	224744	32545	225339	800	224745	224745	224745	449488	17935	213
	Q9	5	600000	600000	600000	5	5	4	600000	7	6
	Q10	3131	77	3158	33	3132	3131	3131	6261	6	9
	Q11	29	12	30	28	30	53	29	29	30	34
	Q12	104	103	104	2	104	105	104	104	104	105
	Q13	1626	36	1650	45	1629	1633	1626	3251	1629	1629
	Q14	3	17	11	11	3	14	27	36	69	23
LUBM5	Q1	35	115	65	68	20	1561	1	5193	6	5
	Q1a	69	268	237	154	87	11635	4	74926	22	8
	Q2	80	600000	600000	600000	89	600000	451	600000	455	454
	Q3	58	237	149	146	102	11463	4	43096	5	5
	Q3a	58	389	252	244	59	29740	7	42909	10	7
	Q4	3042	77	23	26	3045	3046	3043	3042	128	119
	Q5	90328	336	336	90875	90322	90323	90321	90320	28	26
	Q6	778	294	820	820	782	781	778	809	777	400
	Q7	32	600000	600000	600000	34	33	32	600000	31	29
	Q8	82398	600000	164788	94349	82395	82395	82395	600000	17415	192
	Q9	3	600000	600000	600000	4	3	3	600000	5	5
	Q10	42266	542	42423	42428	42266	42266	42266	84531	9	8
	Q11	178	69	180	181	182	349	175	176	181	179
	Q12	3219	3218	3219	3219	3222	3222	3219	3219	3220	3220
	Q13	10009	348	10825	10344	10011	10013	10009	20017	10012	10013
	Q14	25	131	50	51	37	368	848	116	434	350
LUBM10	Q1	67	296	129	134	34	7615	1	26936	5	5
	Q1a	138	557	513	334	213	23934	35	144861	6	7
	Q2	173	600000	600000	600000	168	600000	1031	600000	1034	1035
	Q3	121	632	353	317	239	23542	5	86399	4	4
	Q3a	110	848	575	573	182	60622	5	85864	8	5
	Q4	11644	119	45	34	11628	11630	11627	11626	101	116
	Q5	175106	798	752	177008	175106	175110	175106	175105	24	26
	Q6	1513	654	1607	1606	1513	1514	1513	1570	1513	663
	Q7	30	600000	600000	600000	36	31	30	600000	26	30
	Q8	50300	600000	600000	100599	50300	50300	50300	600000	16652	188
	Q9	21	600000	600000	600000	12	11	12	600000	14	15
	Q10	88753	1093	89117	89128	88754	88753	88753	177505	8	7

		Jena	Jena_P	Jena_R	Jena_O	TDB	SDB	D2R	D2R_R	OSII_P	OSII_O
LUBM10	Q11	405	161	410	411	411	799	400	401	406	404
	Q12	13936	13922	13923	13923	13926	13925	13923	13923	13923	13923
	Q13	20706	860	600000	21682	20710	20714	20707	41412	20710	20710
	Q14	37	243	110	122	95	609	1789	231	887	491
LUBM20	Q1	136	593	291	369	68	7990	1	57611	4	4
	Q1a	317	1471	1237	857	369	49967	64	600000	7	7
	Q2	374	600000	600000	600000	360	600000	2276	600000	2279	2278
	Q3	274	1138	731	765	423	24584	8	183217	5	4
	Q3a	273	1728	1198	1153	347	126555	6	178565	8	4
	Q4	24260	275	110	76	24252	24251	24251	24250	98	108
	Q5	7149	1517	1516	14297	7152	7150	7149	600000	27	23
	Q6	3024	1414	3222	3230	3026	3024	3023	3146	3023	1035
	Q7	28	600000	600000	600000	32	28	28	600000	18	27
	Q8	222240	600000	600000	444478	222241	222240	222240	600000	16683	195
	Q9	41	600000	600000	600000	21	22	21	600000	23	23
	Q10	190496	2487	191230	191224	190496	190497	190507	380992	5	7
	Q11	422	412	434	437	430	823	413	416	420	420
	Q12	65987	65979	65980	65980	65982	65980	65980	65980	65981	65980
	Q13	43666	3207	600000	47082	43667	43670	43666	87330	43670	43669
	Q14	76	503	230	229	195	485	3913	561	1776	907

Tabela 4.12: Uteženi časi izvajanja povpraševanj (v ms)

Na osnovi uteženih rezultatov smo izvedli dva testa raznolikosti:

- **Celotna množica povpraševanj**: uporabili smo celotno množico povpraševanj, rezultate iz različnih podatkovnih množic smo združili tako, da smo rezultate posameznih podatkovnih množic nanizali drugega za drugim. Na tak način smo dobili za vsak pristop 64 meritev (16 za vsako podatkovno množico).
- **Povpraševanja brez sklepanja**: uporabili smo samo povpraševanja, ki ne zahtevajo sklepanja (to so povpraševanja Q1, Q1a, Q2, Q3, Q3a in Q14). Podobno kot za celotno množico povpraševanj smo rezultate iz različnih podatkovnih množic združili, tako smo za vsak pristop dobili 24 meritev (6 meritev za vsako podatkovno množico).

Razdelitev na dva testna primera nam je omogočila testiranje raznolikosti za osnovna scenarija, to je za povpraševanje nad podatki s sklepanjem, ter za povpraševanje brez sklepanja.

Tabela 4.13 prikazuje rezultate Friedmanovega testa za celotno množico povpraševanj, medtem ko tabela 4.14 prikazuje rezultate Friedmanovega testa za množico povpraševanj brez sklepanja. Rezultati obeh testov so pokazali, da obstaja signifikantna razlika med pristopi, ki smo jih primerjali.

Pristop	Srednji rang
Jena	4,52
Jena_P	5,41
Jena_R	6,78
Jena_O	6,21
TDB	5,52
SDB	7,26
D2R	3,98
D2R_R	8,00
OSII_P	3,94
OSII_O	3,39

(a) Razvrstitev pristopov

Karakteristika	Vrednost
N	64
χ^2	153,599
df	9
Sig	0,000

(b) Statistika testa

Tabela 4.13: Friedmanov test za celotno množico povpraševanj

Pristop	Srednji rang
Jena	3,25
Jena_P	7,58
Jena_R	6,38
Jena_O	6,00
TDB	3,92
SDB	8,46
D2R	3,29
D2R_R	9,13
OSII_P	3,79
OSII_O	3,21

(a) Razvrstitev pristopov

Karakteristika	Vrednost
N	24
χ^2	127,377
df	9
Sig	0,000

(b) Statistika testa

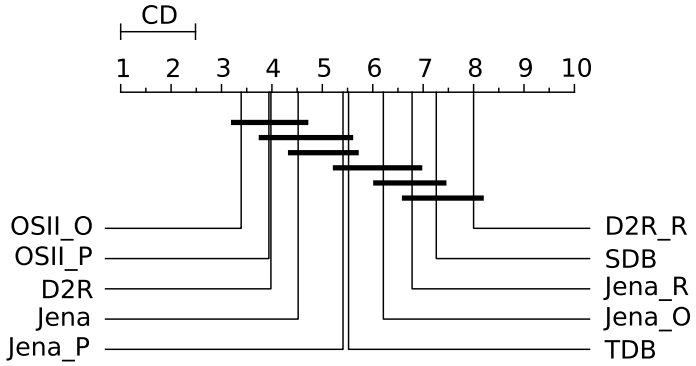
Tabela 4.14: Friedmanov test za množico povpraševanj brez sklepanja

Za tem, ko smo izvedli Friedmanov test, s katerim smo pokazali signifikantno raznolikost med pristopi, smo izvedli Bonferroni-Dunnov test. S tem testom

izračunamo kritično razdaljo CD, na osnovi katere lahko primerjamo posamezne pristope. V kolikor se ranga dveh pristopov razlikujeta za več, kot je vrednost kritične razdalje CD, potem lahko trdimo, da sta ta pristopa signifikantno različna [21]. Učinkovitejši je pristop, ki ima nižji rang. Enačba 4.1 podaja formulo za izračun kritične razdalje, kjer je *k* število pristopov, *N* število meritev, q_a pa ima za $N = 10$ ter $p < 0,05$ ob uporabi Bonferroni-Dunnove korekcije vrednost $2,773$[21]. Na osnovi te formule smo dobili kritično razdaljo za celotno množico povpraševanj $CD_1 = 1,485$ ter kritično razdaljo za množico povpraševanj brez sklepanja $CD_2 = 2,424$ pri signifikantnosti $p < 0,05$.

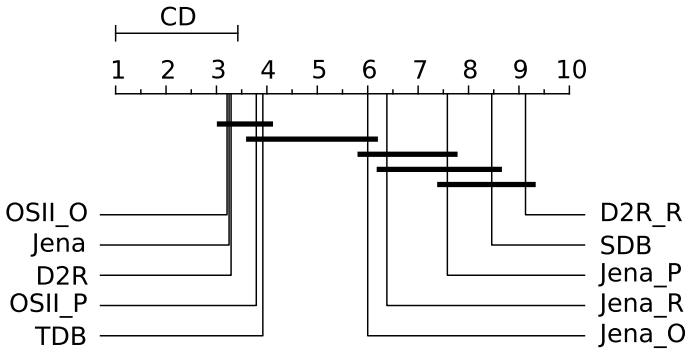
$$CD = q_a \sqrt{\frac{k(k+1)}{6N}} \tag{4.1}$$

Slika 4.4 prikazuje grafični prikaz Bonferroni-Dunnovega testa za celotno množico povpraševanj, medtem ko slika 4.5 prikazuje rezultate za množico povpraševanj brez sklepanja. Pristopi, katerih učinkovitost ne odstopa signifikantno, so med seboj povezani z odebeljenimi črtami pod osjo. Vrednosti na osi predstavljajo rang, ki je bil izračunan na osnovi Friedmanovega testa. Pristopi, ki so izven označega intervala, so signifikantno različni od pristopov, ki jih interval povezuje ($p < 0,05$).



Slika 4.4: Rezultati Bonferroni-Dunnovega testa za celotno množico povpraševanj ($p < 0,05$)

Rezultati Bonferroni-Dunnovega testa so pokazali, da noben pristop ni signi-



Slika 4.5: Rezultati Bonferroni-Dunnovega testa za množico povpraševanj brez sklepanja ($p < 0,05$)

fikantno boljši niti od optimističnega (OSII_O), niti od pesimističnega (OSII_P) testnega primera za pristop, ki smo ga razvili. Ravno nasprotno, na osnovi Friedmanovega testa je bil pristopu OSII_O dodeljen najvišji rang za obe množici povpraševanj. Pri celotni množici povpraševanj se ni OSII_O signifikantno razlikoval zgolj od OSII_P, D2R in Jene. Od vseh ostalih pristopov je OSII_O signifikantno učinkovitejši za celotno množico povpraševanj. Tudi OSII_P se je izkazal za zelo učinkovitega, saj je za celotno množico povpraševanj dobil drugi najvišji rang, signifikantno pa se ne razlikuje zgolj od OSII_O, D2R, Jena, Jena_P.

Izmed vseh pristopov, od katerih se OSII_O in OSII_P signifikantno ne razlikujeta, samo Jena_P omogoča sklepanje. Ta pristop v primerjavi z ostalimi pristopi, ki omogočajo sklepanje, izvaja povpraševanja precej počasi. Visok rang je pridobil zaradi tega, ker je uspel podati celotne odgovore na največ povpraševanj.

Za množico povpraševanj, ki ne zahtevajo sklepanja, opazimo enak vzorec kot pri celotni množici povpraševanj. OSII_O ima glede na Friedmanov test najvišji rang, signifikantno se ne razlikuje zgolj od najhitrejših alternativnih pristopov (Jena, D2R, TDB in OSII_P). Pristop OSII_P je dobil četrti najvišji rang in se signifikantno ne razlikuje od pristopov OSII_O, Jena, D2R, TDB in Jena_O. Slednji je na meji kritične razdalje in je signifikantno različen za $p < 0,1$.

Za validacijo Bonferroni-Dunnovega testa smo opravili še Wilcoxon-ov test z upoštevanjem Bonferronijeve korekcije. Rezultati za celotno množico povpraše-

vanj so prikazani v tabeli 4.15, rezultati za množico povpraševanj brez sklepanja pa so prikazani v tabeli 4.16.

Wilcoxonov test je pokazal, da med pristopoma OSII_O in OSII_P ni signifikantnih razlik. Edini pristop, od katerega se OSII_O in OSII_P signifikantno ne razlikujeta za celotno množico povpraševanj je samo D2R. Od vseh ostalih pristopov sta OSII_O in OSII_P signifikantno učinkovitejša.

Rezultati Wilcoxonovega testa so podobni rezultatom Bonferroni-Dunnovega testa tudi za množico povpraševanj, ki ne zahtevajo sklepanja. Oba pristopa sta absolutno učinkovitejša od vseh ostalih pristopov, vendar se signifikantno ne razlikujeta od pristopov Jena, TDB in D2R. Dodatno se OSII_P signifikantno ne razlikuje še od pristopov Jena_R in Jena_O, vendar je pri obeh primerih signifikanca zelo nizka ($0,55$ in $0,65$).

Pristop	Z	Sig
Jena in OSII_O	-4,063	,000
Jena_P in OSII_O	-4,227	,000
Jena_R in OSII_O	-5,664	,000
Jena_O in OSII_O	-5,395	,000
TDB in OSII_O	-4,488	,000
SDB in OSII_O	-6,260	,000
D2R in OSII_O	-2,719	,059
D2R_R in OSII_O	-6,420	,000

(a) Optimističen primer

Pristop	Z	Sig
Jena in OSII_P	-3,651	,002
Jena_P in OSII_P	-3,531	,004
Jena_R in OSII_P	-5,410	,000
Jena_O in OSII_P	-4,855	,000
TDB in OSII_P	-4,180	,000
SDB in OSII_P	-5,845	,000
D2R in OSII_P	-1,926	,487
D2R_R in OSII_P	-6,239	,000

(b) Pesimističen primer

Tabela 4.15: Wilcoxonov test za celotno množico povpraševanj z upoštevanjem Bonferronijeve korekcije

Pristop	Z	Sig
Jena in OSII_O	-,286	1,000
Jena_P in OSII_O	-3,429	,005
Jena_R in OSII_O	-3,000	,024
Jena_O in OSII_O	-3,000	,024
TDB in OSII_O	-,571	1,000
SDB in OSII_O	-4,114	,000
D2R in OSII_O	-,415	1,000
D2R_R in OSII_O	-4,029	,001

(a) Optimističen primer

Pristop	Z	Sig
Jena in OSII_P	-,171	1,000
Jena_P in OSII_P	-2,943	,029
Jena_R in OSII_P	-2,743	,055
Jena_O in OSII_P	-2,686	,065
TDB in OSII_P	-,343	1,000
SDB in OSII_P	-3,914	,001
D2R in OSII_P	-,371	1,000
D2R_R in OSII_P	-3,971	,001

(b) Pesimističen primer

Tabela 4.16: Wilcoxonov test za množico povpraševanj brez sklepanja z upoštevanjem Bonferronijeve korekcije

Poglavje 5

Razprava

V okviru doktorske disertacije smo raziskali smiselnost uporabe storitev za integracijo heterogenih podatkovnih virov na osnovi ontologij. Osnovne prednosti, ki jih prinašajo ontologije v okviru integracije podatkovnih virov, so:

- predstavitev podatkovnega modela z ontologijo je človeku bližje kot relacijski model, ki je bolj primeren za strojno procesiranje,
- globalna shema je predstavljena eksplicitno, posledično so eksplicitno predstavljeni tudi posamezni koncepti ter relacije med njimi,
- ontologije omogočajo razmeroma enostavno razširljivost – v kolikor želimo v globalno shemo vključiti nove stvari, v večini primerov ni potrebno posegati v obstoječo strukturo,
- predstavitev podatkov z ontologijo je precej bolj naravna kot npr. z relacijskimi bazami, zaradi tega je v večini primerov enostavneje vključiti podatke iz »nerelacijskih virov« (npr. datoteke, el. pošta, ipd.).

Opravili smo pregled literature o integraciji podatkov na nivoju ontologij ter pregled literature o semantičnem označevanju storitev. Na osnovi rezultatov pregleda literature smo razvili model za integracijo podatkovnih virov na osnovi ontologij imenovan OSII. OSII vsebuje formalni model za integracijo podatkovnih virov, arhitekturo sistema za integracijo podatkovnih virov ter algoritme za transformacijo povpraševanj, ki omogočajo samodejno kompozicijo ter proženje storitev v času izvajanja.

OSII je hibriden LAV pristop za integracijo podatkovnih virov. Hibriden je zato, ker omogoča materializacijo podatkov. OSII v času povpraševanja samodejno pridobiva podatke iz spletnih storitev ter jih integrira na osnovi globalne ontologije. Del podatkov je lahko tudi materializiran v lokalni RDF podatkovni shrambi. Povpraševanje po integriranih podatkih poteka na osnovi globalne ontologije v sparql povpraševalnem jeziku.

Ključni del OSII modela so algoritmi, ki omogočajo povpraševanje po podatkih, ki so distribuirani med storitvami. Sparql algebro smo razširili z novim operatorjem web_service, ki omogoča samodejno proženje storitev znotraj samega povpraševanja. Definirali smo transformacijo, ki na osnovi preslikav storitev transformira vhodno sparql povpraševanje v novo povpraševanje, ki vsebuje klice storitev ter pripravo vhodnih podatkov storitev. Na osnovi vhodnega sparql povpraševanja, OSII v času izvajanja identificira storitve, izvede kompozicijo storitev ter pripravi vhodne podatke. Rezultat transformacije povpraševanja je načrt proženja podan v sparql algebri, ki ga lahko izvede standardni sparql pogon povpraševanja, ki ima podano implementacijo za operator web_service. OSII omogoča sklepanje na podmnožici profila OWL 2 QL.

V nadaljevanju podajamo analizo hipotez, ki smo jih postavili v okviru doktorske disertacije.

H1

Semantično označene storitve omogočajo distribuirano in dinamično pridobivanje podatkov v sistemih za integracijo podatkovnih virov na osnovi ontologij.

Eno izmed ključnih raziskovalnih vprašanj v okviru naše doktorske disertacije je bilo, ali je mogoče storitve uporabiti v sistemu za integracijo podatkovnih virov na osnovi ontologij na povsem transparenten način. Sistem za integracijo bi tako v takšnem scenariju moral samostojno odkriti storitve, ki so potrebne za pridobitev odgovora na povpraševanje, pripraviti vhodne podatke storitev ter storitve nato tudi avtonomno prožiti.

Ugotovili smo, da semantične oznake storitev omogočajo avtomatizacijo odkrivanja, kompozicije ter proženja storitev. Razvili smo model ter implementirali

prototip, ki na osnovi semantičnih opisov storitev le-te uporablja kot podatkovne vire v času izvajanja povpraševanja na povsem transparenten način. Na osnovi razvitega modela, implementiranega in preizkušenega prototipa ter uspešne evalvacije na splošno sprejetem testnem okolju to hipotezo potrdimo.

H2

Sistemu za integracijo podatkovnih virov na osnovi ontologij, ki uporablja semantično označene storitve za pridobivanje podatkov, je možno ovrednotiti učinkovitost ter ga na ta način primerjati s konvencionalnimi pristopi.

S pregledom literature nismo uspeli odkriti pristopa, s katerim bi bilo mogoče v celoti in objektivno ovrednotiti ter primerjati različne pristope za integracijo podatkovnih virov na osnovi ontologij. V okviru doktorske disertacije smo tako na osnovi obstoječih znanstvenih rezultatov razvili dvonivojski model za evalvacijo pristopov, ki je sestavljen iz kvalitativnega ter kvantitativnega dela. Kvalitativni del opisuje karakteristike in zmožnosti sistema za integracijo, kvantitativni del, ki temelji na splošno sprejetem LUBM primerjalnem testu, pa meri učinkovitost izvajanja povpraševanj. Kvantitativna analiza meri čas izvajanja povpraševanj ter pravilnost rešitev na množici preddefiniranih povpraševanj. Razvit model za evalvacijo je možno uporabiti na poljubnem sistemu za integracijo informacijskih virov na osnovi ontologij. Na osnovi podanih dejstev potrdimo hipotezo.

H3

Vpeljava dodatnega nivoja semantično označenih storitev v večini primerov ohranja učinkovitost sistemov za integracijo podatkov na osnovi ontologij.

V okviru doktorske disertacije smo razvili prototip, ki implementira OSII model za integracijo podatkovnih virov na osnovi ontologij s semantično označenimi storitvami. Prototip predstavlja razširitev odprtokodne knjižnice Jena, ki omogoča delo s tehnologijami semantičnega spleta. Na osnovi prototipa ter splošno sprejetega primerjalnega testa LUBM smo izvedli eksperiment, kjer smo pod istimi pogoji merili hitrost ter pravilnost izvajanja povpraševanj.

Pri pregledu literature smo odkrili sorodne pristope za integracijo podatkovnih virov na osnovi ontologij. Nobeden od teh pristopov nima implementacije ali pa le-ta ni prosto dostopna. Zaradi tega ni bilo možno primerjati pristopa, ki smo ga razvili v okviru doktorske disertacije, s temi sorodnimi pristopi. Namesto tega smo OSII primerjali s temeljnimi tehnologijami za delo s semantičnimi tehnologijami. Nekateri od opisanih sorodnih pristopov uporabljajo za temeljne tehnologije rešitve, ki smo jih primerjali z našim pristopov. Ker smo izvedli primerjavo s temeljnimi tehnologijami za delo s semantičnimi tehnologijami, obstaja precejšnja verjetnost, da bodo potencialni novo-razviti pristopi prav tako uporabili rešitve, ki smo jih primerjali.

Na osnovi rezultatov eksperimenta smo izvedli statistično analizo. Rezultati statistične analize so pokazali, da med primerjanimi pristopi obstaja signifikantna razlika v učinkovitosti ($p < 0,05$). Nato smo izvedli post-hoc teste, ki so pokazali, da ne obstaja pristop, ki je signifikantno učinkovitejši od razvitega pristopa OSII ($p < 0,05$). OSII je bil v več testnih primerih celo najučinkovitejši izmed pristopov. Analizo smo izvedli za celotno množico povpraševanj ter za podmnožico povpraševanj, ki ne podpirajo sklepanja. Na celotni množici povpraševanj je bil OSII pristop, za optimističen kot tudi za pesimističen scenarij, najučinkovitejši izmed primerjanih pristopov. Tudi na množici povpraševanj, ki ne podpirajo sklepanja, se je OSII pristop izkazal kot izjemno učinkovit. Pristopi, ki po učinkovitosti signifikantno ne odstopajo od optimističnega scenarija razvitega prototipa, so v obeh testnih primerih zgolj trije najučinkovitejši ($p < 0,05$). Pesimistični scenarij OSII ima podobno učinkovitost kot optimistični scenarij, od katerega se signifikantno ne razlikuje.

Rezultati eksperimenta so pokazali, da je OSII v primerjavi z drugimi pristopi zelo skalabilen. Vsi obravnavanih sorodni pristopov razen D2R so pokazali sorazmerno slabo skalabilnost. Ti pristopi so občutljivi na povečanje velikosti podatkovne množice tudi za povpraševanja, katerim se rezultat s povečanjem podatkovne množice ne spremeni. Na osnovi rezultatov statistične analize, ki je bila opravljena na podlagi eksperimenta na splošno sprejetem primerjalnem testu LUBM, potrjujemo tudi to hipotezo.

Poglavje 6

Zaključek

V okviru doktorske disertacije smo razvili hibridni LAV pristop za integracijo podatkovnih virov na osnovi ontologij. Razviti pristop omogoča transparentno integracijo distribuiranih heterogenih podatkovnih virov na osnovi semantično označenih storitev ter sklepanje nad tako pridobljenimi podatki. Pristop vsebuje formalni model za LAV integracijo podatkovnih virov s semantično označenimi storitvami, arhitekturo sistema za integracijo, ontologijo za semantično označevanje storitev ter algoritme za distribuirano povpraševanje.

Enega izmed osrednjih znanstvenih prispevkov doktorske disertacije predstavljajo algoritmi, ki omogočajo distribuirano povpraševanje in sklepanje po podatkih, ki jih vračajo storitve. Razviti pristop za integracijo heterogenih podatkovnih virov OSII uporablja sparql povpraševalni jezik za izvajanje povpraševanj nad integriranimi podatki. Sparql algebro smo razširili tako, da omogoča samodejno proženje storitev neposredno iz sparql povpraševanj. Osrednjega pomena je algoritem za transformacijo povpraševanj, ki na osnovi preslikav med spletnimi storitvami in globalno ontologijo samodejno odkrije storitve, ki ponujajo delni odgovor na povpraševanje, naredi kompozicijo storitev ter pripravi vhodne podatke. Rezultat algoritma za transformacijo povpraševanj je načrt izvajanja storitev, ki je zapisan v razširjeni sparql algebri.

Pomembnejši znanstveni prispevki doktorske disertacije so še model za integracijo podatkovnih virov na osnovi ontologij, model za evalvacijo učinkovitosti takšnih sistemov, prototip ter ovrednotenje razvitega modela in podrobna analiza primerjave z obstoječimi pristopi. Doktorska disertacija med drugim predstavlja

rezultate sistematičnega pregleda literature pristopov za integracijo podatkovnih virov na osnovi ontologij ter rezultate pregleda literature semantično opisanih spletnih storitev. Rezultati pregleda literature so služili kot osnova za razvoj pristopa.

Razvili smo prototip, ki implementira OSII pristop za integracijo podatkovnih virov. Na osnovi prototipa smo izvedli eksperiment, ki primerja učinkovitost OSII pristopa z drugimi standardnimi rešitvami za dostop do podatkov na osnovi ontologije. Pri tem smo uporabili splošno sprejet primerjalni test LUBM, ki definira ontologijo, podatkovno množico ter množico povpraševanj. Rezultati eksperimenta so pokazali, da je procesiranje povpraševanj ter sklepanje na osnovi semantičnih opisov storitev izjemno učinkovito tako iz vidika hitrosti kot iz vidika popolnosti rezultatov. Sorazmerno hitro je tudi samo proženje storitev. Procesiranje povpraševanj je na testnem primeru povprečno trajalo manj kot 1 milisekundo, medtem ko je bil povprečni čas proženja ene storitve okrog 4 milisekunde. Ta čas vključuje transformacijo vhodnih in izhodnih podatkov, proženje storitve ter čakanje na odgovor storitve.

Na osnovi eksperimenta smo izvedli statistično analizo, ki je pokazala, da nobeden izmed primerjanih pristopov ni signifikantno učinkovitejši od pristopa, ki smo ga razvili. Ravno nasprotno, analiza je pokazala, da je razvit pristop signifikantno učinkovitejši od drugih pristopov, ki omogočajo sklepanje. Ta rezultat je presenetljiv, saj OSII pridobiva podatke dinamično v času izvajanja povpraševanj, medtem ko imajo ostali pristopi, ki omogočajo sklepanje vse podatke shranjene v pomnilniku. Tudi v kolikor primerjamo OSII s sistemi, ki ne podpirajo sklepanja, ugotovimo, da je najučinkovitejši izmed primerjanih pristopov. Signifikantno se ne razlikuje zgolj od najbolj učinkovitih pristopov.

Kot sklep doktorske disertacije lahko tako podamo ugotovitev, da je semantično označene storitve smotrno uporabiti kot podatkovne vire za integracijo na osnovi ontologij. Izkazalo se je celo, da je ta pristop v primerjavi z drugimi tehnologijami zelo učinkovit in izredno skalabilen. Vsekakor bi bilo smotrno takšen pristop uporabiti in ovrednotiti v industrijskem okolju.

Literatura

[1] RAMA AKKIRAJU. *Semantic Web Service - Theory, Tools, and Applications*, chapter Semantic Web Services, pages 191–216. Idea Group, 2006. (Citirano na straneh xix, 43 in 51.)

[2] FRANZ BAADER, DIEGO CALVANESE, DEBORAH L. MCGUINNESS, DANIELE NARDI, AND PETER F. PATEL-SCHNEIDER, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003. (Cititrano na strani 23.)

[3] ROBERT BATTLE AND EDWARD BENSON. Bridging the semantic web and web 2.0 with representational state transfer (rest). *Web Semant.*, **6**:61–69, February 2008. (Citirano na straneh 43 in 59.)

[4] FRANCINE BERMAN. Got data?: a guide to data preservation in the information age. *Commun. ACM*, **51**:50–56, December 2008. (Cititrano na strani 1.)

[5] TIM BERNERS-LEE, JAMES HENDLER, AND ORA LASSILA. The semantic web: Scientific american. *Scientific American*, May 2001. (Cititrano na strani 37.)

[6] PHILIP A. BERNSTEIN AND LAURA M. HAAS. Information integration in the enterprise. *Commun. ACM*, **51**:72–79, September 2008. (Citirano na straneh 9, 12 in 13.)

[7] C BIZER AND A SEABORNE. *D2RQ – Treating Non-RDF Databases as Virtual RDF Graphs*. Springer, 2004. (Cititrano na strani 116.)

[8] CHRISTIAN BIZER AND ANDREAS SCHULTZ. The berlin sparql benchmark. *Int. J. Semantic Web Inf. Syst.*, **5**[2]:1–24, 2009. (Cititrano na strani 101.)

[9] W N BORST. *Construction of engineering ontologies*. PhD thesis, University of Twente, 1997. (Cititrano na strani 27.)

[10] DAN BRICKLEY, R. V. GUHA, AND BRIAN MCBRIDE. Rdf vocabulary description language 1.0: Rdf schema. W3C Recommendation, 2004 February. (Cititrano na strani 31.)

[11] JOS DE BRUIJN, HOLGER LAUSEN, AXEL POLLERES, AND DIETER FENSEL. *The Semantic Web: Research and Applications*, chapter The Web Service Modeling Language WSML: An Overview, pages 590–604. Springer Berlin / Heidelberg, 2006. (Citirano na straneh xvii, 33, 34 in 44.)

[12] C. BUIL-ARANDA AND O. CORCHO. Federating queries to rdf repositories. Technical report, Universidad Politécnica de Madrid, http://oa.upm.es/3302/, 2010. (Cititrano na strani 59.)

[13] CARLOS BUIL-ARANDA, MARCELO ARENAS, AND OSCAR CORCHO. Semantics and optimization of the sparql 1.1 federation extension. In GRIGORIS ANTONIOU, MARKO GROBELNIK, ELENA SIMPERL, BIJAN PARSIA, DIMITRIS PLEXOUSAKIS, PIETER DE LEENHEER, AND JEFF PAN, editors, *The Semanic Web: Research and Applications*, **6644** of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin / Heidelberg, 2011. (Cititrano na strani 60.)

[14] DIEGO CALVANESE, GIUSEPPE DE GIACOMO, DOMENICO LEMBO, MAURIZIO LENZERINI, RICCARDO ROSATI, AND MARCO RUZZI. Using owl in data integration. In R. DE VIRGILIO, F. GIUNCHIGLIA, AND L. TANCA, editors, *Semantic Web Information Management - a Model Based Perspective*, pages 397–424. Springer, 2009. (Cititrano na strani 2.)

[15] DIEGO CALVANESE, GIUSEPPE GIACOMO, DOMENICO LEMBO, MAURIZIO LENZERINI, ANTONELLA POGGI, RICCARDO ROSATI, AND MARCO RUZZI. Data integration through dl-lite(a) ontologies. In KLAUS-DIETER SCHEWE AND BERNHARD THALHEIM, editors, *Semantics in Data and Knowledge*

Bases, chapter Data Integration through *DL-Lite*$_A$ Ontologies, pages 26–47. Springer-Verlag, Berlin, Heidelberg, 2008. (Cititrano na strani 53.)

[16] J. CARDOSO. The semantic web vision: Where are we? *Intelligent Systems, IEEE*, **22**[5]:84–88, Sept.-Oct. 2007. (Citirano na straneh xvii, 34, 35 in 36.)

[17] ERIKA CHRISTENSEN, FRANCISCO CURBERA, GREG MEREDITH, AND SANJIVA WEERAWARANA. Web services definition language (wsdl). W3C Note, March 2001. (Citirano na straneh 43 in 49.)

[18] ISABEL CRUZ AND HUIYONG XIAO. Ontology driven data integration in heterogeneous networks. In ANDREAS TOLK AND LAKHMI JAIN, editors, *Complex Systems in Knowledge-based Environments: Theory, Models and Applications*, **168** of *Studies in Computational Intelligence*, pages 75–98. Springer Berlin / Heidelberg, 2009. (Cititrano na strani 55.)

[19] RANDALL DAVIS, HOWARD SHROBE, AND PETER SZOLOVITS. What is a knowledge representation? *AI Magazine*, **14**[1]:17–33, 1993. (Citirano na straneh 16 in 17.)

[20] DIGITAL UNIVERSE DECADE AND ARE YOU READY. 2010 digital universe study. **2009**[Figure 1]:18, 2010. (Cititrano na strani 1.)

[21] JANEZ DEMŠAR. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, **7**:1–30, December 2006. (Citirano na straneh 148 in 152.)

[22] MARIN DIMITROV, ALEX SIMOV, VASSIL MOMTCHEV, AND MIHAIL KONSTANTINOV. Wsmo studio - a semantic web services modelling environment for wsmo. In *ESWC*, pages 749–758, 2007. (Cititrano na strani 46.)

[23] THOMAS ERL. *SOA Principles of Service*. Prentince Hall/PearsonPTR, 2005. (Cititrano na strani 4.)

[24] JOEL FARELL AND HOLGER LAUSEN. Semantic annotations for wsdl and xml schema. W3C Recommendation, August 2007. (Cititrano na strani 49.)

[25] ADAM FARQUHAR, RICHARD FIKES, AND JAMES RICE. The ontolingua server: a tool for collaborative ontology construction. In *International Journal of Human-Computer Studies*, 1996. (Cititrano na strani 30.)

[26] ACHILLE FOKOUE, AARON KERSHENBAUM, LI MA, EDITH SCHONBERG, AND KAVITHA SRINIVAS. The summary abox: Cutting ontologies down to size. *The Semantic Web - ISWC 2006*, pages 343–356, 2006. (Cititrano na strani 101.)

[27] JOHN GANTZ, DAVID REINSEL, CHRISTOPHER CHUTE, WOLFGANG SCHLICHTING, JOHN MCARTHUR, STEPHEN MINTON, IRIDA XHENETI, ANNA TONCHEVA, AND ALEX MANFREDIZ. Idc - the expanding digital universe: A forecast of worldwide information growth through 2010. 2007. (Cititrano na strani 1.)

[28] MICHAEL R. GENESERETH AND RICHARD E. FIKES. Knowledge interchange format. version 3.0. reference manual. Technical report, Stanford University, 1992. (Cititrano na strani 30.)

[29] ASUNCION GOMEZ-PEREZ, OSCAR CORCHO, AND MARIANO FERNANDEZ-LOPEZ. *Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition (Advanced Information and Knowledge Processing)*. Springer, July 2004. (Citirano na straneh xix, 17, 25, 26, 27 in 30.)

[30] BOSTJAN GRAŠIČ AND VILI PODGORELEC. Automating ontology based information integration using service orientation. *W. Trans. on Comp.*, **9**:547–556, June 2010. (Citirano na straneh 5 in 43.)

[31] BOŠTJAN GRAŠIČ AND VILI PODGORELEC. Towards automated execution of semantic web services in semantic networks. In *Zbornik 11. mednarodne multikonference Informacijska družba - IS 2008*, pages 219–222, 2008. (Cititrano na strani 5.)

[32] BOŠTJAN GRAŠIČ AND VILI PODGORELEC. On interoperability of ontology based knowledge systems. In *Proceedings of Fourth International Conference*

on Knowledge Management in Organizations: Knowledge Management and Service Science, 2009. (Cititrano na strani 6.)

[33] STEPHAN GRIMM, PASCAL HITZLER, AND ANDREAS ABECKER. *Semantic Web Services: Concepts, Technologies, and Applications*, chapter Knowledge Representation and Ontologies, pages 51–105. Springer-Verlag New York, Inc., 2007. (Citirano na straneh 16, 17, 20, 21 in 23.)

[34] BENJAMIN GROSOF. Semantic web services: Obstacles and attractions. In *Panel at 12th Intl. Conference on WWW*, 2003. (Cititrano na strani 43.)

[35] THOMAS R. GRUBER. A translation approach to portable ontology specifications. *Knowledge Acquisition*, **5**:199–220, 1993. (Cititrano na strani 27.)

[36] NICOLA GUARINO. Formal ontology and information systems. In *Formal Ontology in Information Systems (FOIS98)*, pages 3–15. IOS Press, 1998. (Citirano na straneh xvii, 28 in 29.)

[37] YUANBO GUO, ZHENGXIANG PAN, AND JEFF HEFLIN. Lubm: A benchmark for owl knowledge base systems. *J. Web Sem.*, **3**[2-3]:158–182, 2005. (Citirano na straneh 100, 101 in 106.)

[38] ALON HALEVY, ANAND RAJARAMAN, AND JOANN ORDILLE. Data integration: the teenage years. In *VLDB'2006: Proceedings of the 32nd international conference on Very large data bases*, pages 9–16. VLDB Endowment, 2006. (Citirano na straneh 9 in 13.)

[39] A. HALLER, E. CIMPIAN, A. MOCAN, E. OREN, AND C. BUSSLER. Wsmx - a semantic service-oriented architecture. In *International Conference on Web Service (ICWS 2005)*, 2005. (Citirano na straneh 44 in 45.)

[40] OLAF HARTIG, CHRISTIAN BIZER, AND JOHANN-CHRISTOPH FREYTAG. Executing sparql queries over the web of linked data. In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, pages 293–309, Berlin, Heidelberg, 2009. Springer-Verlag. (Cititrano na strani 61.)

[41] PASCAL HITZLER, MARKUS KRÖTZSCH, BIJAN PARSIA, PETER F. PATEL-SCHNEIDER, AND SEBASTIAN RUDOLPH, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at http://www.w3.org/TR/owl2-primer/. (Citirano na straneh 32, 68 in 90.)

[42] IAN HORROCKS, PETER F. PATEL-SCHNEIDER, HAROLD BOLEY, SAID TABET, BENJAMIN GROSOF, AND MIKE DEAN. Swrl: A semantic web rule language combining owl and ruleml. W3C Submission, May 2004. (Cititrano na strani 32.)

[43] ISKANDAR ISHAK AND NAOMIE SALIM. Database integration approaches for heterogeneous biological data sources: An owerview. In *Postgraduate Annual Research Seminar 2006 (PARS 2006)*. Postgraduate Studies Department FSKSM, UTM Skudai, 2006. (Citirano na straneh 9 in 10.)

[44] P. D. KARP, C. K. VINAY, AND J. THOMERE. Xol: An xml-based ontology exchange language. Pangaea Systems and SRI, International, 1999. (Cititrano na strani 30.)

[45] MICK KERRIGAN, ADRIAN MOCAN, MARTIN TANLER, AND DIETER FENSEL. The web service modeling toolkit - an integrated development environment for semantic web services. In *ESWC*, pages 789–798, 2007. (Cititrano na strani 46.)

[46] MATTHIAS KLUSCH AND ZHIGUO XING. Semantic web services in the web: A preliminary reality check. In TOMMASO DI NOIA, RUBÉN LARA, AXEL POLLERES, IOAN TOMA, TAKAHIRO KAWAMURA, MATTHIAS KLUSCH, ABRAHAM BERNSTEIN, MASSIMO PAOLUCCI, ALAIN LEGER, AND DAVID L. MARTIN, editors, *SMRR*, **243** of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007. (Citirano na straneh xix in 51.)

[47] J KOPECKY, T VITVAR, C BOURNEZ, AND J FARRELL. Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, **11**[6]:60–67, 2007. (Citirano na straneh xvii in 50.)

[48] H. Y. Lam, L. Marenco, G. M. Shepherd, P. L. Miller, and K. H. Cheung. Using web ontology language to integrate heterogeneous databases in the neurosciences. *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium*, pages 464–468, 2006. (Citirano na straneh 56 in 116.)

[49] Andreas Langegger. *A Flexible Architecture for Virtual Information Integration Based on Semantic Web Concepts*. PhD thesis, Johannes Kepler University Linz, Austria, 2010. (Citirano na straneh 13 in 58.)

[50] Andreas Langegger, Wolfram Wöss, and Martin Blöchl. A semantic web middleware for virtual data integration on the web. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, *ESWC*, **5021** of *Lecture Notes in Computer Science*, pages 493–507. Springer, 2008. (Cititrano na strani 58.)

[51] O. Lassila and Deborah L. McGuinness. The role of frame-based representation on the semantic web. Technical Report KSL-01-02, Stanford University, Stanford, 2001. (Citirano na straneh xvii, 28 in 29.)

[52] Holger Lausen, Rubén Lara, Axel Polleres, Jos de Bruijn, and Dumitru Roman. *Semantic Web Services: Concepts, Technologies, and Applications*, chapter Semantic Annotation for Web Services, pages 179–209. Springer Verlag, 2007. (Citirano na straneh 44 in 46.)

[53] Douglas B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989. (Cititrano na strani 30.)

[54] Maurizio Lenzerini. Data integration: a theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '02, pages 233–246, New York, NY, USA, 2002. ACM. (Citirano na straneh 9, 12, 13, 14, 16, 68 in 69.)

[55] ALON Y. LEVY, ANAND RAJARAMAN, AND JOANN J. ORDILLE. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22th International Conference on Very Large Data Bases*, VLDB '96, pages 251–262, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc. (Citirano na straneh 1, 4 in 52.)

[56] SEAN LUKE AND JAMES HEFLIN. SHOE 1.0 proposed specification. (Cititrano na strani 30.)

[57] LI MA, YANG YANG, ZHAOMING QIU, GUOTONG XIE, YUE PAN, AND SHENGPING LIU. Towards a complete owl ontology benchmark. In YORK SURE AND JOHN DOMINGUE, editors, *The Semantic Web: Research and Applications*, **4011** of *Lecture Notes in Computer Science*, pages 125–139. Springer Berlin / Heidelberg, 2006. (Cititrano na strani 100.)

[58] ROBERT M. MACGREGOR. Inside the loom description classifier. *SIGART Bull.*, **2**[3]:88–92, 1991. (Cititrano na strani 30.)

[59] FRANK MANOLA AND ERIC MILLER. Rdf primer. W3C Recommendation, 2004. (Cititrano na strani 30.)

[60] DAVID MARTIN, MARK BURSTEIN, DREW MCDERMOTT, SHEILA MCILRAITH, MASSIOMO PAOLUCCI, KATIA SYCARA, DEBORAH L. MCGUINESS, EVREN SIRIN, AND NAVEEN SRINIVASAN. Bringing semantics to web services with owl-s. *World Wide Web*, **10**:243–277, 2007. (Citirano na straneh xvii, 46, 47 in 48.)

[61] DEBORAH L. MCGUINESS. Owl web ontology language. W3C Recommendation, February 2004. (Cititrano na strani 31.)

[62] BORIS MOTIK AND ULRIKE SATTLER. A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes. In MIKI HERMANN AND ANDREI VORONKOV, editors, *Proc. of the 13th Int. Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2006)*, **4246** of *LNCS*, pages 227–241, Phnom Penh, Cambodia, November 13–17 2006. Springer. (Cititrano na strani 101.)

[63] E. MOTTA. *Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 1999. (Cititrano na strani 30.)

[64] GILLES NACHOUKI AND MOHAMED QUAFAFOU. Mashup web data sources and services based on semantic queries. *Information Systems*, **36**[2]:151 – 173, 2011. Special Issue: Semantic Integration of Data, Multimedia, and Services. (Cititrano na strani 54.)

[65] CHRIS PREIST. *Semantic Web Services: Concepts, Technologies, and Applications*, chapter Combining Web Services with Semantic Web Technology, pages 159–178. Springer Verlag, 2007. (Cititrano na strani 43.)

[66] ERIC PRUD'HOMMEAUX AND ANDY SEABORNE. Sparql query language for rdf. W3C Recommendation, January 2008. (Citirano na straneh 37, 40 in 74.)

[67] BASTIAN QUILITZ AND ULF LESER. Querying distributed rdf data sources with sparql. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, ESWC'08, pages 524–538, Berlin, Heidelberg, 2008. Springer-Verlag. (Cititrano na strani 56.)

[68] DUMITRU ROMAN, UWE KELLER, HOLGER LAUSEN, JOS DE BRUIJN, RUBÉN LARA, MICHAEL STOLLBERG, AXEL POLLERES, CRISTINA FEIER, CHRISTOPH BUSSLER, AND DIETER FENSEL. Web service modeling ontology. *Applied Ontology*, **1**[1]:77–106, 2005. (Cititrano na strani 44.)

[69] DUMITRU ROMAN, HOLGER LAUSEN, AND UWE KELLER. Wsmo final draft. WSMO Working Draft, http://www.wsmo.org/TR/d2/v1.3/, October 2006. (Cititrano na strani 44.)

[70] STUART RUSSELL AND PETER NORVIG. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2003. (Cititrano na strani 20.)

[71] SATYA S. SAHOO, OLIVIER BODENREIDER, JONI L. RUTTER, KAREN J. SKINNER, AND AMIT P. SHETH. An ontology-driven semantic mashup of gene and biological pathway information: Application to the domain of nicotine

dependence. *Journal of Biomedical Informatics*, **41**[5]:752 – 765, 2008. Semantic Mashup of Biomedical Data. (Cititrano na strani 54.)

[72] SIMON SCHENK AND STEFFEN STAAB. Networked graphs: a declarative mechanism for sparql rules, sparql views and rdf data integration on the web. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 585–594, New York, NY, USA, 2008. ACM. (Cititrano na strani 57.)

[73] MICHAEL SCHMIDT, THOMAS HORNUNG, NORBERT KÜCHLIN, GEORG LAUSEN, AND CHRISTOPH PINKEL. An experimental comparison of rdf data management approaches in a sparql benchmark scenario. In *Proceedings of the 7th International Conference on The Semantic Web*, ISWC '08, pages 82–97, Berlin, Heidelberg, 2008. Springer-Verlag. (Cititrano na strani 101.)

[74] JIA-LANG SENG AND I.L. KONG. A schema and ontology-aided intelligent information integration. *Expert Systems with Applications*, **36**[7]:10538 – 10550, 2009. (Citirano na straneh 55 in 96.)

[75] M. OMAIR SHAFIQ, MATTHEW MORAN, EMILIA CIMPIAN, ADRIAN MOCAN, MICHAL ZAREMBA, AND DIETER FENSEL. Investigating semantic web service execution environments: A comparison between wsmx and owl-s tools. In *ICIW*, page 31. IEEE Computer Society, 2007. (Citirano na straneh xix, 49 in 51.)

[76] EVREN SIRIN, BIJAN PARSIA, BERNARDO CUENCA GRAU, ADITYA KALYANPUR, AND YARDEN KATZ. Pellet: A practical owl-dl reasoner. *Web Semant.*, **5**:51–53, June 2007. (Citirano na straneh 101 in 115.)

[77] RUDI STUDER, RICHARD BENJAMINS, AND DIETER FENSEL. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, **25**[1-2]:161–198, MAR 1998. (Citirano na straneh 3 in 27.)

[78] STIJN VERSTICHEL, FEMKE ONGENAE, LEANNEKE LOEVE, FREDERIK VERMEULEN, PIETER DINGS, BART DHOEDT, TOM DHAENE, AND FILIP DE TURCK.

Efficient data integration in the railway domain through an ontology-based methodology. *Transportation Research Part C: Emerging Technologies*, **19**[4]:617 – 643, 2011. (Citirano na straneh 55 in 116.)

[79] H. WACHE, T. VÖGELE, U. VISSER, H. STUCKENSCHMIDT, G. SCHUSTER, H. NEUMANN, AND S. HÜBNER. Ontology-based integration of information - a survey of existing approaches. pages 108–117, 2001. (Cititrano na strani 2.)

[80] TIMO WEITHÖNER, THORSTEN LIEBIG, MARKO LUTHER, AND SEBASTIAN BÖHM. What's Wrong with OWL Benchmarks? In *Proc. of the Second Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006)*, pages 101–114, Athens, GA, USA, November 2006. (Cititrano na strani 101.)

[81] JINGTAO YAO, VIJAY V. RAGHAVAN, AND ZONGHUAN WU. Web information fusion: A review of the state of the art. *Inf. Fusion*, **9**:446–449, October 2008. (Cititrano na strani 9.)

Priloga A: OWL ontologija za opis storitev

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:calc="http://ii.uni-mb.si/osii/calculator.owl#"
    xmlns:lubm="http://ii.uni-mb.si/osii/lubm_1.owl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns="http://ii.uni-mb.si/osii/OSIIServices.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://ii.uni-mb.si/osii/OSIIServices.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Mediator"/>
  <owl:Class rdf:ID="Service_REST">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Service"/>
```

```
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Service_Main_Concept"/>
<owl:Class rdf:ID="Lifting_Mediator">
  <rdfs:subClassOf rdf:resource="#Mediator"/>
</owl:Class>
<owl:Class rdf:ID="Lowering_Mediator">
  <rdfs:subClassOf rdf:resource="#Mediator"/>
</owl:Class>
<owl:Class rdf:ID="Service_SOAP">
  <rdfs:subClassOf rdf:resource="#Service"/>
</owl:Class>
<owl:Class rdf:ID="Service_Output"/>
<owl:Class rdf:ID="Mapping"/>
<owl:Class rdf:ID="Service_Input"/>
<owl:Class rdf:ID="XMLMapping">
  <rdfs:subClassOf rdf:resource="#Mapping"/>
</owl:Class>
<owl:Class rdf:ID="Transformation_Mediator">
  <rdfs:subClassOf rdf:resource="#Mediator"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasOutput">
  <rdfs:domain rdf:resource="#Service"/>
  <rdfs:range rdf:resource="#Service_Output"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isMediatorFor">
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Service_Output"/>
```

```
          <owl:Class rdf:about="#Service_Input"/>
          <owl:Class rdf:about="#Service"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="hasMediator"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="#Mediator"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasMainConcept">
    <rdfs:domain rdf:resource="#Service"/>
    <rdfs:range rdf:resource="#Service_Main_Concept"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#hasMediator">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Service_Output"/>
          <owl:Class rdf:about="#Service_Input"/>
          <owl:Class rdf:about="#Service"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="#Mediator"/>
    <owl:inverseOf rdf:resource="#isMediatorFor"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasInputMediator">
    <rdfs:subPropertyOf rdf:resource="#hasMediator"/>
  </owl:ObjectProperty>
```

```
  <owl:ObjectProperty rdf:ID="hasOutputMediator">
    <rdfs:subPropertyOf rdf:resource="#hasMediator"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasURI"/>
  <owl:ObjectProperty rdf:ID="forwardTransformation">
    <rdfs:range rdf:resource="#Mediator"/>
    <rdfs:domain rdf:resource="#Transformation_Mediator"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasMapping">
    <rdfs:range rdf:resource="#Mapping"/>
    <rdfs:domain rdf:resource="#Mediator"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasInput">
    <rdfs:range rdf:resource="#Service_Input"/>
    <rdfs:domain rdf:resource="#Service"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="hasInputSchema">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSch
ema#string"/>
    <rdfs:domain rdf:resource="#Service"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="hasRestriction">
    <rdfs:domain rdf:resource="#Service"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSch
ema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="soap_operation">
    <rdfs:domain rdf:resource="#Service_SOAP"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="mapping_xPath">
```

```
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSch
ema#string"/>
    <rdfs:domain rdf:resource="#XMLMapping"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="soap_nameSpace">
    <rdfs:domain rdf:resource="#Service_SOAP"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="soap_address">
    <rdfs:domain rdf:resource="#Service_SOAP"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="hasCode">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSch
ema#string"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Lifting_Mediator"/>
          <owl:Class rdf:about="#Lowering_Mediator"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="mapping_xml_variable">
    <rdfs:domain rdf:resource="#XMLMapping"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSch
ema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="soap_portType">
    <rdfs:domain rdf:resource="#Service_SOAP"/>
  </owl:DatatypeProperty>
```

```
  <owl:DatatypeProperty rdf:ID="hasOutputSchema">
    <rdfs:domain rdf:resource="#Service"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSch
ema#string"/>
  </owl:DatatypeProperty>
</rdf:RDF>
```

Priloga B: Primer opisa storitve

```
<Service_SOAP rdf:ID="Faculty_teacherOf">
   <soap_address rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >http://localhost:8080/LUBM-war/FacultyService</soap_address>
   <soap_nameSpace rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >http://lubm.cases.osii.ii.unimb.si/</soap_nameSpace>
   <soap_portType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >FacultyPort</soap_portType>
   <soap_operation rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >teacherOf</soap_operation>

   <hasMainConcept rdf:resource="http://ii.uni-mb.si/osii/lubm_1.owl#Course"/>

   <hasInputMediator rdf:resource="#Faculty_teacherOf_IMediator"/>
   <hasOutputMediator rdf:resource="#Faculty_teacherOf_OMediator"/>

   <hasInput>
     <Service_Input rdf:about="http://ii.uni-mb.si/osii/lubm_1.owl#Faculty">
       <hasMediator>
         <Lowering_Mediator rdf:ID="Faculty_teacherOf_IMediator">
           <hasCode rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
           >PREFIX lubm: &lt;http://ii.uni-mb.si/osii/lubm_1.owl#&gt; SELECT ?X
{?mainConcept &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type&gt; lubm:Faculty}
           </hasCode>
           <hasMapping>
             <XMLMapping rdf:ID="Faculty_teacherOf_IMediator_Mapping">
               <mapping_xml_variable rdf:datatype="http://www.w3.org/2001/XMLSchema
#string">
```

```
                    ?http://ii.uni-mb.si/osii/OSIIServices.owl#Faculty_teacherOf$Faculty
                  </mapping_xml_variable>
                  <mapping_xPath rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                  >/teacherOf/arg0</mapping_xPath>
                </XMLMapping>
              </hasMapping>
            </Lowering_Mediator>
          </hasMediator>
        </Service_Input>
      </hasInput>

      <hasOutput>
        <Service_Output rdf:about="http://ii.uni-mb.si/osii/lubm_1.owl#Course">
          <hasMediator>
            <Lifting_Mediator rdf:ID="Faculty_teacherOf_OMediator">
              <hasMapping>
                <XMLMapping rdf:ID="Faculty_teacherOf_OMediator_Mapping">
                  <mapping_xPath rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                  >//ns:teacherOfResponse/return</mapping_xPath>
                  <mapping_xml_variable rdf:datatype="http://www.w3.org/2001/XMLSch
ema#string">?http://ii.uni-mb.si/osii/OSIIServices.owl#Faculty_teacherOf$Course
                  </mapping_xml_variable>
                </XMLMapping>
              </hasMapping>
              <hasCode rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
PREFIX lubm: &lt;http://ii.uni-mb.si/osii/lubm_1.owl#&gt; SELECT ?X {?mainConcept
&lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#type&gt; lubm:Course}</hasCode>
            </Lifting_Mediator>
          </hasMediator>
        </Service_Output>
      </hasOutput>
    </Service_SOAP>
```

Priloga C: Skripta za kreiranje LUBM relacijske baze

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';

-- -----------------------------------------------------
-- Table `AssistantProfessor`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `AssistantProfessor` ;
CREATE  TABLE IF NOT EXISTS `AssistantProfessor` (
  `idAssistantProfessor` INT(11) NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(60) NULL DEFAULT NULL ,
  `worksFor` VARCHAR(60) NULL DEFAULT NULL ,
  `emailAddress` VARCHAR(60) NULL DEFAULT NULL ,
  `telephone` VARCHAR(60) NULL DEFAULT NULL ,
  `researchInterest` VARCHAR(60) NULL DEFAULT NULL ,
  `uri` VARCHAR(60) NULL DEFAULT NULL ,
  `undergraduateDegreeFrom` VARCHAR(60) NULL DEFAULT NULL ,
  `mastersDegreeFrom` VARCHAR(60) NULL DEFAULT NULL ,
  `doctoralDegreeFrom` VARCHAR(60) NULL DEFAULT NULL ,
  PRIMARY KEY (`idAssistantProfessor`) ,
  INDEX `URI` (`uri` ASC) ,
  INDEX `worksFor` (`worksFor` ASC) )
ENGINE = MyISAM
AUTO_INCREMENT = 2483
```

```
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `AssociateProfessor`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `AssociateProfessor` ;
CREATE  TABLE IF NOT EXISTS `AssociateProfessor` (
  `idAssociateProfessor` INT(11) NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(60) NULL DEFAULT NULL ,
  `worksFor` VARCHAR(60) NULL DEFAULT NULL ,
  `emailAddress` VARCHAR(60) NULL DEFAULT NULL ,
  `telephone` VARCHAR(60) NULL DEFAULT NULL ,
  `researchInterest` VARCHAR(60) NULL DEFAULT NULL ,
  `uri` VARCHAR(60) NULL DEFAULT NULL ,
  `undergraduateDegreeFrom` VARCHAR(60) NULL DEFAULT NULL ,
  `mastersDegreeFrom` VARCHAR(60) NULL DEFAULT NULL ,
  `doctoralDegreeFrom` VARCHAR(60) NULL DEFAULT NULL ,
  PRIMARY KEY (`idAssociateProfessor`) ,
  INDEX `URI` (`uri` ASC) ,
  INDEX `worksFor` (`worksFor` ASC) )
ENGINE = MyISAM
AUTO_INCREMENT = 2817
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `Course`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Course` ;
CREATE  TABLE IF NOT EXISTS `Course` (
  `CourseID` INT(11) NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(60) NULL DEFAULT NULL ,
  `uri` VARCHAR(60) NULL DEFAULT NULL ,
  PRIMARY KEY (`CourseID`) ,
  INDEX `uri` (`uri` ASC) )
ENGINE = MyISAM
DEFAULT CHARACTER SET = utf8;
```

Priloga C: Skripta za kreiranje LUBM relacijske baze

```
-- -----------------------------------------------------
-- Table `FullProfessor`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `FullProfessor` ;
CREATE  TABLE IF NOT EXISTS `FullProfessor` (
  `idFullProfessor` INT(11) NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(60) NULL DEFAULT NULL ,
  `worksFor` VARCHAR(60) NULL DEFAULT NULL ,
  `emailAddress` VARCHAR(60) NULL DEFAULT NULL ,
  `telephone` VARCHAR(60) NULL DEFAULT NULL ,
  `researchInterest` VARCHAR(60) NULL DEFAULT NULL ,
  `uri` VARCHAR(60) NULL DEFAULT NULL ,
  `undergraduateDegreeFrom` VARCHAR(60) NULL DEFAULT NULL ,
  `mastersDegreeFrom` VARCHAR(60) NULL DEFAULT NULL ,
  `doctoralDegreeFrom` VARCHAR(60) NULL DEFAULT NULL ,
  PRIMARY KEY (`idFullProfessor`) ,
  INDEX `URI` (`uri` ASC) ,
  INDEX `worksFor` (`worksFor` ASC) )
ENGINE = MyISAM
AUTO_INCREMENT = 2001
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `GraduateCourse`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `GraduateCourse` ;
CREATE  TABLE IF NOT EXISTS `GraduateCourse` (
  `GraduateCourseID` INT(11) NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(60) NULL DEFAULT NULL ,
  `uri` VARCHAR(60) NULL DEFAULT NULL ,
  PRIMARY KEY (`GraduateCourseID`) ,
  INDEX `uri` (`uri` ASC) )
ENGINE = MyISAM
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `GraduateStudent`
```

```
-- -----------------------------------------------------
DROP TABLE IF EXISTS `GraduateStudent` ;
CREATE  TABLE IF NOT EXISTS `GraduateStudent` (
  `GraduateStudentID` INT(11) NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(60) NULL DEFAULT NULL ,
  `memberOf` VARCHAR(60) NULL DEFAULT NULL ,
  `emailAddress` VARCHAR(60) NULL DEFAULT NULL ,
  `telephone` VARCHAR(60) NULL DEFAULT NULL ,
  `uri` VARCHAR(60) NULL DEFAULT NULL ,
  `advisor` VARCHAR(60) NULL DEFAULT NULL ,
  `undergraduateDegreeFrom` VARCHAR(60) NULL DEFAULT NULL ,
  PRIMARY KEY (`GraduateStudentID`) ,
  INDEX `URI` (`uri` ASC) ,
  INDEX `memberOf` (`memberOf` ASC) )
ENGINE = MyISAM
AUTO_INCREMENT = 24363
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `GraduateStudentTakesCourse`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `GraduateStudentTakesCourse` ;
CREATE  TABLE IF NOT EXISTS `GraduateStudentTakesCourse` (
  `idgradSTakesCourse` INT(11) NOT NULL AUTO_INCREMENT ,
  `student` VARCHAR(80) NULL DEFAULT NULL ,
  `course` VARCHAR(60) NULL DEFAULT NULL ,
  PRIMARY KEY (`idgradSTakesCourse`) ,
  INDEX `student` (`student` ASC) ,
  INDEX `course` (`course` ASC) )
ENGINE = MyISAM
AUTO_INCREMENT = 3739
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `Lecturer`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Lecturer` ;
```

Priloga C: Skripta za kreiranje LUBM relacijske baze

```
CREATE  TABLE IF NOT EXISTS `Lecturer` (
  `idLecturer` INT(11) NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(60) NULL DEFAULT NULL ,
  `worksFor` VARCHAR(60) NULL DEFAULT NULL ,
  `emailAddress` VARCHAR(60) NULL DEFAULT NULL ,
  `telephone` VARCHAR(60) NULL DEFAULT NULL ,
  `researchInterest` VARCHAR(60) NULL DEFAULT NULL ,
  `uri` VARCHAR(60) NULL DEFAULT NULL ,
  `undergraduateDegreeFrom` VARCHAR(60) NULL DEFAULT NULL ,
  `mastersDegreeFrom` VARCHAR(60) NULL DEFAULT NULL ,
  `doctoralDegreeFrom` VARCHAR(60) NULL DEFAULT NULL ,
  PRIMARY KEY (`idLecturer`) ,
  INDEX `URI` (`uri` ASC) ,
  INDEX `worksFor` (`worksFor` ASC) )
ENGINE = MyISAM
AUTO_INCREMENT = 1396
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `Publication`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Publication` ;
CREATE  TABLE IF NOT EXISTS `Publication` (
  `PublicationID` INT(11) NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(60) NULL DEFAULT NULL ,
  `uri` VARCHAR(80) NULL DEFAULT NULL ,
  PRIMARY KEY (`PublicationID`) ,
  INDEX `uri` (`uri` ASC) )
ENGINE = MyISAM
AUTO_INCREMENT = 29996
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `PublicationAuthor`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `PublicationAuthor` ;
CREATE  TABLE IF NOT EXISTS `PublicationAuthor` (
```

```
  `PublicationAuthorID` INT(11) NOT NULL AUTO_INCREMENT ,
  `publication` VARCHAR(80) NULL DEFAULT NULL ,
  `author` VARCHAR(80) NULL DEFAULT NULL ,
  PRIMARY KEY (`PublicationAuthorID`) ,
  INDEX `publication` (`publication` ASC) ,
  INDEX `author` (`author` ASC) )
ENGINE = MyISAM
AUTO_INCREMENT = 53171
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `ResearchAssistant`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ResearchAssistant` ;
CREATE  TABLE IF NOT EXISTS `ResearchAssistant` (
  `ResearchAssistantID` INT(11) NOT NULL AUTO_INCREMENT ,
  `uri` VARCHAR(60) NULL DEFAULT NULL ,
  PRIMARY KEY (`ResearchAssistantID`) ,
  INDEX `uri` (`uri` ASC) )
ENGINE = MyISAM
AUTO_INCREMENT = 8206
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `TeachingAssistant`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `TeachingAssistant` ;
CREATE  TABLE IF NOT EXISTS `TeachingAssistant` (
  `TeachingAssistantID` INT(11) NOT NULL AUTO_INCREMENT ,
  `uri` VARCHAR(60) NULL DEFAULT NULL ,
  `teachingAssistantOf` VARCHAR(60) NULL DEFAULT NULL ,
  PRIMARY KEY (`TeachingAssistantID`) ,
  INDEX `uri` (`uri` ASC) )
ENGINE = MyISAM
AUTO_INCREMENT = 6106
DEFAULT CHARACTER SET = utf8;
```

Priloga C: Skripta za kreiranje LUBM relacijske baze

```
-- -----------------------------------------------------
-- Table `UndergraduateStudent`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `UndergraduateStudent` ;
CREATE  TABLE IF NOT EXISTS `UndergraduateStudent` (
  `UndergraduateStudentID` INT(11) NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(60) NULL DEFAULT NULL ,
  `memberOf` VARCHAR(60) NULL DEFAULT NULL ,
  `emailAddress` VARCHAR(70) NULL DEFAULT NULL ,
  `telephone` VARCHAR(60) NULL DEFAULT NULL ,
  `uri` VARCHAR(80) NULL DEFAULT NULL ,
  `advisor` VARCHAR(60) NULL DEFAULT NULL ,
  PRIMARY KEY (`UndergraduateStudentID`) ,
  INDEX `URI` (`uri` ASC) ,
  INDEX `memberOf` (`memberOf` ASC) )
ENGINE = MyISAM
AUTO_INCREMENT = 47529
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `UndergraduateStudentTakesCourse`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `UndergraduateStudentTakesCourse` ;
CREATE  TABLE IF NOT EXISTS `UndergraduateStudentTakesCourse` (
  `idgradSTakesCourse` INT(11) NOT NULL AUTO_INCREMENT ,
  `student` VARCHAR(80) NULL DEFAULT NULL ,
  `course` VARCHAR(60) NULL DEFAULT NULL ,
  PRIMARY KEY (`idgradSTakesCourse`) ,
  INDEX `student` (`student` ASC) ,
  INDEX `course` (`course` ASC) )
ENGINE = MyISAM
AUTO_INCREMENT = 17752
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `takesCourse`
-- -----------------------------------------------------
```

```
DROP TABLE IF EXISTS `takesCourse` ;
CREATE  TABLE IF NOT EXISTS `takesCourse` (
  `student` VARCHAR(80) NULL DEFAULT NULL ,
  `course` VARCHAR(60) NULL DEFAULT NULL ,
  `takesCourseID` INT(11) NOT NULL AUTO_INCREMENT ,
  PRIMARY KEY (`takesCourseID`) ,
  INDEX `course` (`course` ASC) ,
  INDEX `student` (`student` ASC) )
ENGINE = MyISAM
AUTO_INCREMENT = 68210
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `teacherOf`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `teacherOf` ;
CREATE  TABLE IF NOT EXISTS `teacherOf` (
  `teacherOfID` INT(11) NOT NULL AUTO_INCREMENT ,
  `teacher` VARCHAR(60) NULL DEFAULT NULL ,
  `course` VARCHAR(60) NULL DEFAULT NULL ,
  PRIMARY KEY (`teacherOfID`) ,
  INDEX `teacher` (`teacher` ASC) ,
  INDEX `course` (`course` ASC) )
ENGINE = MyISAM
AUTO_INCREMENT = 4882
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Table `University`
-- -----------------------------------------------------

DROP TABLE IF EXISTS `University` ;
CREATE TABLE IF NOT EXISTS `University` (
  `idUniversity` int(11) NOT NULL AUTO_INCREMENT,
  `uri` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`idUniversity`) ,
  INDEX `uri` (`uri` ASC)
```

```
) ENGINE=MyISAM DEFAULT CHARSET=latin1$$


-- -----------------------------------------------------
-- Table `Department`
-- -----------------------------------------------------

DROP TABLE IF EXISTS `Department` ;
CREATE  TABLE IF NOT EXISTS `Department` (
  `idDepartment` INT NOT NULL AUTO_INCREMENT ,
  `uri` VARCHAR(45) NULL ,
  `subOrganizationOf` VARCHAR(45) NULL ,
  PRIMARY KEY (`idDepartment`) ,
  INDEX `uri` (`uri` ASC) ,
  INDEX `subOrganizationOf` (`uri` ASC) )
ENGINE = MyISAM$$

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Življenjepis

Kratek življenjepis

18.12.1981	Rojen v Mariboru
1988-1996	Osnovna šola Ivana Cankarja v Mariboru
1996-2000	III. gimnazija Maribor, pridobljen naziv: gimnazijski maturant
2000-2006	Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru, pridobljen naziv: univ. dipl. ing. rač. in inf.
2002-2003	Brandenburgische Technische Universität, Nemčija. Izmenjava v okviru programa Socrates-Erasmus (2 semestra)
2006	Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru, vpis na podiplomski študij računalništva in informatike
2006	Fakulteta za ekonomsko poslovne vede, Univerza v Mariboru, vpis na magistrski študij poslovodnje in organizacija (MBA)
2008	Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru, prehod na enovit doktorski študij

Povzetek strokovne biografije

2005-2008	Delo v podjetju Jamada d.o.o., naloge: razvoj in podpora ERP sistemov za srednja in manjša podjetja, vodenje projektov manjšega obsega, vpeljava novih zaposlenih, delovno mesto: višji programer in podpornik
2008	Zaposlitev na Univerzi v Mariboru na delovno mesto mladi raziskovalec
2008-2010	Asistent pri predmetu Ekspertni sistemi univerzitetnega študijskega programa Računalništvo in informatika, FERI, UM
2010	Asistent pri predmetu Portalne tehnologije in upravljanje z znanjem študijskega programa Informatika in tehnologije komuniciranje, FERI, UM

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA O OBJAVI ELEKTRONSKE VERZIJE DOKTORSKE DISERTACIJE IN OSEBNIH PODATKOV, VEZANIH NA ZAKLJUČEK ŠTUDIJA

Ime in priimek doktoranda: **Boštjan Grašič**
Vpisna številka: **95029572**
Študijski program: **računalništvo in informatika**
Naslov doktorskega dela: **Integracija podatkovnih virov na osnovi ontologij s semantično označenimi storitvami**

Mentor: **izr. prof. dr. Vili Podgorelec**

Podpisani soglašam z objavo doktorske disertacije v Digitalni knjižnici Univerze v Mariboru.

Tiskana verzija doktorske disertacije je istovetna elektronski verziji, ki sem jo oddal v Digitalno knjižnico Univerze v Mariboru.

Podpisani hkrati izjavljam, da dovoljujem objavo osebnih podatkov, vezanih na zaključek študija (ime, priimek, leto in kraj rojstva, datum diplomiranja, naslov diplomskega dela) na spletnih straneh in v publikacijah Univerze v Mariboru.

Datum in kraj:

Maribor, 27.09.2011

Podpis doktoranda:

Boštjan Grašič

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA DOKTORSKEGA KANDIDATA

Podpisani **Boštjan Grašič**,

vpisna številka **95029572**

i z j a v l j a m,

da je doktorska disertacija z naslovom **Integracija podatkovnih virov na osnovi ontologij s semantično označenimi storitvami**

- rezultat lastnega raziskovalnega dela,
- da predložena disertacija v celoti ali v delih ni bila predložena za pridobitev kakršnekoli izobrazbe po študijskem programu druge fakultete ali univerze,
- da so rezultati korektno navedeni in
- da nisem kršil avtorskih pravic in intelektualne lastnine drugih.

Podpis doktorskega kandidata:

Boštjan Grašič

Univerza v Mariboru
Fakulteta za elektrotehniko,
računalništvo in informatiko

IZJAVA KANDIDATOVEGA MENTORJA O USTREZNOSTI DOKTORSKE DISERTACIJE

Podpisani **izr. prof. dr. Vili Podgorelec**, mentor doktorskemu kandidatu, izjavljam, da je doktorska disertacija z naslovom

Integracija podatkovnih virov na osnovi ontologij s semantično označenimi storitvami

ki jo je izdelal doktorski kandidat **Boštjan Grašič**, v skladu z odobreno temo, Pravilnikom o pripravi in zagovoru doktorske disertacije ter mojimi navodili in predstavlja izviren prispevek k razvoju znanstvene discipline.

Datum in kraj:

Maribor, 28.09.2011

Podpis mentorja: