UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA

APLIKACIJE POVEZANIH PODATKOV

MARKO VUKELIĆ

2014
ZAKLJUČNA NALOGA
VUKELIĆ

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Aplikacije povezanih podatkov

(Applications of linked data)

Ime in priimek: Marko Vukelić
Študijski program: Računalništvo in informatika
Mentor: doc. dr. Iztok Savnik

Koper, junij 2014

Ključna dokumentacijska informacija

Ime in PRIIMEK: Marko VUKELIĆ

Naslov zaključne naloge: Aplikacije povezanih podatkov

Kraj: Koper Število slik: 31 Število tabel: 4

Leto: 2014

Število listov: 62

Število referenc: 20

Mentor: doc. dr. Iztok Savnik

Ključne besede: Povezani podatki, semantični splet, RDF model, trojice, LOD

Izvleček:

Svetovni splet je omogočil globalni informacijski prostor povezanih dokumentov. Računalnik ima pri tem samo nalogo posrednika, ki nam postreže dokumente v čitljivi obliki. Izziv, ki se je pri tem pojavil je ustvariti splet v katerem računalnik ni samo posrednik dokumentov, ampak lahko dostopa do podatkov in jih razume. Takšen splet imenujemo semantični splet, splet v katerem so neobdelani podatki logično povezani med seboj in temeljijo na odprtih standardih.

Osnovni gradnik povezanih podatkov je RDF model, ki predstavlja konceptualen način opisa informacij na spletu v obliki stavkov. Da podatki postanejo informacije moramo vsak podatek ontološko razložiti, zato se uporabljajo globalno razviti slovarji, s katerimi opredelimo pomen vsakega podatka. Tako podatki kot slovarji so v RDF stavkih identificirani s pomočjo tehnologije URI. Da bi lahko podatke najboljše izkoristili na globalni ravni so se razvila gibanja, ki se zavzemajo za ustvarjanje LOD zbirk ali natančneje zbirk podatkov, ki so prosto dostopne vsakomur pod določenimi licencami.

Jasno je, da prehod iz spleta v katerem so dostopni samo dokumenti v splet, v katerem se pojavljajo tudi neobdelani podatki zahteva specifične aplikacije. V zaključni nalogi bom predstavil nekaj aplikacij, ki nam pomagajo pri ustvarjanju povezanih podatkov, torej pri pripravi podatkov, njihovi objavi in njihovi uporabi. Ogledali si bomo nekaj najbolj razvitih in poznanih aplikacij, ki uporabljajo povezane podatke in nekaj priporočenih praks pri razvoju novih aplikacij, ki izkoriščajo prednosti povezanih podatkov.

Key words documentation

Name and SURNAME: Marko VUKELIĆ

Title of the final project paper: Applications of linked data

Place: Koper Number of figures: 31 Number of tables: 4

Year: 2014

Number of pages: 62

Number of references: 20

Mentor: Assist. Prof. Iztok Savnik

Keywords: Linked data, semantic web, RDF model, triples, LOD

Abstract:

One of more important challenges in computer science in the last two decades is creation

of World Wide Web in which computers do not only store and retrieve documents, but they can also access data, process it and even more, understand it. Such web is called Semantic web, a web in which data is logically linked, and, it is based on different open standards.

Basic linked data technology is RDF data model developed from graph data models and knowledge representation languages. To give data informative value concepts and facts are linked into ontology using universal vocabularies for representing every single object. Like data in RDF sentences also vocabularies are identified using URI technology. In order to make linked data globally available for use, there are movements that are committed to publish linked data with open license called LOD.

It is reasonable that transition from the document only web to a web that stores structured data requires the change in the development process of Web applications. Designing specific applications is not only needed to make applications that consume linked data, but also applications that can make linked data available.

The presented work describes the design of Semantic web together with some basic tools used for the development of Web applications that can utilize linked data. Some principles and methods for the design of new web applications are presented in more detail. In particular, guidelines for preparing, publishing and using linked data in web applications are given.

Finally, some of the most widely used applications that consume linked data are presented together with some best practices used to make full advantage of linked data. Such

applications are general purpose browsers and search engines developed for Semantic web, and, popular applications such as Facebook and BBC Music that can consume and produce linked data.

ZAHVALA

Želel bi se iskreno zahvaliti svojemu mentorju, doc. dr. Iztoku Savniku, za razpoložljivost in pomoč pri celotni pripravi zaključne naloge.

Posebno se zahvaljujem staršema, ki sta mi vsa leta študija stala ob strani in me finančno in moralno podpirala.

Posebna zahvala gre tudi sestri, ki je nalogo lektorirala.

KAZALO VSEBINE

KAZALO PREGLEDNIC

KAZALO SLIK

SEZNAM KRATIC

FOAF	Friend of a friend
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
LOD	Linked Open Data
OWL	Web Ontology Language
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language

1 UVOD

Splet je korenito spremenil način izmenjave znanja. Omogočil je objavo in dostop do dokumentov v globalnem informacijskem prostoru. Na eni strani imamo brskalnike, ki nam omogočajo raziskovanje spleta s pomočjo hiperpovezav, na drugi strani pa iskalnike, ki dokumente strukturalno analizirajo in poiščejo povezave med njimi, glede na iskalne poizvedbe uporabnikov.

Čeprav je težko oporekati vsem prednostim, ki jih je razcvet spleta prinesel je imel eno veliko pomankljivost, omejen je bil izključno na dokumente. V osnovi so bili podatki na spletu dostopni kot dokumenti v obliki formatov, kot so HTML, CSV ali XML. Z razvojem spleta se je informacijski prostor razširil iz povezav med dokumenti v informacijski prostor povezav med dokumenti in podatki. Podatkom, ki so v takšnem spletu povezani pravimo *povezani podatki*. Čeprav povezani podatki tvorijo v spletu novo poglavje pri izmenjavi podatkov, ni zagotovljeno, da so takšni podatki dostopni za vsakomur. Da bi podatki bili prosto dostopni, so se pojavila gibanja, kot je W3C SWEO Community project, ki se zavzema za objavo povezanih podatkov pod odprtimi licencami. Takšne podatke zadnje čase imenujemo *Linked data*.

S prihodom spleta podatkov, je prišla tudi potreba po novih, drugačnih aplikacijah. Takšne aplikacije lahko delimo na dve osnovni skupini, splošne in specifične aplikacije. Med splošne spadajo aplikacije, ki niso omejene samo na eno kategorijo podatkov, ampak lahko procesirajo vse povezane podatke. V to skupino sodijo npr. brskalniki in iskalniki. Splet podatkov odpira nove možnosti za nastanek specifičnih aplikacij, torej aplikacij, ki so razvite za procesiranje omejene skupine podatkov npr. podatki o nepremičninskih zavarovanjih. Aplikacije povezanih podatkov omogočajo procesiranje množice povezanih podatkov. Le-ta je iz dneva v dan večja. Tovrstne aplikacije ponujajo dodatne možnost, ki jih brez povezanih podatkov nimamo, vendar zahtevajo uporabo specifičnih tehnologij in smernic.

V zaključni nalogi si bomo ogledali sledeče. Razvoj spleta, kaj so povezani podatki in kakšno vlogo imajo v semantičnem spletu. Ogledali si bomo tehnologije na katerih temelji splet podatkov, tehnologije, kot so RDF model, RDF/XML jezik, URI identifikatorji, SPARQL povpraševalni jezik in slovarji. Predstavil bom bistvo gibanja W3C Linked Open Data, ki se zavzema za objavo zbirk podatkov z odprtimi licencami in korake pri objavi lastnih zbirk z odprto licenco. Ogledali si bomo nekaj primerov aplikacij, kot so Tabulator, The Eurostat Company Profiles, Talis Aspire, ki uporabljajo povezane podatke. Predstavil bom nekaj osnovnih korakov pri razvoju aplikacij povezanih podatkov in orodij, ki so nam pri tem na voljo, v nadaljevanju pa natančneje baze, na kaj moramo biti pri tem pozorni ter katere možnosti intehnologije imamo na razpolago.

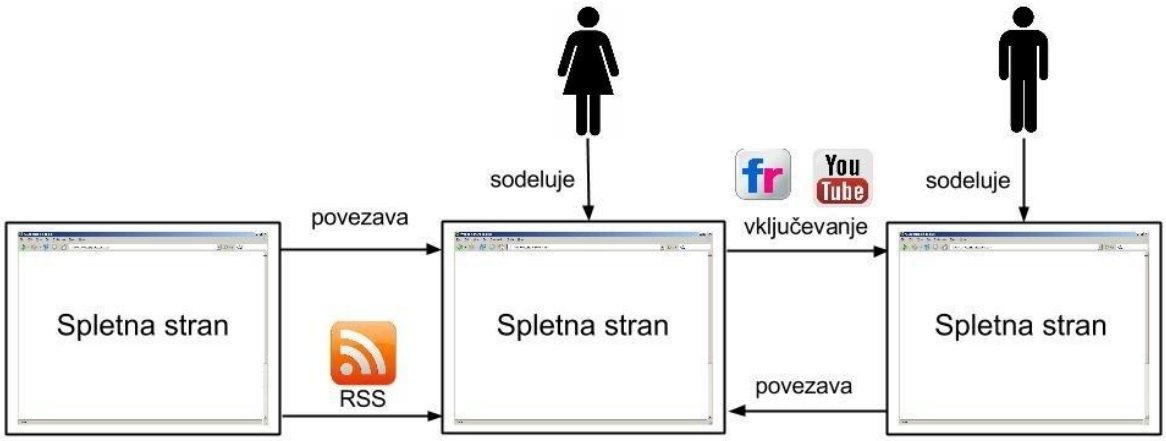
2 SEMANTIČNI SPLET

2.1 Uvod

Prvotno zamisel svetovnega spleta (WWW) je zasnoval Tim Berners-Lee v letu 1989. Če želimo razvoj spleta deliti na različice lahko rečemo, da je najprej nastal t. i. Splet 1.0, ki je bil okolje s pomočjo katerega so računalniški sistemi dostopali do hipertekstnih dokumentov, ki se nahajajo na računalniških terminalih v spletu. Spletni brskalnik ima nalogo, da dokumente v hipertekstu grafično predstavi človeku, v točno definirani obliki, ki jo sestavljajo besedila, slike, povezave itd. Splet 1.0 je velikokrat opisan kot bralni splet podatkov (ang. read-only), kar je močno omejevalo potencial spleta.

Slika 1: Splet 1.0, Splet dokumentov

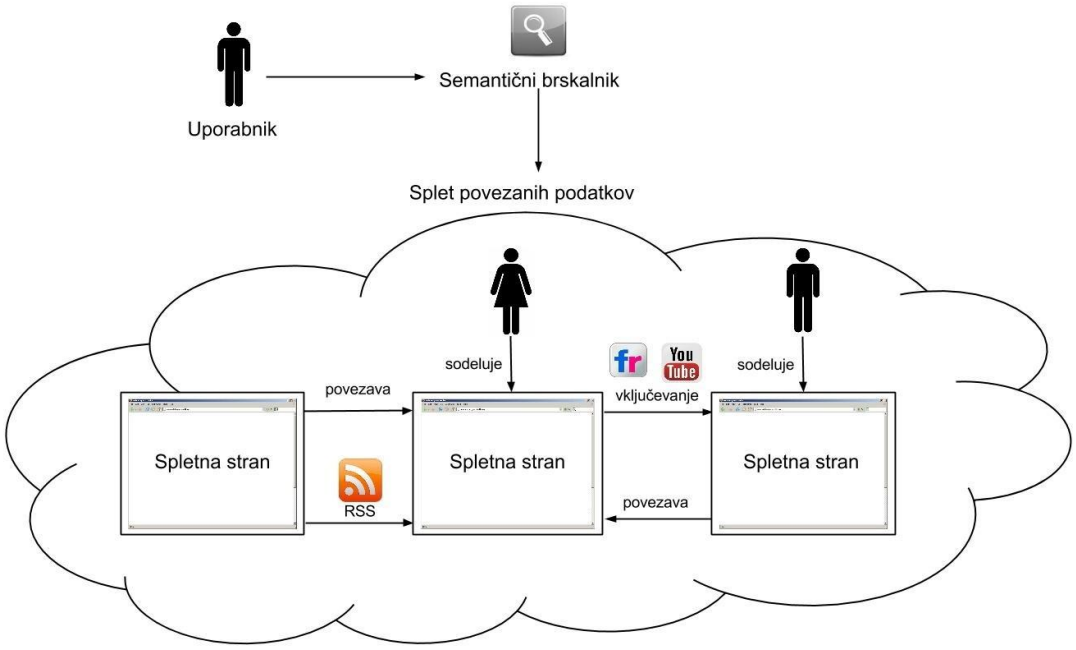
Z evolucijo spleta se je pojavil splet 2.0, ki ni bil namenjen le branju dokumentov, ampak je dodal novo razsežnost. Uporabnikom je omogočil interakcijo na socialnih mrežah, spletnih dnevnikih (ang. blogs), wikijih in izmenjavo oz. vključevanje vsebin v spletne strani. Pomen računalnika pa se pri tem ni bistveno spremenil, opravljal je dve nalogi, izmenjava podatkov in oblikovanje izpisa dokumentov "brskalcem". Računalnik v spletu 1.0 in 2.0 lahko z drugimi besedami označimo kot posrednika, ki zagotavlja, da so podatki dostavljeni in čitljivi človeku, pri tem pa sam podatke ne procesira in ne "razume".



Slika 2: Splet 2.0, Splet vsebin

Ko uporabnik spletnega nakupovanja išče izdelek, ki bi ustrezal vsem želenim kriterijem, kot so cena, trajanje garancijskega obdobja, stroški dostave, rok dobave itd., mora sam poiskati in preveriti oz. primerjati podatke artiklov različnih spletnih trgovin. Izziv, ki se ponuja pri takšnem omejevanju potenciala računalnika, je evolucija spleta, ki bi omogočila računalnikom zbiranje in predvsem logično selekcijo in predstavitev podatkov. Z drugimi besedami bi želeli računalnikom omogočiti razumevanje podatkov, da bi lahko opravili delo, ki ga še vedno opravljamo sami. S takšnim izzivom se ukvarja razvoj spleta 3.0, ki ni več omejen na relacijo človek-računalnik, ampak omogoča tudi relacijo računalnik-računalnik[1].

[1] Več na: http://en.wikipedia.org/wiki/Semantic_Web#Purpose



Slika 3: Splet 3.0, splet podatkov

S spletom 3.0 se pojavi tudi izraz *semantični splet*. Semantični splet je težko natančno definirati, saj si znanstveniki, ki se ukvarjajo z razvojem spleta, njegov obseg predstavljajo drugače. Nekateri uporabljajo izraz semantični splet, kot sinonim spleta 3.0, drugi pa ga opisujejo, kot popolnoma neodvisno implementacijo spleta.

Preglednica 1: Primerjava spleta [2]

Splet 1.0	Splet 2.0	Splet 3.0
splet samo za branje	splet "branje-pisanje"	prenosni osebni splet
45 mio uporabnikov	Več kot 1 mrd	osredotočen na posameznike
osredotočen na poslovne uporabnike	Osredotočen na množico	živi podatki (ang. lifestream)
domače strani (index.html)	blogi	utrjevanje dinamične vsebine
lastitev vsebine	izmenjava vsebine	semantični splet
Britannica online	Wikipedija	pripomočki, d&d prepletene storitve
HTML portali	XML, RSS	uporabniško okolje
spletni vzorci	spletne aplikacije	iGoogle, NetVibes
direktoriji (taksonomija)	označevanjen (folksonomija)	človeško angažiranje
Netscape	Goolge	t. i. advertainment

Zgornja tabela prikazuje bistvene razlike med spleti 1.0, 2.0 in 3.0. Spletu 1.0 lahko rečemo splet samo za branja, saj govorimo o statičnem spletu, pri katerem ima uporabnik možnost samo prebiranja spletnih vsebin in to počne s surfanjem prek direktnih povezav s pomočjo Netscape brskalnika. Spletne vsebine so grajene na tehnologiji HTML in so last ustvarjalca, torej nimamo izmenjave vsebin. Spletu 2.0 lahko rečemo splet branja in pisanja, saj uporabniki niso več izključeni, ampak sodelujejo pri ustvarjanju spleta, tako da prispevajo informacije v obliki spetnih dnevnikov in spletnih strani, kot so Flickr in Youtube. S pomočjo tehnologij, kot so XML in RSS, se je pojavil splet izmenjave vsebine in iskalnika Google. Za lažjo klasifikacijo se pojavi vsebinsko označevanje, kjer ima uporabnik samo možnost označevanja spletnih zazanamkov. Semantični splet se pojavi s Spletom 3.0, ki je osredotočen na posameznika in ponuja individualne iskanike v obliki

[2] Vir: http://www.labnol.org/internet/web-3-concepts-explained/8908/

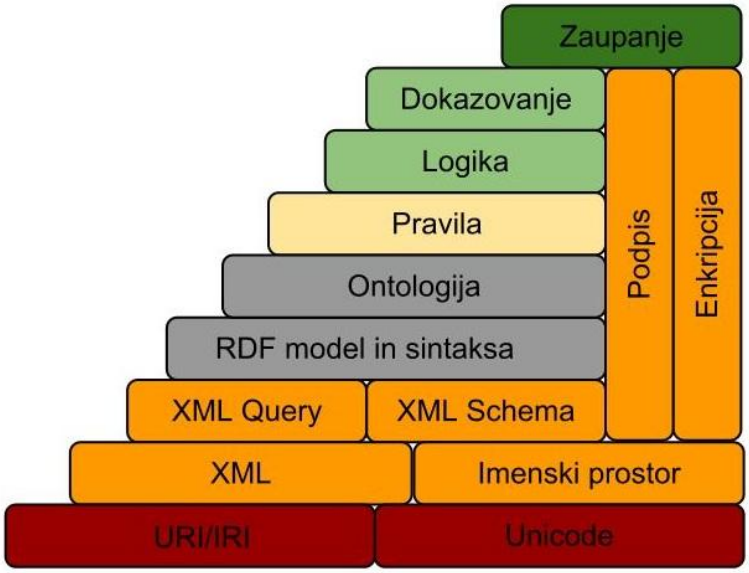
iGoogle. Spletne vsebine niso več lastniške, ampak namenjene za izmenjavo. Oglaševanje poteka preko t. i. advertainment[3] politike, ki uporablja zabavne vsebine za oglaševanje npr. Youtube, iGoogle, Blogger ipd.

Tim Berners-Lee, ki je trenutno direktor na World Wide Web Consortium (W3C), je povedal, da sanja o spletu, v katerem bi računalniki postali sposobni analize vseh podatkov na spletu. Semantični splet, ki bi to moral omogočiti se mora še razviti, vendar ko se bo, bodo mehanizmi trgovanja, birokracije in našega vsakdanja upravljali računalniki, ki bodo komunicirali z računalniki. Inteligentni agenti, o katerih smo ljudje premišljevali, bodo končno postali realnost.

V resnici ne gre več za definicijo spleta kot medmrežje dokumentov, ki bi bili razumljivi samo človeku, ampak spleta podatkov, katerih pomen je razumljiv tudi samemu računalniku in jih zna tudi sam procesirati. Tim Berners-Lee je leta 2001 napovedal, da bo semantični splet prinesel smiselno strukturirano vsebino spletnih strani, na osnovi katere bo okolje v katerem bodo softverski agenti romali med spletnimi stranmi in opravili kompleksna opravila za uporabnike. Oglejmo si primer, agenta, ki bo dostopal spletne strani klinike. Le-ta bo znal več kot le opredelitev ključnih besed zdravljenja, zdravila, terapij (ki bi bile v tem primeru smiselne), ampak bo razbral, da je "Dr. Hartman" zaposlen na kliniki in se tam nahaja ob ponedeljkih, sredah in petkih, in še več, znal nam bo rezervirati termin obiska, vse to brez obširnejše in kompleksnejše umetne inteligence [8].

Če povzamem, bi radi implementirali spletno okolje, ki bi šlo dlje od sintakse podatkov in se osredotočilo na razumevanje podatkov prek semantike. Semantični splet obsega množico tehnologij in standardov, med katerimi velja dobro definirana hierarhija.

[3] Oglaševanje prek zabavnih vsebin.



Slika 4: Semantični splet v slojih

Na najnižjem nivoju sta naslov digitalne vsebine in Unicode standard za kodiranje znakov.

V sredini se nahaja abstrakcija pomena semantičnega spleta, ki obsega več nivojev. Temeljna sta jezik za opisovanje strukturiranih podatkov XML in imenski prostor. XML Query je jezik, namenjen poizvedovanju po XML dokumentih. Prisoten je še jezik XML Schema za določanje sheme v XML podatkovni strukturi. RDF je grafovski podatkovni model, ki omogoča predstavitev podatkov s trojicami oz. na osnovi povezav grafa. Nad njim se nahaja ontologija konceptov, ki definira osnovne tipe podatkov in razmerij med podatki. Pravila dodatno definirajo stvari, ki jih ni mogoče razumeti iz ontologije.

Najvišji nivo, ki predstavlja novejše raziskovalne teme in je potreben za doseganje semantičnega spleta, vsebuje logiko, dokazovanje in zaupanje. Jezik za opis ontologij OWL realizira logični nivo, ki omogoča pisanje pravil. Dokazovanje pa pravila izvede in odloča skupaj z nivojem zaupanja o tem, če je določeno pravilo verodostojno oz. vredno zaupanja [12, 17].

2.2 RDF

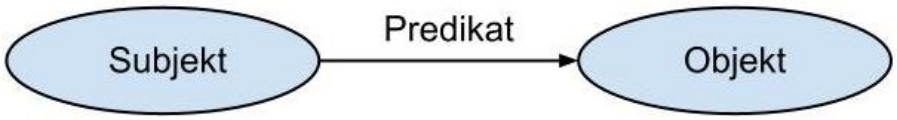
Resource Description Framework (okr. RDF) je grafovski podatkovni model, ki uporablja vozlišča in povezave za predstavitev podatkov na vseh nivojih abstrakcije. Pri opisu dokumenta s podatki, kot so ime avtorja, naslov, naziv ali ime lastnika, datum zadnjega popravka ali oznanila o licencah, predstavlja dokument vozlišče grafa, podatki pa povezave grafa. [19].

Razvoj spleta je pokazal obširnejši spekter uporabe RDF. Ne omogoča samo opis dokumenta, ampak tudi opis informacij vsakemu objektu z definiranim tako imenovanim identifikatorjem URI (Uniform Resource Identifiers). Lahko gre za opis npr. izdeleka na eni od spletnih trgovin, in sicer informacije o razpoložljivosti izdelka, o njegovih specifikacijah, ceni itd.

Namen RDFja je omogočiti izmenjavo in procesiranje informacij med aplikacijami brez, da bi informacije izgubile pomen. RDF ustvarja okolje, v katerem so informacije predstavljene neposredno aplikacijam in ne človeku, kateremu so v končni fazi namenjene.

Celoten koncept RDFja temelji na identifikaciji virov z uporabo identifikatorjev URI in definiranjem njegovih osnovnih lastnosti. Za predstavitev RDF podatkovnega modela uporabljamo različne formalne in neformalne sintakse.

Temelj abstraktne predstavitve RDF modela so t. i. trojice. Trojica ima tri osnovne gradnike subjekt, predikat in objekt. Subjekt predstavlja vir, katerega obravnavamo, predikat definira lastnost, kateri določimo vrednost z objektom. Vsaka trojica opiše neko lastnosti subjekta. Primer trojice je grafično prikazan na spodnji sliki.



Slika 5: Primer trojice

Pomen trojice v RDF modelu je podoben enostavnemu stavku v jezikoslovju, zato lahko RDF trojico imenujemo kar RDF stavek.

Grafična notacija RDF modela je t. i. RDF graf, ki ga sestavlja množica z najmanj eno trojico. Vozlišča, ki v grafu predstavljajo subjekte in objekte, delimo na tri vrste, IRI (Internationalized Resource Identifier), literali in prazna vozlišča. IRI in literali identificirajo vire, ki so lahko dokumenti, predmeti, abstraktni koncepti, števila ali nizi. Literali imajo natanko določen tip podatkov, kot so nizi, števila in datumi. Vozlišča, ki so literali, predstavimo v obliki pravokotnikov, ostala vozlišča pa v obliki ovalov [11].

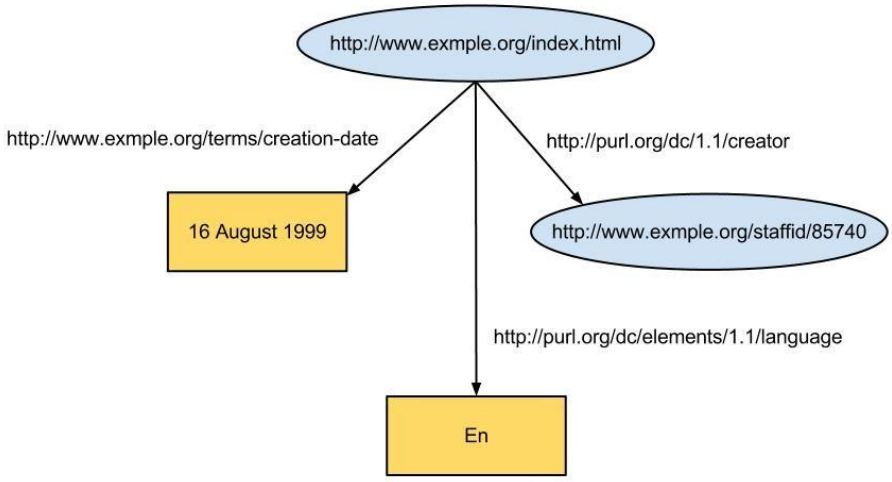
Če bi želeli v RDF modelu povedati sledeče [14].

"http://www.exmple.org/index.html ima ustvarjalca z imenom John Smith."

" http://www.exmple.org/index.html ima datum izdelave katere vrednost je August 16, 1999."

" http://www.exmple.org/index.html je v jeziku katerega vrednost je angleščina"

bi to grafično prikazali s tremi povezanimi trojicami, kot je prikazano na spodnji sliki.

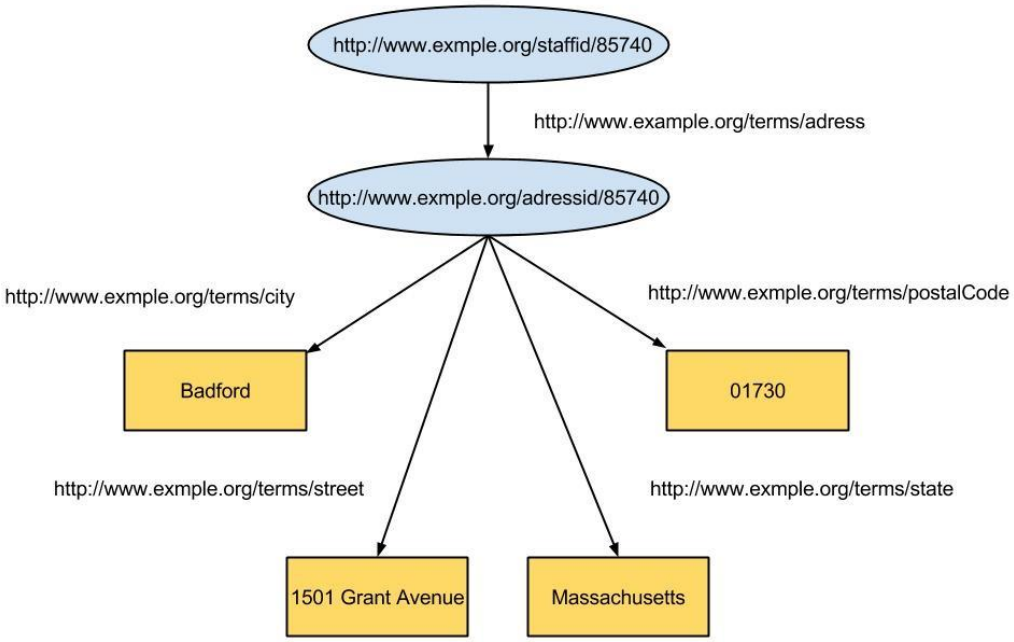


Slika 6: Graf povezanih trojic

Če želimo, da je pomen RDF stavkov enolično določen, moramo vsi uporabljati enake definicije pojmov. Zato uporabljamo globalno razvite slovarje, ki vsebujejo enolično definirane pojme. Kot je razvidno iz zgornjega primera predikate predstavljajo povezave do definicij pojmov v slovarjih. Več si bomo o slovarjih ogledali v poglavju 2.3.

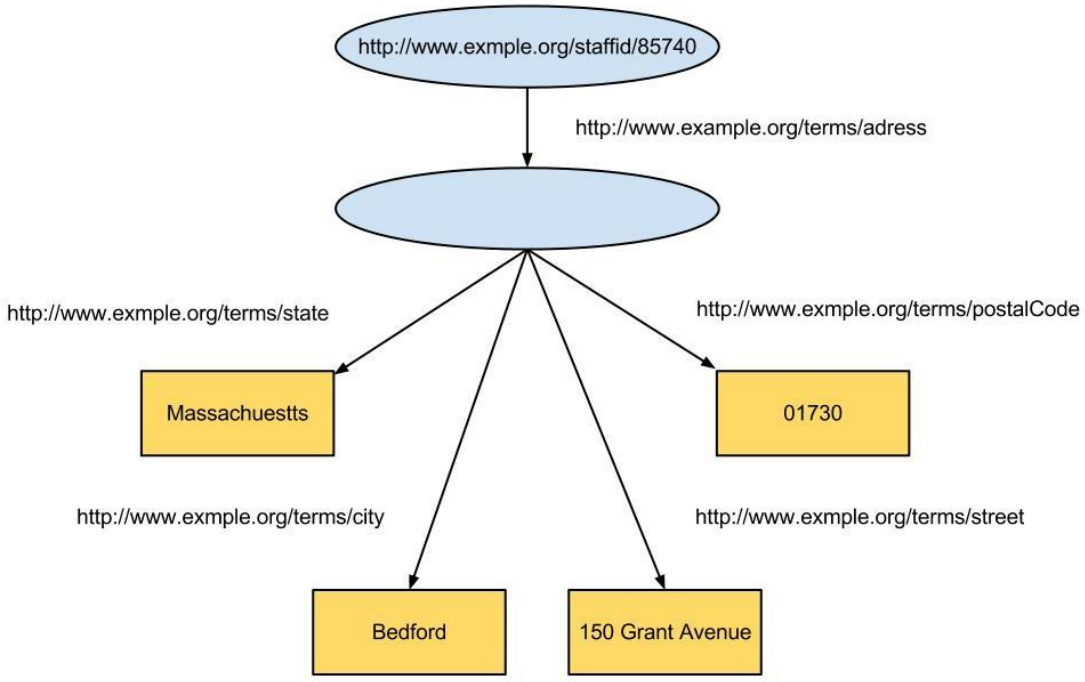
Vrednosti lastnosti niso samo enostavni tipi, ampak so lahko tudi vozlišča grafa. Natančneje so lahko kompleksna drevesa in grafi, ki so sintaktično lahko vgnezdene ali referencirane vrednosti. Primer strukturiranih vrednosti je prikazan spodaj. Za lažjo predstavitev primera bomo uporabili imenske prostore. Okrajšave exstaff:, exadressid: in exterms: uporabljamo namesto polnih naslovov do imen, ki so predstavljeni kot pripone. Več o imenskih prostorih je predstavljeno v poglavju RDF/XML 2.2.2.

```
exstaff:85740 exterms:address exaddressid:85740 .
exaddressid:85740 exterms:street "1501 Grant Avenue" .
exaddressid:85740 exterms:city "Bedford" .
exaddressid:85740 exterms:state "Massachusetts" .
exaddressid:85740 exterms:postalCode "01730" .
```



Slika 7: Strukturirane vrednosti

Za predstavitev informacij na bolj strukturiran način se lahko v grafu večkrat skličemo tudi na prazna vozlišča. Uporaba je prikazana na spodnji sliki.

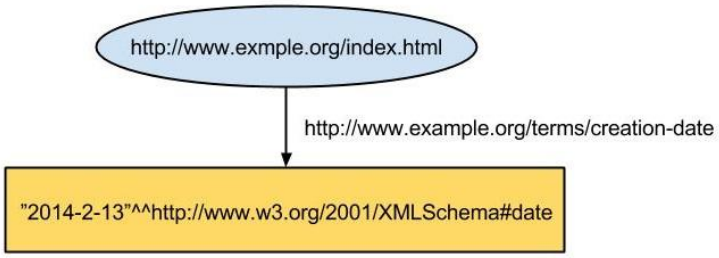


Slika 8: Primer grafa s praznim vozliščem

Podatki v obliki skalarjev so lahko različnih tipov. Ker RDF nima vgrajenih tipov, lahko določimo tip skalarja s pomočjo XML Schema, ki je podrobneje predstavljen v zadnjem odstavku poglavja 2.2.2.

Kako definiramo tip skalarja je prikazano na spodnjem primeru.

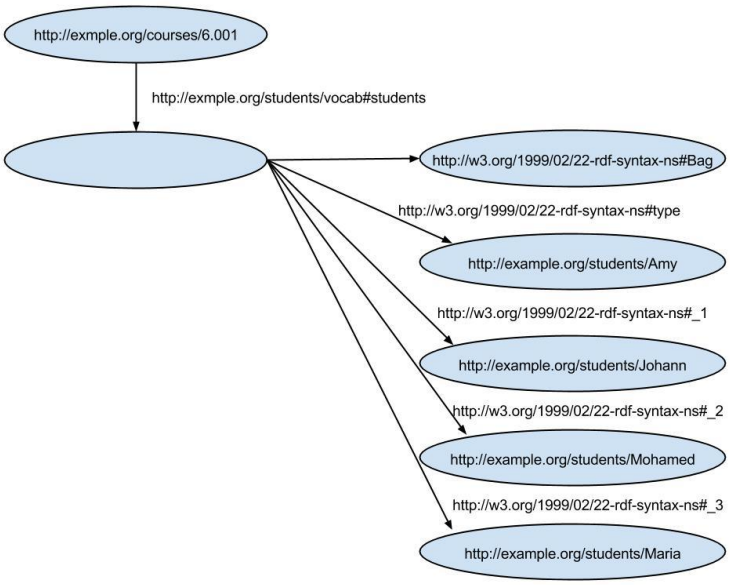
ex:index.html exterms:creation-date "1999-08-16"^^xsd:date.



Slika 9: Definicija tipa skalarja

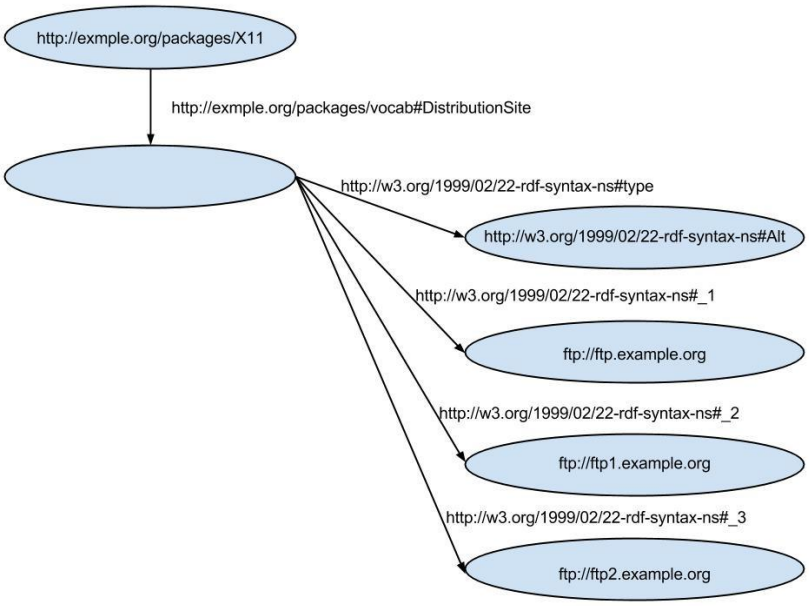
V RDF modelu imajo pomembno vlogo t. i. *kontejnerji*. Kontejnerji nam omogočajo grupiranje virov. Omogočajo nam pisanje izjav o celotnem kontejnerju ali samo njegovih članih. Kontejnerje delimo v tri skupine, te so vreče, alternative in zaporedja.

Vreče uporabljamo, ko imamo skupino objektov, ki ustrezajo določenemu predikatu, kot prikazuje spodji graf.



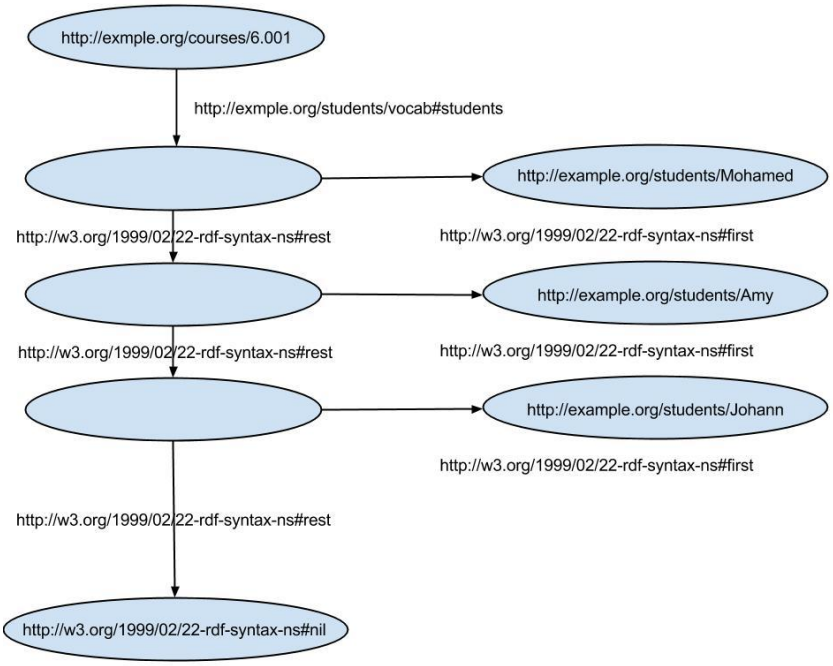
Slika 10: Primer vreče

Alternative uporabljamo za naštevanje različnih možnosti, ki so popolnoma enakovredne, torej med možnosti ni definirana urejenost.



Slika 11: Primer alternativ

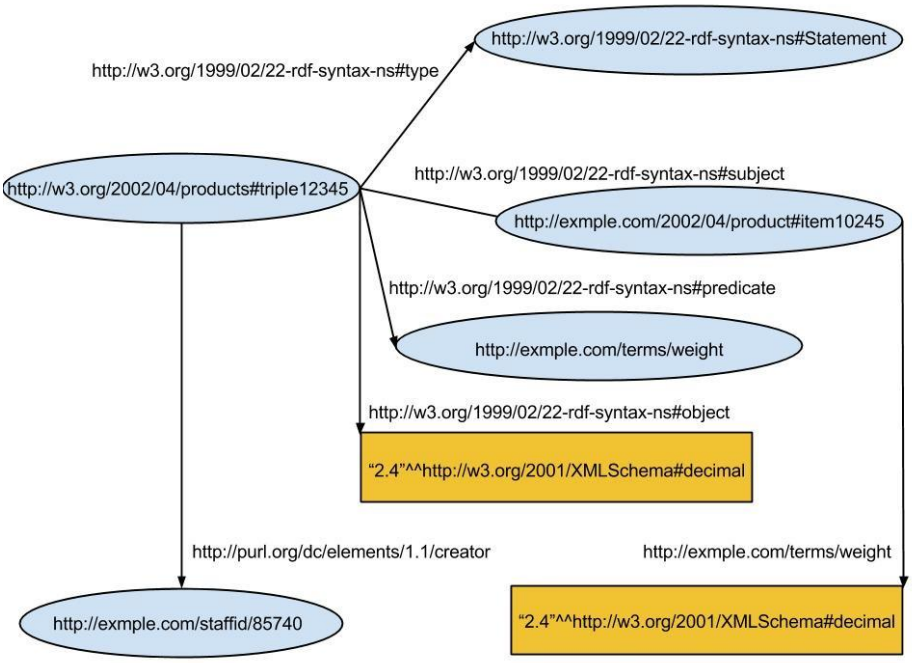
Zaporedja ali kolekcije nam na drugi strani omogočajo naštevanje virov. Tukaj je pomembno vedeti, da kolekcije ne naštevajo vseh možnosti. Dopuščajo, da obstajajo druge možnosti, ki so definirane v neki drugi RDF datoteki na spletu in za katero ne vemo ali obstaja.



Slika 12: Primer kolekcije

Kot že rečeno na začetku poglavaj, pravimo RDF trojicam stavki. Tukaj bi bilo dobro omeniti še stavke višjega reda, gre za stavke, ki govorijo o stavkih. Torej meta-podatki, lastnosti o lastnostih ali celo meta-podatki o meta-podatkih. Obravnavanje abstraktnega

koncepta kot konkretno stvar, ki ima lastnosti, imenujemo reifikacija. Z *reifikacijo* opisujemo podatke drugih stavkov npr. kdo in kdaj je napisal stavek. Sam RDF ima vgrajen slovar za opisovanje stavkov. Pri tem imamo tip rdf:Statement, ki se uporablja za definicijo stavčnih lastnosti subject, predicate in object.



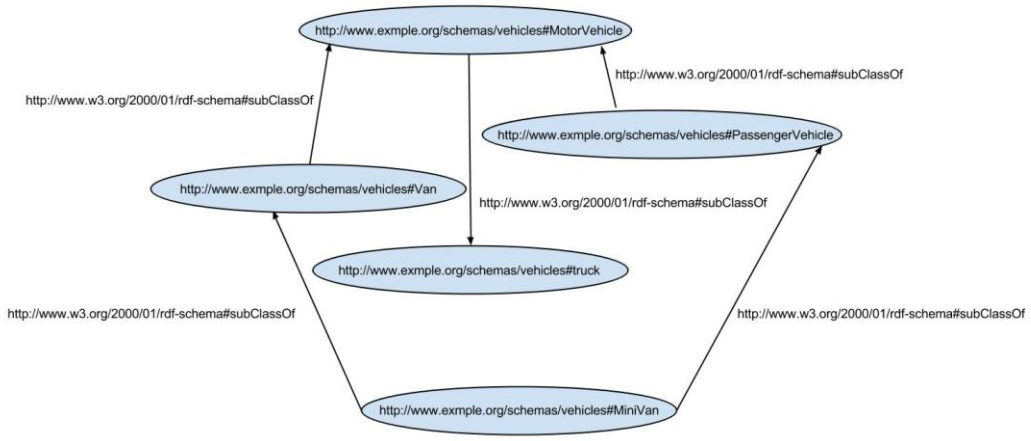
Slika 13: Primer reifikacije

Jezik RDF je lahko osnova za definicijo konceptualnega modela, za predstavitev podatkov in znanja. Da bi lahko definirali konceptualni model je definiran poseben slovar imenovan RDF Schema (okr. RDFS). Razrede, podrazrede, lastnosti in podlastnosti opišemo z naslednjimi lastnostmi:

- rdfs:Class
- rdf:type
- rdfs:subClassOf

Naslednji RDF stavki predstavljajo primer definicije razredov iz klasifikacijske hierarhije vozil.

```
ex:MotorVehicle rdf:type rdfs:Class .
ex:PassengerVehicle rdf:type rdfs:Class .
ex:Van rdf:type rdfs:Class .
ex:Truck rdf:type rdfs:Class .
ex:MiniVan rdf:type rdfs:Class .
ex:PassengerVehicle rdfs:subClassOf ex:MotorVehicle .
ex:Van rdfs:subClassOf ex:MotorVehicle .
ex:Truck rdfs:subClassOf ex:MotorVehicle .
ex:MiniVan rdfs:subClassOf ex:Van .
ex:MiniVan rdfs:subClassOf ex:PassengerVehicle .
```



Slika 14: Primer uporabe RDFS razredov

V nadaljevanju so predstavljeni RDFS gradniki za definiranje RDF lastnosti z uporabo razreda rdfs:property, ki definira lastnosti prve in druge komponente in relacijo podlastnosti s sledečimi gradniki

- rdfs:domain
- rdfs:range
- rdfs:subPropertyOf

Primer relacij lastnosti razredov:

```
ex:Book rdf:type rdfs:Class .
ex:Person rdf:type rdfs:Class .
ex:author rdf:type rdf:Property .
ex:author rdfs:domain ex:Book .
ex:author rdfs:range ex:Person .
```

2.2.1 N3 notacija

V primerih, ko risanje grafov ni primerno, se uporabljajo bolj formalni načini zapisa RDF modela [14]. N3 ali Notation3 spada med osnovne sintakse formalnega zapisa RDF modela. Vsak stavek je sestavljen iz trojice subjekt, predikat in objekt, ki se nahajajo v treh lomljenih oklepajih, zaključi pa se s piko, kot je prikazano na spodnjem primeru.

```
<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/creator>
<http://www.example.org/staffid/85740> .
```

Pomen notacije je sledeč, index.html torej spletno stran je ustvaril staffid 85740. Opisan primer ustreza povezavi v grafu, katere skrajni točki predstavljata subjekt in objekt in povezujejo predikat. Vsak element trojice je v N3 notaciji napisan kot URI, le objekt je lahko literal. Primer N3 notacije, v kateri je objekt oblike literala, je prikazan spodaj.

```
<http://www.example.org/index.html>
<http://www.example.org/terms/creation-date> "August 16, 1999" .
```

Če želimo napisati več lastnosti istega subjekta ločimo lastnosti med seboj s podpičjem, kot je prikazano spodaj.

```
<http://www.example.org/index.html>
<http://www.example.org/terms/creation-date> "August 16, 1999" ;
<http://purl.org/dc/elements/1.1/language> "en" .
```

V primerih, ko stavek uporablja URI naslove do istega dotičnega dokumenta za subjekt, predikat in objekt, lahko uporabimo okrajšavo in namesto, da pišemo celoten URI naslov, napišemo samo del poti, ki se razlikuje od ostalih naslovov. Da bi okrajšavo označili napišemo pred naslov znak #, kot prikazuje spodnji primer [6].

```
<#pat> <#knows> <#jo> .
```

Slovar definiran na spletu uporabljamo tako, da referenciramo gradnike definirane v slovarju z URI naslovi. Za opis prej omenjenega izraza "title" lahko uporabimo URI "http://purl.org/dc/elements/1.1/title".

Za okrajšavo različnih URI naslovov nam N3 omogoča uporabo tako imenovanih predpon (ang. prefix). Delu naslova, ki je enak pri več identifikatorjih rečemo naslovni prostor in mu določimo okrajšavo z oznako "@prefix". Spodnji primer prikazuje definicijo in uporabo predpon za URI naslov "http://purl.org/dc/elements/1.1/title" [5].

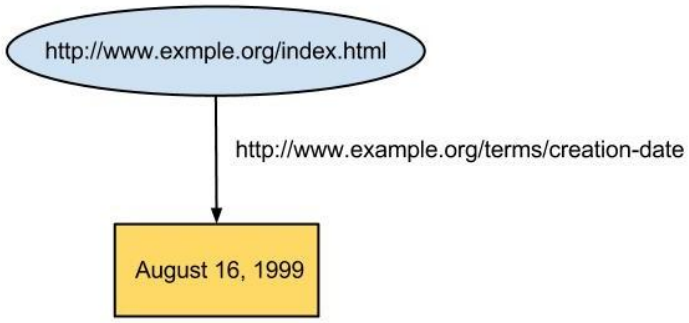
```
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
<> dc:title
  "Naslov dokumenta".
```

2.2.2 RDF/XML

RDF/XML imenujemo formalen način zapisa konceptnega modela RDF z XML sintakso. Imenski prostor v XML datoteki je določen z uporabo predpon QName (Qualified name), kateri določa veljavne identifikatorje znotraj XML imenskega prostora. Za boljšo preglednost bomo v sledečih primerih RDF/XML sintakse uporabljali spodaj definirane QName predpone.

```
prefix rdf:, namespace URI: http://www.w3.org/1999/02/22-rdf-syntax-ns#
prefix rdfs:, namespace URI: http://www.w3.org/2000/01/rdf-schema#
prefix dc:, namespace URI: http://purl.org/dc/elements/1.1/
prefix owl:, namespace URI: http://www.w3.org/2002/07/owl#
prefix ex:, namespace URI: http://www.example.org/
prefix xsd:, namespace URI: http://www.w3.org/2001/XMLSchema#
```

Z uporabo okrajšav si lahko pogledamo še primer RDF modela in njegovega zapisa v RDF/XML sintaksi.



Slika 15: Primer grafa

Zgornjemu grafu je pomensko enakovreden spodnji zapis v RDF/XML sintaksi.

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

  <rdf:Description rdf:about="http://www.example.org/index.html">
      <exterms:creation-date>August 16, 1999</exterms:creation-date>
  </rdf:Description>

</rdf:RDF>
```

Prva vrstica določa verzijo XML zapisa. Sledeči dve vrstici določata naslovni prostor, ki ga bomo uporabljali za zapis stavka. Torej vse značke (ang. tags) s predpono `rdf` so definirane v naslovnem prostoru `http://www.w3.org/1999/02/22-rdf-syntax-ns` in vse zanačke s predpono `exterms` v `http://www.example.org/terms`. Deklaraciji naslovnega prostora sledi zapis RDF modela, ki se začne z `<rdf:Description*>` in konča z `</rdf:Description>`. Vrstica, ki se nahaja med njima predstavlja, v RDF grafu, povezavo iz vozlišča dotičnega subjekta. V našem primeru je subjekt definiran z `about`, `creation-date` definira predikat, objekt pa je v našem primeru literal `"August 16, 1999"`.

V RDF/XML zapisu lahko namesto *rdf:about* uporabimo *rdf:ID*, za specifikacijo objekta. Uporaba t. i. *ID* referenc nam omogoča eksplicitno določanje identifikatorjev objekta, kar je zelo uporabno pri npr. definiciji seznama objektov na določenem naslovu.

```
    <?xml version="1.0"?>
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                      xmlns:ex="http://example.org/stuff/1.0/"
                      xml:base="http://example.org/here/">
        <rdf:Description rdf:ID="snack">
            <ex:prop rdf:resource="fruit/apple"/>
        </rdf:Description>
    </rdf:RDF>
```

Zgornji primer prikazuje uporabo ID-jev, kjer xml:base določa naslov kolekcije objektov, rdf:ID, pa določa objekt "snack". Če vrednost xml:base ni določena, je uporabljen trenuten naslov oz. naslov dokumenta. Naslovu v zgornjem primeru je ekvivalenten dereferencabilni naslov rdf:about="#snack". Z uporabo rdf:ID je mogoče dodeliti določen ID samo enkrat znotraj naslova rdf:base, kar onemogoča, da bi naredili dvojnike [3].

Za definicijo gramatike XML dokumenta se uporablja *XML Schema*. Kot vsi jeziki za opis vsebine XML dokumentov, je namen XML Schema difinirati kateri elementi so v XML strukturi dovoljeni, kateri tipi podatkov so asocirani in kakšno je razmerje med njimi, torej je mogoče definirati elemente, atribute, hierarhijo med elementi, fiksne vrednosti, tipe podatkov in veliko več. Tako lahko uporabljamo XML Schema za preverjanje skladnosti XML dokumentov. Primer je prikazan spodaj[4].

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

2.3 Slovarji

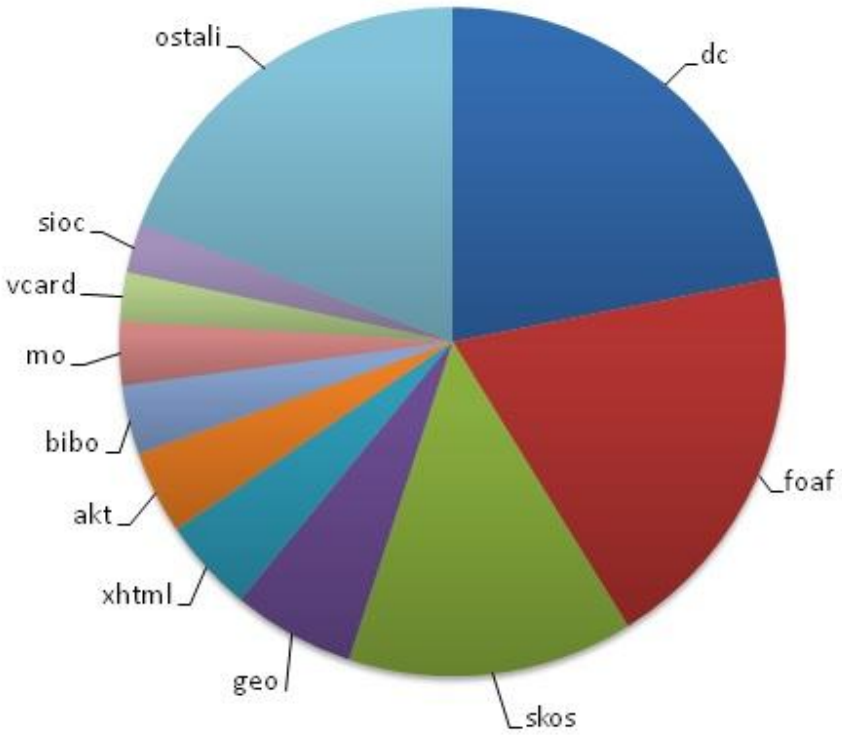
V semantičnem spletu se velikokrat pojavita besedi ontologija in slovarji in je včasih nekoliko zmešnajave pri interpretaciji teh dveh pojmov. Dejansko je slovar formalno določena množica izrazov, ki so razumljivi osebkom istogovorečega jezika. Če želimo vsi razumeti, pomen naprimer besede "title" moramo uporabljati isti slovar. V nasprotju je onotlogija veda, ki določa lastnosti in relacije med enitetami oz. izrazi. Ontologija nam omogoča sestavljanje izrazov v nek vsebinsko obogaten stavek. Ločevanje med slovarjem in ontologijo je izven sveta informacijskih tehnologij jasno in nedvoumno, v nasprotju, ko govorimo o informacijskih tehnologijah in natančneje semantičnem spletu, pa se pojma v večini pokrivata. Vsemantičnem spletu uporabljamo za definicijo lastnosti in relacij med izrazi ravno slovarje.

[4] Več o uporabi XML Schema na naslovu http://www.w3schools.com/schema/schema_howto.asp

Pri širjenju semantičnega spleta je bistvenega pomena, da vsi uporablajmo iste slovarje. Spodnja preglednica predstavlja delež uporabe najbolj razširjenih slovarjev.

Preglednica 2: Najbolj uporabljani slovarji

Predpone slovarjev	Povezave slovarjev	Delež uporabe podatkov
dc	http://purl.org/dc/elements/1.1/	92 (31.19 %)
foaf	http://xmlns.com/foaf/0.1/	81 (27.46 %)
skos	http://www.w3.org/2004/02/skos/core#	58 (19.66 %)
geo	http://www.w3.org/2003/01/geo/wgs84_pos#	25 (8.47 %)
xhtml	http://www.w3.org/1999/xhtml/vocab#	19 (6.44 %)
akt	http://www.aktors.org/ontology/portal#	17 (5.76 %)
bibo	http://purl.org/ontology/bibo/	14 (4.75 %)
mo	http://purl.org/ontology/mo/	13 (4.41 %)
vcard	http://www.w3.org/2006/vcard/ns#	10 (3.39 %)
sioc	http://rdfs.org/sioc/ns#	10 (3.39 %)



Slika 16: Grafikon uporabljanosti slovarjev po deležih[5]

Kot je razvidno iz grafikona je najpogosteje uporabljen slovar DC oz. slovar organizacije Dublin Core. Slovar je prvotno nastal kot zbirka metapodatkovnih značk za opis virov na metapodatkovnem nivoju. Z razvojem spleta se je izkazalo, da ima slovar visoko uporabnost v semantičnem spletu [15].

Slovar v osnovi obsega 15 lastnosti

- contributor - so avtor
- coverage - obseg vsebine vira
- creator - odgovorni za nastanek vsebine
- date - datum nekega dogodka povezanega z vsebino
- descrition - opis vsebine
- format - format datoteke, fizične dimenzije
- identifier - unikatna referenca na vir
- language - jezik vsebine
- publisher - odgovorni za objavo vira
- relation - povezava na soroden vir
- rights - informacije o pravicah vira
- source - referenca na izvirnik vsebine
- subject - ključne besede ali klasifikacija, ki predstavlja vsebino vira
- title - naslov vira
- type - kategorija vsebine vira

Primer

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/">

 <rdf:Description rdf:about="http://example.org/123">
   <dc:title xml:lang="en">Learning Biology</dc:title>
 </rdf:Description>

</rdf:RDF>
```

[5] Vir: http://lod-cloud.net/state/#terms

2.4 SPARQL

SPARQL ali natančneje SPARQL Protocol and RDF Query Language je jezik, namenjen RDF povpraševanju [16]. Standardiziran in priporočen s strani World Wide Web Consortium, je SPARQL postal eden ključnih tehnologij na področju semantičnega spleta. SPARQL deluje na osnovi grafovskih vzorcev, ki so podobne RDF trojicam (subjekt, predikat, objekt) le, da so ti lahko tudi spremenljivke.

Grafovski vzorci se ujemajo s podgrafi RDF modela natanko tedaj, ko lahko zamenjamo gradnike trojic v RDF grafu s spremenljivkami grafovskega vzorca, brez da bi se pri tem spremenil pomen RDF grafa.

2.4.1 Enostavne poizvedbe

V primerih enostavnih poizvedb se najpogosteje, v SPARQL povpraševanjih pojavita samo dva gradnika, SELECT in WHERE. SELECT določa obliko izpisa rezultata poizvedbe, medtem, ko WHERE definira iskani vzorec, s katerim se morajo ujemati podatki RDF podgrafa. Poleg SELECT imamo tudi druge načine izpisa rezultatov, le-te bomo obravnavali kasneje.

Spodaj je podan primer podatkovnega modela.

```
<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title>
"SPARQL Tutorial" .
```

Oglejmo si primer SPARQL povpraševanja nad zgoraj definiranimi podatki.

```
SELECT ?title
WHERE
{
  <http://example.org/book/book1>
<http://purl.org/dc/elements/1.1/title> ?title .
}
```

Poizvedba nam kot rezultat vrne naslov *SPARQL Tutorial*.

S poizvedbo iščemo rešitev, ki bi povezala spremenljivko ?title s trojico subjekta book1 v podatkovnem grafu. Spremenljivke so označene s predznakom "?". Del SELECT definira izpis spremenljivke title, WHERE pa določa vzorec vseh vnosov v podatkovnem modelu, ki imajo za subjekt book1 definiran predikat title. Poizvedba izpiše naslov knjige (ang. title) book1 v podatkovnem modelu. V dotičnem primeru samo "SPARQL Tutorial".

2.4.2 Ujemanje vrednosti

Bistvo vsakega povpraševalnega jezika je iskanje vrednosti na osnovi ujemanja vzorcev. Torej iščemo podatke trojic, katerih določene vrednosti že poznamo in si s tem omejimo izpis želenih rezultatov. Oglejmo si konkreten primer za sledeče podatke.

Primer poizvedbe:

```
SELECT ?v WHERE { ?v ?p "cat"@en }
```

V tem primeru iščemo subjekt, ki ima za nek nedefiniran predikat definiran objekt "cat"@en. Opozoriti je treba, da je @, v literalu predznak za jezikovne značke v SPAQL-u. Pomen besede cat v angleškem jeziku je mačka, torej iščemo nekaj (?v) kar je na nek način povezano z mačko.

Primer poizvedbe:

```
SELECT ?v WHERE { ?v ?p "42"^^xsd:integer }
```

Iščemo subjekt, ki ima za nek nedefiniran predikat definiran objekt "42"^^xsd:integer. Pri tem je treba opozoriti, da predznak ^^ pri literalu napoveduje definicijo tipa literala, torej v primeru xsd:integer, kjer je xsd predpona za http://www.w3.org/2001/XMLSchema#, kar pomeni, da gre za primitivni tip.

Primer poizvedbe

```
SELECT ?v
WHERE { ?v ?p "abc"^^<http://example.org/datatype#specialDatatype> }
```

Podobno kot v prejšnjima dvema primeroma iščemo subjekt, ki ima za nek nedefiniran predikat definiran objekt "abc"^^dt:specialDatatype. Torej gre za vrednost, ki je poljubnega tipa.

2.4.3 Kreiranje

Vsaka poizvedba, ki je bila do zdaj obravnavana, nam je izpisala rezultate, kot jih določa SELECT izpis, in sicer v obliki spremenljivk (ang. variable binding). Kot že omenjeno imamo več oblik izpisov, med katerimi je tudi oblika, ki jo določa CONSTRUCT izpis. Slednji nam omogoča izpis, ki ga sami sestavimo.

Primer poizvedbe CONSTRUCT

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

CONSTRUCT { ?x foaf:name ?name }
WHERE  { ?x org:employeeName ?name }
```

Rezultat

```
@prefix org: <http://example.com/ns#> .

_:x foaf:name "Alice" .
_:y foaf:name "Bob" .
```

2.4.4 Dodatne možnosti povpraševanja

Kot vsi povpraševalni jezik tudi SPARQL omogoča dodatne možnosti pri iskanju rešitev s pomočjo različnih operacij. Najpogosteje so operacije FILTER, OPTIONAL in UNION, ki se jih uporablja znotraj dela WHERE sekcije v poizvedbi.

Operacijo FILTER uporabljamo za omejevanje skupine rešitev. Spodnji primer prikazuje uporabo operacije FILTER v povezavi z opcijo regex, ki omogoča iskanje enostavnih literalov brez upoštevanja jezikovnih značk, obravnavanih v sekciji 2.3.2.

Primer uporabe:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE { ?x dc:title ?title
FILTER regex(?title, "^SPARQL")
}
```

Rezultat poizvedbe obsega vse naslove, ki se začnejo z nizom SPARQL.

Zelo uporabna je operacija OPTIONAL, za katero bi lahko rekli, da deluje nasprotno operaciji FILTER. Medtem,ko nam operacija FILTER dodatno omeji množico rešitve, nam operacijaa OPTIONAL množico rešitev razširi. OPTIONAL nam omogoča poleg izpisa rešitev še izpis dodatnih informacij, če so le-te na voljo, ne vpliva pa na osnovni grafovski vzorec.

Primer uporabe:

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

Grafovski vzorec zajema vse trojice, ki imajo določeno spremenljivko ?name, ne glede na to, če imajo določeno tudi spremenljivko ?mbox. Če je ?mbox spremenljivka tudi določena jo izpiše.

Operacijo, ki je tudi vredno omeniti je UNION. Ta operacija nam omogoča združitev več različnih grafovskih vzorcev v eni poizvedbi. Med rešitve spadajo vsi tisti deli RDF podatkov, ki ustrezajo enemu ali večim grafovskim vzorcem.

Primer uporabe:

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

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

3 POVEZANI PODATKI

Povezani podatki omogočajo strukturiranje, objavo in povezovanje podatkov, informacij in znanja na Semantičnem spletu z uporabo URI in RDF tehnologij.

Splet dokumentov je nastal, ko smo ljudje začeli povezovati HTML dokumente med seboj s pomočjo hiperpovezav. Ko smo začeli povezovati podatke med sabo v obliki RDF trojice je nastal splet podatkov. Seveda splet podatkov ni nastal z uporabo naključnih tehnik za povezavo podatkov, Tim Berners-Lee je podal štiri principe povezanih podatkov, ki so izraženi v obliki pravil.

- uporaba URI-jev za identifikacijo objektov,
- uporaba HTTP URI, kar omogoča sklicevanje iz strani uporabnikov in uporabniških agentov,
- zagotavljanje informacij, ki so dostopne prek URI-jev v RDF in SPARQL tehnologij,
- pri objavi podatkov na spletu je potrebno dodajati URIje povezanih podatkov.

Pri prvem pravilu je za semantični splet pomembno, da uporabljamo univerzalni nabor simbolov.

Drugo pravilo narekuje uporabo tehnologije dereferencabilnih identifikatorjev HTTP URI.

Tretje pravilo je redko upoštevano v večjih podatkovnih bazah. Kot primer lahko vzamemo FOAF (Friend of a friend), ki je ontologija uporabljena pri ustvarjanju spletnih socialnih omrežij. Večino spletnih omrežij ne omogoča povezovanja in deljenja podatkov z ostalimi spletnimi mesti. Nasprotno lahko vzamemo primer LiveJournal in Opera Community, ki svoje podatke prosto objavljata v RDF obliki. Uporabnik ima celo možnost, da si izbere svoj URI kar pomeni, da lahko prek URI-ja dostopamo do vseh ostalih informacij uporabnika, npr. imen njegovih prijateljev, ki imajo istočasno svoje URI-je, da lahko pridobimo informacije o njihovih prijateljih. Ravno tako se URI uporabi na različnih spletnih straneh, ki ponujajo ostale informacije, kar ustvarja mrežo povezanih podatkov.

Četrto pravilo narekuje obvezo objavljanja zunanjih povezav, kar bi morala biti že praksa v spletu. Med podatke je nujno potrebno vključevanje zunanjih URI identifikatorjev, ki omogočajo pridobivanje dodatnih informacij iz tujih baz in v končni fazi ustvarjanje globalne mreže povezanih podatkov [4].

3.1 LOD

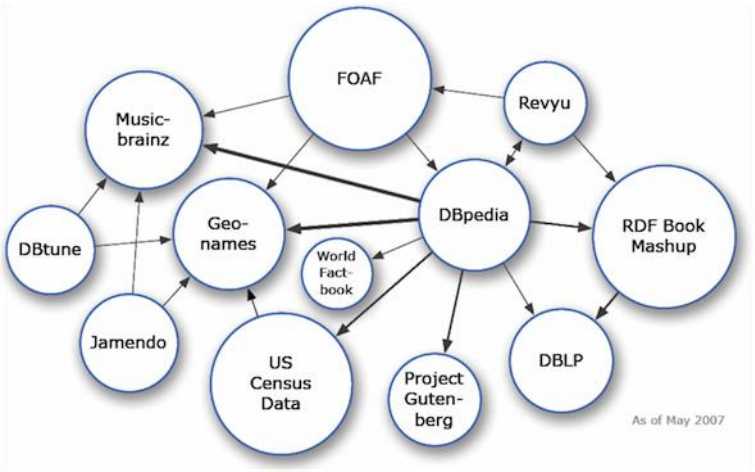
Povezani odprti podatki (ang. linked open data ali krajše LOD) so povezani podatki, ki so prosto dostopni in jih lahko vsakdo uporabi za pridobivanje in širjenje informacij, popolnoma brez omejitev. Ni nujno, da so povezani podatki tudi odprti oz. prosto dostopni vsakomur, lahko so povezani podatki namenjeni samo eni osebi ali neki omejeni skupini, npr. zaposlenim v zdravstvu. Odprti povezani podatki morajo biti objavljeni z odprto licenco, ki potrjuje prosto uporabo povezanih podatkov. Primer take licence je npr. "Open Government Licence", ki omogoča uporabo povezanih podatkov angleške vlade[6].

Podatkovne baze, kot so Wikipedija, Wikiknjige, Geonames so že del gibanja odprtih podatkov pod licenco Creative Commons in Talis. Licence odprtih podatkov zagotavljajo vsakomur pravno osnovo za uporabo, širjenje in spreminjanje podatkov.

Pri objavi odprtih povezanih podatkov je pomembna objava licence, v nasprotnem primeru prosta uporaba podatkov ni omogočena in podatki niso del LOD. Na voljo je več licenc[7], ki so prilagojene željam lastnikov podatkov.

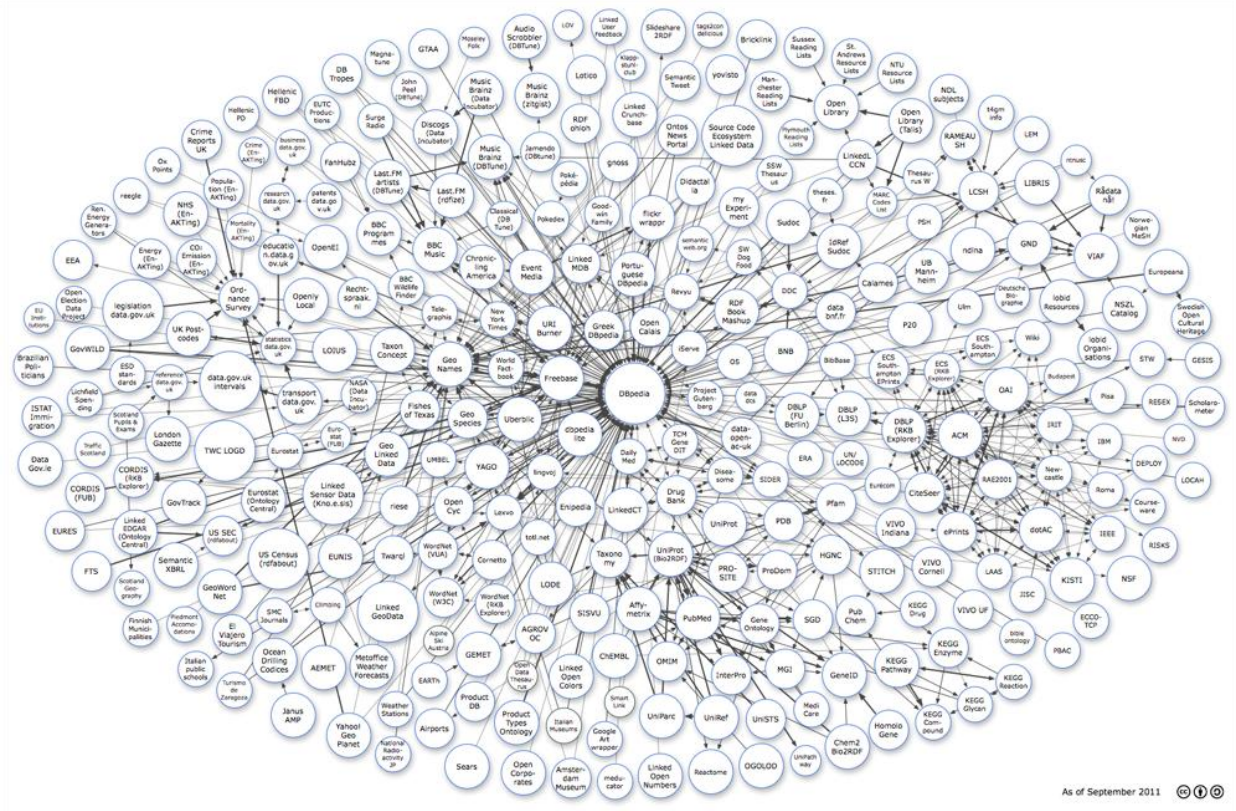
Potrebno je omeniti projekt W3C SWEO (Semantic Web Education and Outreach), skupnost, ki skrbi za širjenje semantičnega spleta in s tem povezavo odprtih podatkov. Projekt se zavzema za objavo odprtih baz v RDF obliki in povezavo podatkov med različnimi bazami. Projekt je do leta 2011 omogočil medsebojno povezavo več kot 295 podatkovnih baz [1].

Na spodnjih slikah je prikazan razvoj LOD zbirk skozi čas.



Slika 17: LOD projekt leta 2007[8]

[6] Več o OGL na naslovu: http://data.gov.uk/
[7] Več o izbiri in načinu licenciranja je moč najti na spletni strani http://opendatacommons.org.
[8] Vir: http://en.wikipedia.org/wiki/File:Linking-Open-Data-diagram_2007-09.png



Slika 18: LOD projekt leta 2011[9]

V želji za popoln semantični splet je Tim Berners-Lee razvil lestvico ocenjevanja odprtih povezanih podatkov. Vsakdo, ki želi odprte povezane podatke narediti čim bolj učinkovite, si lahko pomaga z lestvico, kjer si ogleda koliko zvezdic na lestvici od 1 do 5 dosega njegova podatkovna baza.

Preglednica 3: Vrednotenje baz

*	Dostopno na spletu, z odprto licenco, v kateri koli obliki.
**	Dostopno v obliki, ki omogoča računalniško procesiranje (npr. Exel namesto skenirana tabela)
***	Enako kot za ** le, da so podatki v nelastniškem oz. nekomercialnem formatu, kot je npr. CSV namesto Exel datoteke.
****	Vse zgoraj našteto in uporaba W3C standardov RDF in SPARQL za identifikacijo stvari, kar omogoča ostalim povezovanje in uporabo naših podatkov.
*****	Vse zgoraj našteto in povezovanje svojih podatkov s podatki iz ostalih zbirk.

Pomembno vlogo pri širjenju semantičnega spleta ima izbor slovarjev. Da bi omogočili aplikacijam enostavno razumevanje povezanih podatkov, moramo pri kreaciji povezanih podatkov uporabljati najbolj razširjene slovarje, ki jih uporablja čim več aplikacij in baz.

[9] Vir: http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

Skoraj vse podatkovne baze v LOD oblaku uporabljajo W3C RDF slovarje in RDF sheme, vendar istočasno uporablja več kot polovica podatkovnih virov dodatne nelastniške slovarje [4].

3.1.1 Orodja

S širjenjem povezanih podatkov se je pokazala potreba po orodjih, ki bi nam olajšala korake pri kreaciji povezanih podatkov.

Orodja, ki nam pomagajo pri ustvarjanju povezanih podatkov, lahko delimo v dve osnovni skupini. V prvo spadajo RDF pretvorniki (ang. RDFizers), ki nam pomagajo pri pretvorbi podatkov različnih formatov v RDF, sledijo orodja za objavo, ki so, kot samo ime pove, namenjeni objavi podatkov.

3.1.1.1 RDF pretvorniki

Velika ovira pri doseganju odprte podatkovne baze s petimi zvezdicami je ta, da so že ustvarjene zelo obširne podatkovne baze, ki niso v RDF obliki in je njihova pretvorba v RDF model zelo zahtevna. K enostavnejši pretvorbi podatkovnih baz pripomore nekaj orodij, kot so npr. RDF prevejelniki.

RDF pretvorniki so programsko orodje, ki nam pomagajo pri pretvorbi podatkovnih baz v RDF baze. Programsko orodje deluje popolnoma samostojno v primerih, ko so podatki dobro strukturirani oz. je semantiko podatkov moč avtomatizirano razbrati, npr. oblika elektronske pošte je dobro strukturirana, enostavno je razbrati pomen podatkov. Opredeljen je pošiljatelj, prejemniki, zadeva, vsebina in priponka.

Pri podatkovnih bazah, katerih semantični pomen ni mogoče avtomatizirano razbrati, je potreben človeški poseg.

Spodaj je naštetih nekaj pomembnejših pretvorniki

JPEG -> RDF poišče vse jpeg datoteke v direktoriju, njihove EXIF in IPCT metapodatke razčleni in ustvari RDF/N3 format zbranih informacij.

EMail -> RDF pretvori elektronsko pošto v mbox formatu v RDF/XML.

BibTEX -> RDF pretvori BibTEX datoteko v RDF/XML.

Garmin -> RDF pretvori GPS podatke iz Garmin sprejemnika v RDF.

CSV -> RDF pretvori datoteke, ki vsebuje z vejico ločene vrednosti v RDF.

Outlook -> RDF pretvori Microsoft Outlook koledar v RDF.

Java -> RDF sprehodi se po vsebini java datotek, poišče vse klice metod in ustvari RDF/N3 format odvisnosti med razredi in java paketi [ang. java package]

Javadoc -> RDF tako imenovan doclet program, ki ustvari metapodatke o strukturi razredov, metod, komentarjev java datotek v formatu RDF/N3.

WWW -> RDF Virtuoso Sponger pretvori obširen vir spletnih podatkov v RDF, med temi so (X)HTML strani, spletne storitve, kot so npr. Google API in Flickr ter datoteke tipa MS Office in PDF.

3.1.1.2 Orodja za objavo

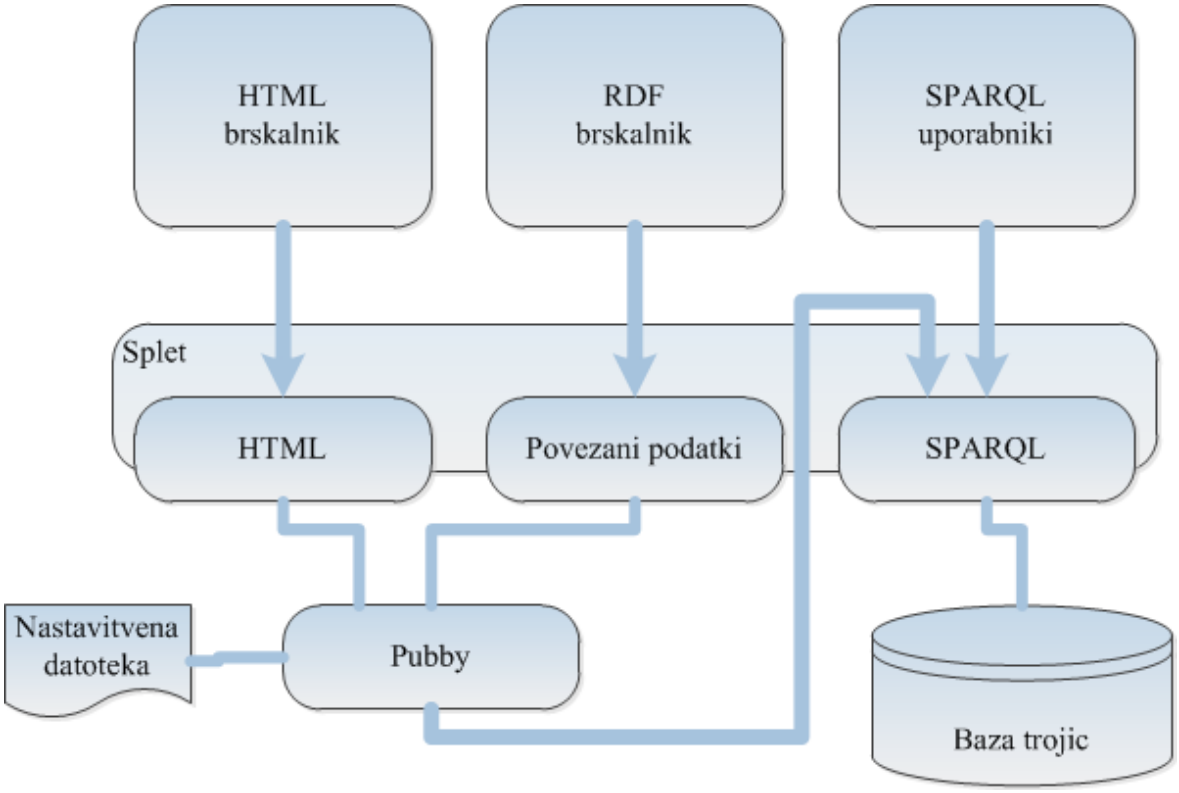
Ko imamo podatkovno bazo v RDF formatu je na voljo množica aplikacij, ki nam zagotovijo objavo povezanih podatkov in podporo SPARQL končne točke (ang. endpoint). SPARQL končna točka skrbi za obdelavo SPARQL poizvedb in streženje rezultatov. Primeri programov za podporo objavi RDF so: D2R Server, Linked Media Framework, Joseki, Pubby, Elda, Paget ipd. [1].

Če za primer vzamemo program Pubby, ki doda SPARQL končni točki funkcionalnost strežnika povezanih podatkov. S takšnim strežnikom lahko komunicirajo RDF brskalniki, kot so Tabulator in OpenLink Browser.

Med drugim Pubby opravlja sledeče naloge:

- zagotavljanje vmesnika povezanih podatkov za serverje s SPARQL protokolom,
- dereferenciranje URI naslova serverjevega imenskega prostora,
- predstavitev podatkov o virih v HTML obliki,
- rokovanje s HTTP 303 preusmeritvami.

Pubby deluje kot vmesnik za komunikacijo prek SPARQL končne točke baze tudi za RDF in HTML brskalnike, kar prikazuje spodnja slika.



Slika 19: Arhitektura delovanja SPARQL končne točke v Pubby orodju

V večini SPARQL baz URI naslovi niso dereferencabilni, kar pomeni, da niso dostopni za semantične brskalnike. V ta namen ima Pubby funkcijonalnost preslikovanja naslovov, ki deluje na nivoju SPARQL končne točke, ta omogoča preslikovanje nedereferencabilnih v dereferencabilne naslove [10].

Npr. naslov tag:dbpedia.org,2007:Berlin bo po funkciji preslikovanja sledeč http://myserver.org:8080/pubby/Berlin v primeru, ko je Pubby server na naslovu http://myserver.org:8080/pubby/.

3.2 Splet podatkov

Splet na katerem so podatki ne samo dostopni, ampak tudi povezani med seboj v obliki povezanih podatkov, imenujemo splet podatkov. To pomeni, da je splet podatkov izjemno velika množica povezanih RDF stavkov v obliki grafa. Stavki v tem grafu lahko obravnavajo različne tematike, kot so vreme, medicina, finance, novice o prometu itd.

Splet podatkov izvira iz projekta imenovanega W3C Linked Open Data, ki je gibanje ustanovljeno v začetku leta 2007 in stremi k pretvorbi in objavi prostolicenčnih podatkov, ki jih najdemo na spletu. Pri tem lahko sodeluje vsak, ki želi objaviti podatke in upošteva principe povezanih podatkov.

3.2.1 Topologija

Tehnologija povezanih podatkov pokriva obširno množico tematskih področij. Podatkovne baze delimo na sedem skupin [13]:

- *uporabniške vsebine:* podatkovne baze, ki so bile prve prenešene v splet podatkov so bile baze, ki so nastale za spletne strani v Spletu 2.0, torej strani z velikim številom uporabnikov in uporabniških vsebin. Naprimer podatkovna baza iz spletne strani Flickr je bila prenešena v FlickrWrappr podatkovno bazo, ki deluje v povezavi z DBpedijo;
- *znanost o življenju:* na področju znanosti o življenju so povezani podatki osvojili velik pomen. Naprimer projekt W3C *Linking Open Drug Data* je več farmacevtskim družbam zagotovil sinergijo z objavo in izmenjavo raziskovalnih podatkov pod odprto licenco. Še pomembnejši je projekt Bio2RDF[10] katerega velikost podatkovne baze obsega milijardo trojic iz devetnajst podatkovnih baz,
- *knjižnice* so nedvomno izjemno velik vir informacij zato je njihov pojav v spletu podatkov samoumeven. Gre za povezavo katalogiranega gradiva na osnovi naslovov, vsebine, področja, lokacije na globalni širini. Smisel integracije takšnih registrov v splet podatkov je, da so registri neposredno povezani z informacijami tretjega tipa, kot so naprimer slikovni in video arhivi;
- *mediji:* med prvimi, ki so začeli uporabljati povezane podatke so bili mediji. Britanski BBC je RDF uporabil za objavo seznama oddaj in njihovega opisa[11] ter za objavo diskografije vsakega glasbenika, katerega glasba je bila predvajana na BBC radio postaji. Baza je povezana z DBpedio, kar omogoča aplikaciji procesiranje podatkov o izvajalcu in kompleksnejša iskanja podobnosti med glasbeniki ter njihovo diskografijo;
- *vladni podatki:* izjemno obširen spekter podatkov obsegajo vladni podatki. Le-ti obsegajo veliko področij, kot so naprimer statistike o izobraževanju, kriminalu, financah, registre podjetij, rezultate volitev. Z željo po večji transparentnosti so svoje podatke začeli povezovati v splet Avstralija, Nova Zelandija, ZDA in Velika Britanija in veliko ostalih. Dosegljivost ter enostavno procesiranje vladnih podatkov ponuja vsakomur analizo in odkrivanje novih trendov;
- *geografski podatki:* lahko povežemo z vsakim objektom. Na tem področju je najbolj znana baza GeoNames[12], ki velikokrat služi kot vir podatkov ostalim bazam, ki vsebujejo geografsko komponento. Danes GeoNames postreže kar 30

[10] Več na: https://github.com/bio2rdf/bio2rdf-scripts/wiki
[11] http://www.bbc.co.uk/programmes
[12] Spletni aslov: http://www.geonames.org/

milijonov poizvedb dnevno. Podobna podatkovna baza na geografskem področju je LinkedGeoData[13], ki zbira informacije iz OpenStreetMap projekta. Tako GeoNames kot LinkedGeoData sta povezana z bazo DBpedije;

- *medkategorija*: nekatere baze, ki so se pojavile med prvimi na spletu podatkov ni bilo mogoče opredeliti, saj se razprostirajo čez več kategorij. Točno ta kategorija je ključnega pomena pri globalizaciji podatkov in nastanku učinkovitega spleta podatkov, saj omogočajo povezavo baz različnih kategorij in zmanjšuje fragmentacijo podatkov. Primer medkategorizirane baze je DBpedija, ki svojo bazo ustvarja z ekstrakcijo podatkov iz Wikipedije.
-

Spodnja preglednica prikazuje razširjenost spleta podatkov po kategorijah.

Preglednica 4: Število podatkovnih baz, trojic, RDF povezav na kategorijo.

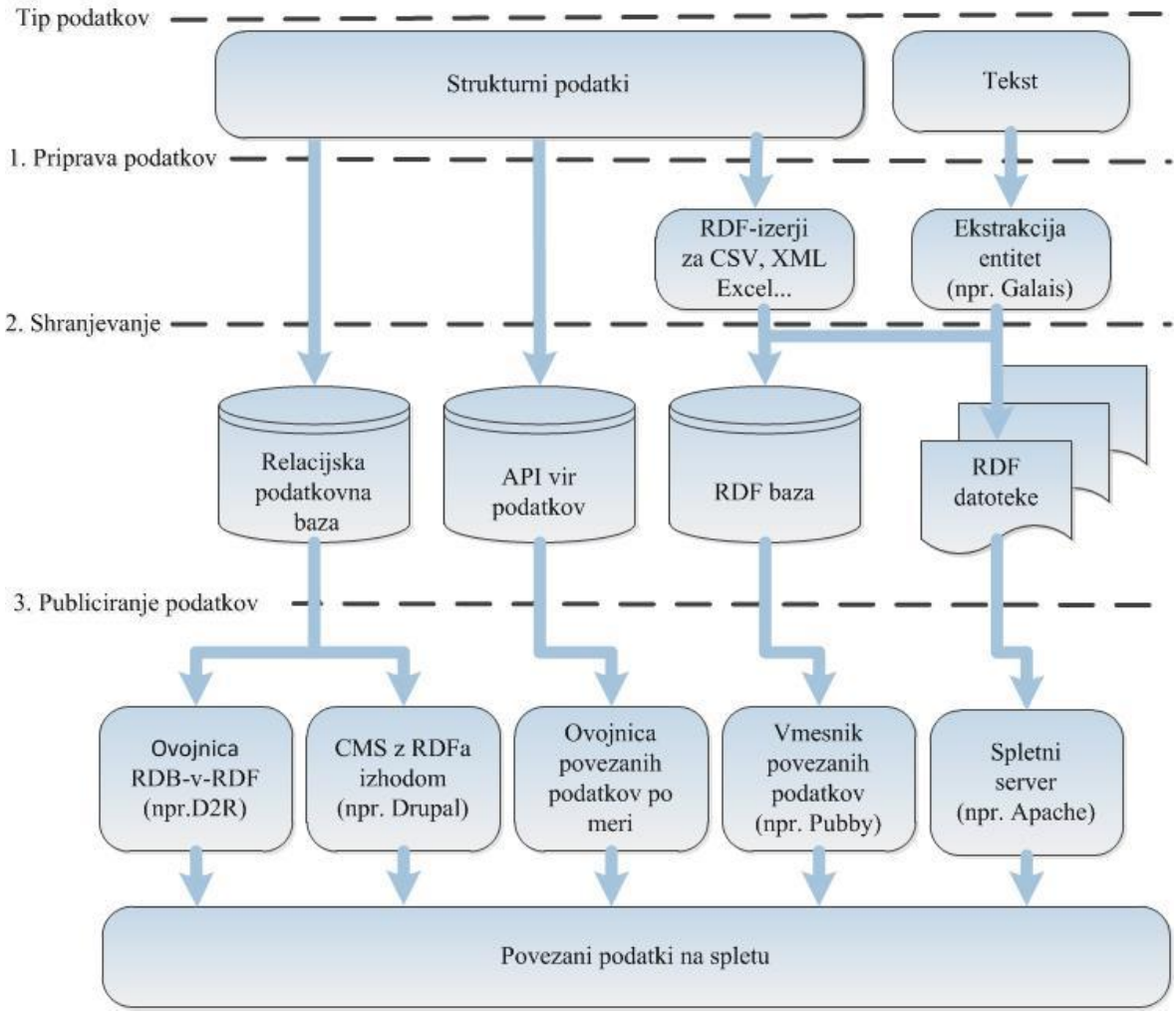
Kategorija	Podatkovne baze	Število trojic	Delež v %	Število RDF povezav	Delež v %
Medkategorija	41	4.184.635.715	13,23	63.183.065	12,54
Geografija	31	6.145.532.484	19,43	35.812.328	7,11
Vladni podatki	49	13.315.009.400	42,09	19.343.519	3,84
Mediji	25	1.841.852.061	5,82	50.440.705	10,01
Knjižnice	87	2.950.720.693	9,33	139.925.218	27,76
Znanost o življenju	41	3.036.336.004	9,60	191.844.090	38,06
Uporabniške vsebine	20	134.127.413	0,42	3.449.143	0,68
Skupaj	295	31.634.213.770		503.998.829	

[13] Spletni aslov: http://linkedgeodata.org/About

3.3 Objava LOD zbirk

Za uspešno objavo LOD zbirk velja določen postopek objave, morajo biti izpolnjene osnovne zahteve, ki smo jih našteli v uvodu poglavja 3. Pri pretvorbi obstoječih zbirk v objavljene LOD zbirke ni nujno, da je, zaradi združljivosti z zahtevami potrebno opustiti trenutni poslovni sistem in sistem za upravljanje podatkov. Potrebno je le dodati tehnični sloj, ki obstoječe sisteme poveže v LOD zbirke. Ker obstaja veliko različnih sistemov, katere je moč povezati v splet podatkov, obstaja tudi množica načinov oz postopkov objava zbirk [13].

Spodnja slika prikazuje možne procese objave LOD zbirk.



Slika 20: Proces objave LOD zbirk

Za določitev primernega postopka pri objavi zbirk si je potrebno postaviti tri vprašanja.

V kakšnem formatu imamo trenutno podatke?

Koliko podatkov moramo objaviti?

Kako pogosto se podatki spreminjajo?

Postopki za objavo so odvisni od odgovorov na zgornja vprašanja in jih bomo obravnavali v sledečih podsekcijah [9].

3.3.1 Tip vhodnih podatkov

Poglavitno pri izbiri postopka objave LOD zbirk je format v katerem imamo trenutno podatke na voljo. Podatkovne zbirke so lahko v različnih formatih, kot so relacijske podatkovne baze, tekstovni dokumenti in ostali formati, npr. CSV, XML itd. V sledečih sekcijah so obravnavana orodja in postopki za pretvorbo naštetih formatov [9].

3.3.1.1 Relacijske podatkovne baze

Podatkovne zbirke, ki so v obliki relacijskih podatkovnih baz je mogoče relativno enostavno objavit s pomočjo adapterjev "relacijska podatkovna baza v RDF" (ang. relational database to RDF wrappers). Takšna orodja preslikujejo relacijske baze v RDF grafe, ki izpolnjujejo principe povezanih podatkov. V takšnih primerih lahko uporabimo npr. orodje D2R server[14], ki omogoča tako spletnim kot RDF brskalnikom povpraševanja v SPARQL povpraševalnem jeziku.

3.3.1.2 Tekstovni dokumenti

V to množico spadajo vsi dokumenti, ki so napisani v naravnem jeziku, npr. članki novic, raziskav itd.. V takšnih primerih uporabimo tako imenovane entitetne ekstraktorje, kot so Calais[15], Ontos[16] ali DBpedia Spotlight[17].

Našteti delujejo tako, da v tekstovnem dokumentu poiščejo entitete in jim dodelijo pripadajoče URI naslove definiranih entitet, kar v spletu povezanih podatkov zagotavlja izmenjavo in črpanje dodatnih informacij iz širšega nabora virov.

3.3.1.3 Ostali formati

Med ostale formate spadajo datoteke tipa CSV, Excel, XML. V primerih naštetih formatov je potrebno podatke pretvoriti v statične RDF zbirke. Orodja, ki nam pri tem služijo, so RDF pretvorniki, ki smo jih obravnavali v sekciji 3.1.1.1. Navadno takšna orodja ponujajo tudi možnost direktne objave podatkov[18].

[14] Spletni naslov: http://d2rq.org/d2r-server
[15] Spletni naslov: http://www.opencalais.com/
[16] Spletni naslov: http://www.ontos.com/o_eng/index.php?cs=1
[17] Več na: http://wiki.dbpedia.org/spotlight
[18] Več na: http://www.w3.org/wiki/ConverterToRdf in http://simile.mit.edu/wiki/RDFizers

3.3.2 Količina podatkov

Količina podatkov ima velik vpliv na primernost procesa objave. Če gre za zbirko, ki vsebuje samo nekaj sto RDF trojic o določeni entiteti je smiselno podatke objaviti v obliki statičnih RDF datotek, kar sicer zahteva več rokovanja pri upravljanju podatkov, predvsem. Če imamo podatke v več formatih, vendar so takšne tehnične ureditve manj kompleksne in s tem cenejše. V primerih, ko imamo podatkovne zbirke z več entitetami je realizacija povezane zbirke v eni sami RDF datoteki odsvetovana. Velike RDF datoteke, ki vsebujejo več entitet so nesmiselne, saj je potrebno za vsako entiteto prenesti celotno datoteko in jo v celoti procesirati. Podatkovne zbirke z več entitetami, deli v več statičnih RDF datotek ali se naloži v tako imenovane Triplestore okolja. To so podatkovne baze, ki so razvite z namenom shranjevanja trojic, istočasno podpirajo SPARQL poizvedbe. Med takšne baze spadajo npr. 4store, D2R Server, SparkleDB in še množica ostalih [2].

3.3.3 Dinamičnost podatkov

Postopek objave zbirk je močno odvisen od pogostosti spreminjanja podatkov. Podatki, ki so zgodovinskega značaja se redko oz. se ne spreminjajo, zato je smiselna objava takšnih podatkov v statičnem RDF formatu. Podatki, ki se pogosto spreminjajo npr. trenutne statistike populacije pa v Triplestore bazah [13, 9].

4 APLIKACIJE LOD

Aplikacije povezanih podatkov zahtevajo množico specifičnih tehnik, tehnologij in principov. Za lažje razumevanje uporabe povezanih podatkov si bomo v sekciji aplikacije LOD ogledali aplikacije, ki so se pojavile v spletu podatkov. Ker je splet podatkov še v svojih začetnih stopnjah razvoja je na razpolago množica LOD aplikacij, ki so v prototipni obliki. Tem bodo sledile močno spremenjene in dodelane aplikacije. Vseeno pa nam takšne aplikacije omogočajo pregled in razumevanje povezanih podatkov in njihovega potenciala.

Aplikacije povezanih podatkov lahko na osnovi njihovega topološkega[19] namena delimo v dve skupini, splošne in specifične aplikacije [13].

4.1 Splošne aplikacije

Splošne aplikacije so namenjene procesiranju vseh topoloških kategorij, torej imamo eno aplikacijo, ki zna rokovati s podatki različnih topoloških pripadnosti. Med sploške aplikacije spadata dva bistvena predstavnika, brskalniki in iskalniki.

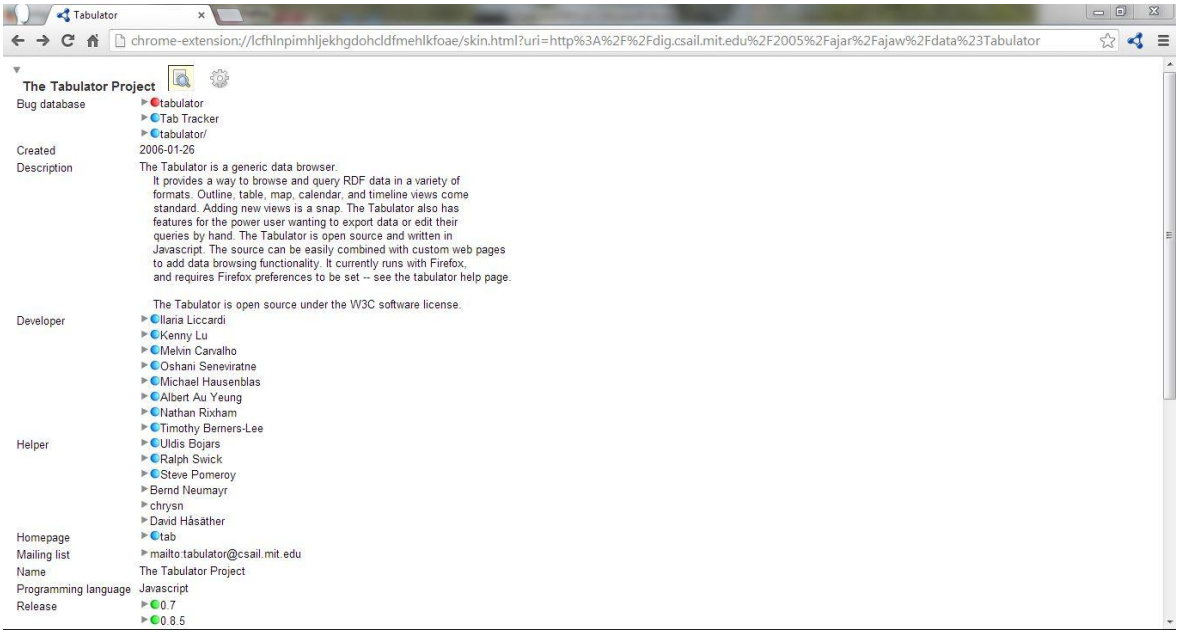
V sledeči sekciji si bomo ogledali nekaj brskalnikov, med katerimi sta pomembnejša Tabulator in MARBLE. V sekciji iskalniki si bomo podrobneje ogledali možnosti, ki jih ponuja iskalnik Falcons.

4.1.1 Brskalniki

Z uporabo brskalnika povezanih podatkov je možno dostopati do vseh virov podatkov, med katerimi obstaja RDF povezava. Na razpolago imamo množico brskalnikov povezanih podatkov

Tabulator je verjetno najbolj poznan brskalnik oz. razširitev brskalnikov Firefox, Camino in Chrome. Nastal je z namenom, da bi uporabnikom predstavil arhitekturo semantičnega spleta in pomagal pri testiranju baz, kar bi pokazalo možne izboljšave pri iskanju zakonitosti med podatki. Tabulator deluje tako, da zbere podatke iz različnih virov o istem objektu. Kot primer poizvedbe lahko vzamemo kar primer "tabulator", na naslovu "http://dig.csail.mit.edu/2005/ajar/ajaw/data#Tabulator"

[19] Obravnavano v sekciji 3.3.1.



Slika 21: Prikaz poizvedbe v Tabulatorju

Rezultat poizvedbe si lahko razložimo v obliki trojic na sledeč način. Subjekt "The Tabulator Project" je prikazan v prvi vrstici. Levi stolpec prikazuje predikate, desni pa objekte. Kot opisano v sekciji 2.2.1 so objekti lahko literali, prazna vozlišča in IRI identifikatorji. V primerih, ko je vozlišče tipa IRI, torej povezava na drugi vir je možno klikniti na puščico, ki nam pokaže rezultat poizvedbe tega vozlišča, kot prikazuje spodnja slika.

Slika 22: Prikaz podatkov subjekta v Tabulatorju

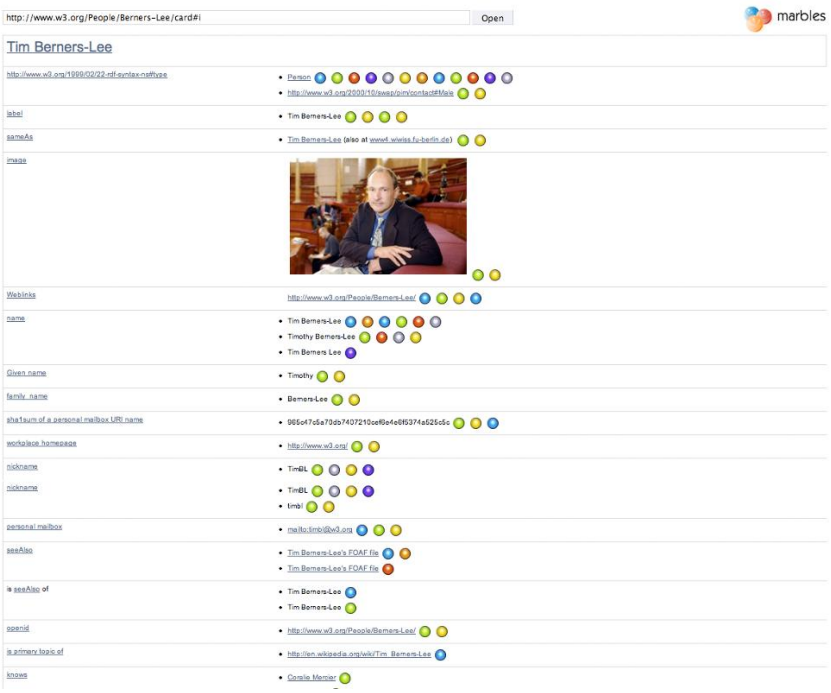
Na zgornji sliki lahko opazimo barvne oznake pred povezavami, vsaka barva ima svoj pomen.

Modra oznaka pomeni, da obstaja povezava iz vozlišča in je zato možno ustvariti poizvedbo o objektu dalje s pritiskom na puščico.

Zelena oznaka pomeni, da obstaja povezava iz vozlišča in so vrednosti poizvedbe že prenešene, rezultati so vidni s pritiskom na puščico.

Rdeča oznaka pomeni, da obstaja povezava iz vozlišča vendar povezava do vira ni mogoča (Napaka).

Zelo zanimiv brskalnik je *MARBLE*, ki tako kot *Tabulator* razišče vir podatkov. Podatke o elementu združi iz različnih virov, vsakeha pa označi s svojo barvo, kar omogoča enostavnejšo sledljivost podatkov [7].



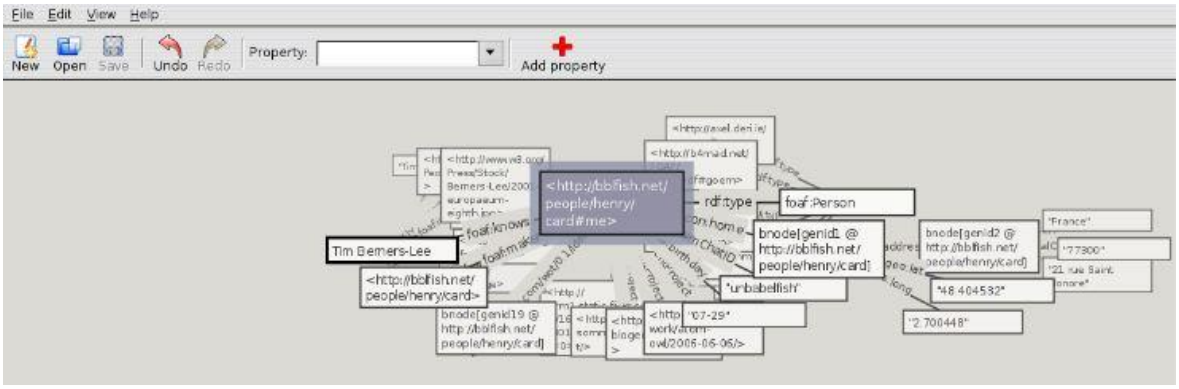
Slika 23: Primer prikaza Marble brskalnika[20]

Prav tako je zanimiv brskalnik *DBpedia Mobile*, brskalnik povezanih podatkov za mobilne naprave, ki s pomočjo GPS lociranja ponudi množico podatkov iz podatkovne baze DBpedije, ki se navezuje na trenutno lokacijo.

Podoben namen ima *OpenLink Data Explorer* krajše ODE, ki je dodatek za brskalnike Firefox, Safari, Chrome, Opera in Internet Explorer.

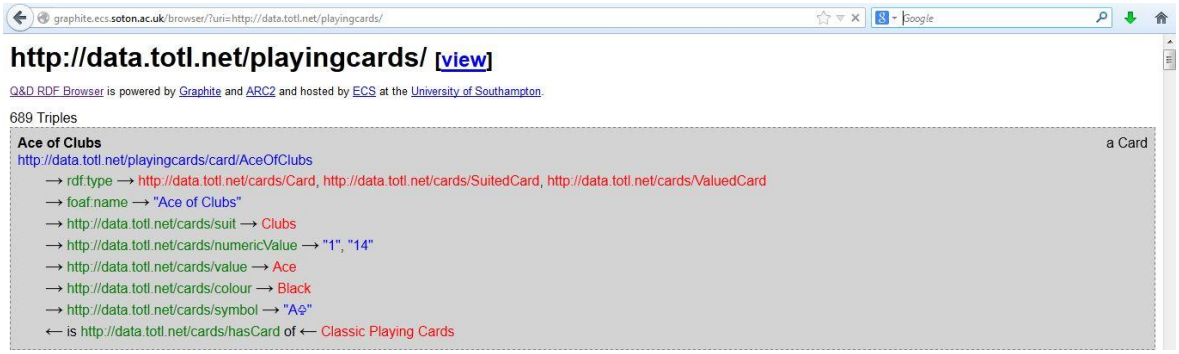
Brskalnik *Fenfire* nam prikaže rezultate v obliki grafa, prek katerega je mogoče potovati po celotni podatkovni bazi.

[20] Vir: http://linkeddatabook.com/editions/1.0/images/MarblesTimBL_small.png



Slika 24: Fenfire prikaz rezultatov[21]

Poznamo tudi brskalnik Ali Quick & Dirty RDF Browser, ki nam pokaže zelo enostavno RDF strukturo in je primeren za razvijalce semantičnega spleta in ustvarjalce RDF baz.

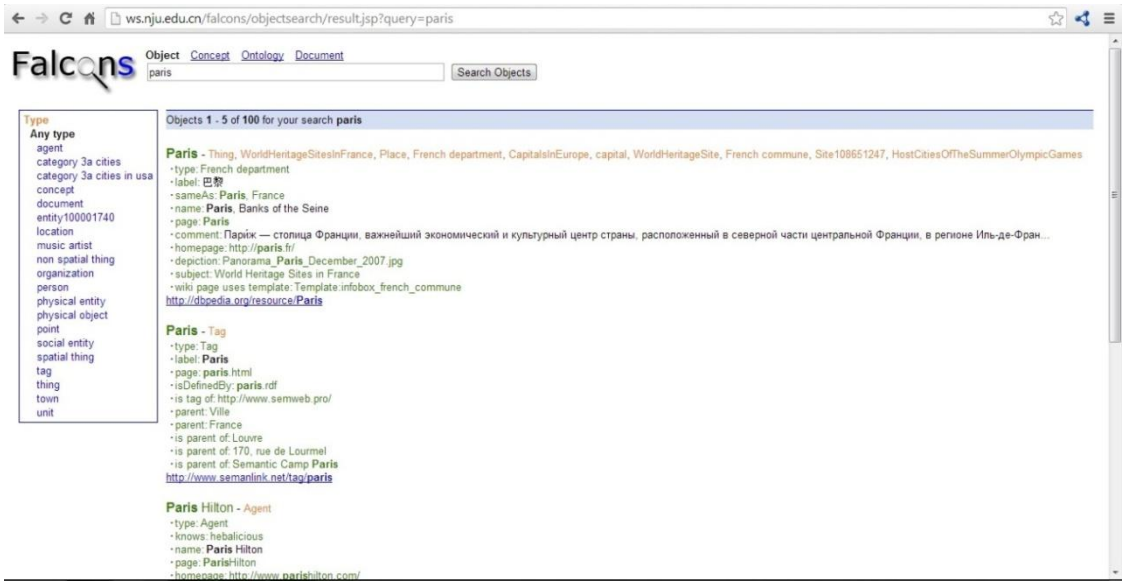


Slika 25: Prikaz rezultata v brskalniku Ali Quick & Dirty RDF Browser

4.1.2 Iskalniki

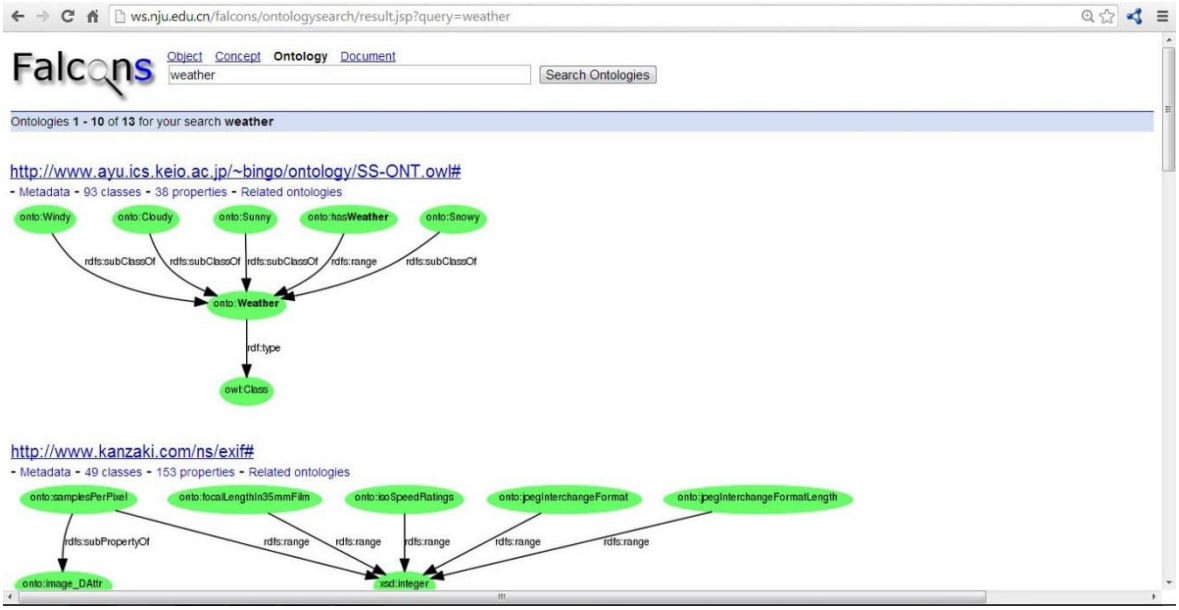
Poleg brskalnikov imamo iskalnike povezanih podatkov (ang. search engines), ki vključujejo na tisoče podatkovnih baz. Uporabniku omogočajo iskanje podatkov na osnovi ključne besede, kakor to omogočata iskalnika Google in Yahoo, vendar obstaja med iskalnikom povezanih podatkov in spletnimi iskalniki bistvena razlika. Iskalnik povezanih podatkov išče med podatki, spletni iskalnik, kot je Google pa išče po vsebinah dokumentov. Npr. če v Google vpišemo Paris nam iskalnik med rezultati ponudi dokumente, ki vsebujejo Paris kot mesto in Paris kot žensko osebo ali dišavo. V iskalniku povezanih podatkov Falcons pa lahko izberemo razred iskanega objekta npr. mesto in s tem močno omejimo nabor rezultatov. Poleg iskalnika Falcons sta poznana še Sig.ma in SWSE, vendar zadnji ni več aktiven.

[21] Vir: http://events.linkeddata.org/ldow2008/papers/14-hastrup-cyganiak-browsing-with-fenfire.pdf



Slika 26: Primer iskanja v Falcons iskalniku, z možnostjo izbire razreda iskanja

Poleg iskanja objektov nam Falcons omogoča tudi iskanje konceptov, ontologije in iskanje nizov po sami vsebini RDF dokumentov. Pri iskanju konceptov, je možno iskanje po določeni lastnosti objekta ali celo izbira uporabljenega slovarja. Zelo zanimiva opcija je iskanje ontologije, za katero je primer prikazan spodaj.



Slika 27: Primer iskanja ontologije v Falcons iskalniku

4.2 Specifične aplikacije

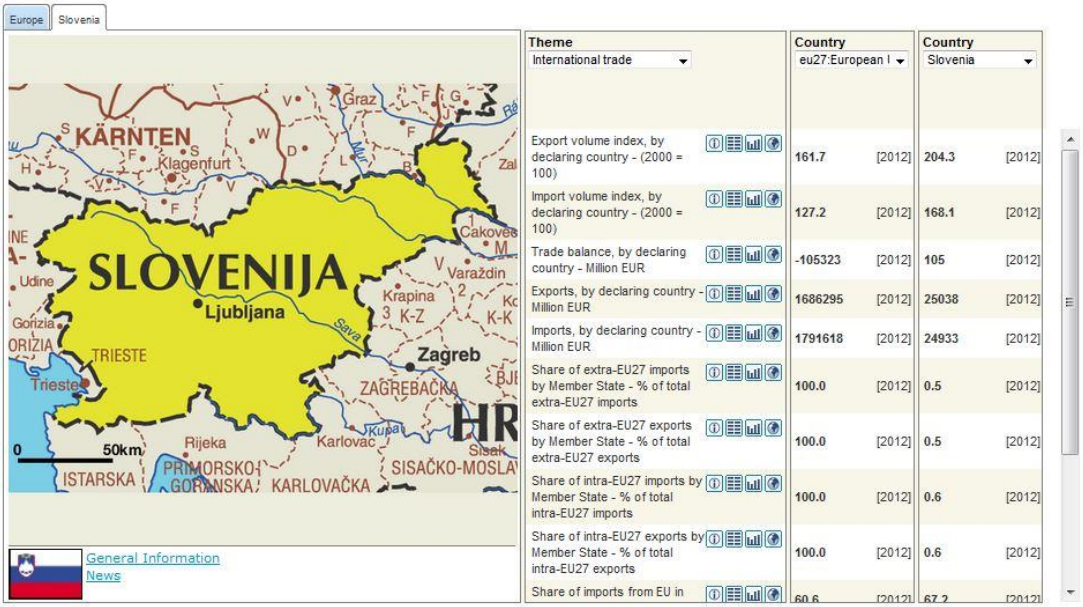
V skupino specifičnih aplikacij spadajo vse aplikacije, ki so razvite za procesiranje specifične topološke kategorije podatkov. Pri tem je potrebno opozoriti, da specifične aplikacije ne uporabljajo zgolj eno kategorijo podatkov, za informacijsko obogatitev rezultatov lahko uporabljajo tudi druge kategorije.

V sekciji si bomo ogledali aplikacijo EUCP, ki procesira Eurostat zbirk, aplikacijo Talis Aspire, je namenjena upravljanju podatkov izobraževalnega gradiva in aplikacijo Faviki, ki omogoča družbeno označevanje strani. Podrobneje si bomo ogledali delovanje aplikacije BBC Music, ki s pomočjo povezanih podatkov ponuja informacijsko obogatitev izdelkov producentske korporacije BBC.

Dodatno si bomo ogledali aplikacijo Facebook Graph API in korake, ki so bili potrebni, da aplikacija streže podatke tudi v RDF formatu.

4.2.1 EUCP

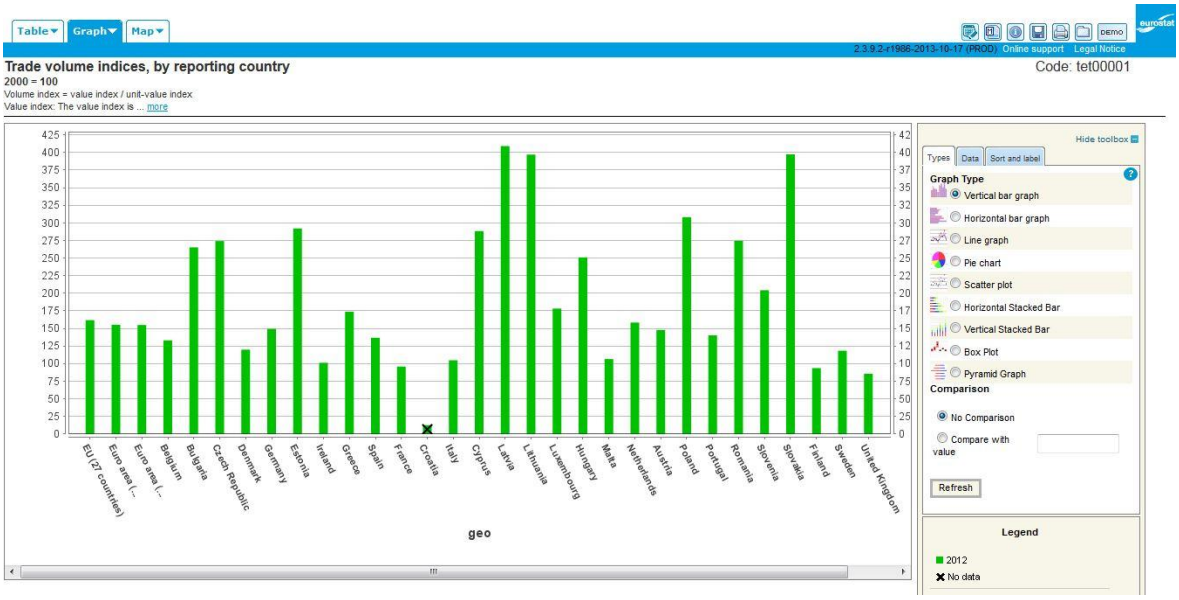
Najpomembnejši projekti specifičnih aplikacij, ki procesirajo povezane podatke so vladni projekti, ki z aplikacijami povezanih podatkov zagotavljajo vse večjo transparentnost vlad. Če za primer vzamemo aplikacijo *The Eurostat Country Profiles*[22]. Ta aplikacija z enostavno prikaže, zbrane Eurostat podatke vseh držav Evropske unije. The Eurostat County Profiles omogoča izbiro več področij podatkov in izbiro dveh območij za primerjavo. Območja so lahko države ali povprečja držav Evropske unije.



Slika 28: Primer prikaza rezultatov aplikacije The Eurostat Company Profiles

[22] Spletni naslov: http://epp.eurostat.ec.europa.eu/guip/introAction.do?profile=cpro&theme=eurind&lang=en

Vsako področje statistik ima več sekcij, kot prikazuje zgornji primer. Vsaka sekcija ima več možnosti prikaza, prikaz dodatnih informacij oz razlage kaj je obravnavano v dotični sekciji, prikaz rezultatov v tabeli, prikaz rezultatov v množici različnih vrst grafov in prikaz rezultatov na zemljevidu. Vse rezultate je moč natisniti in shraniti v datotekah različnih formatov.



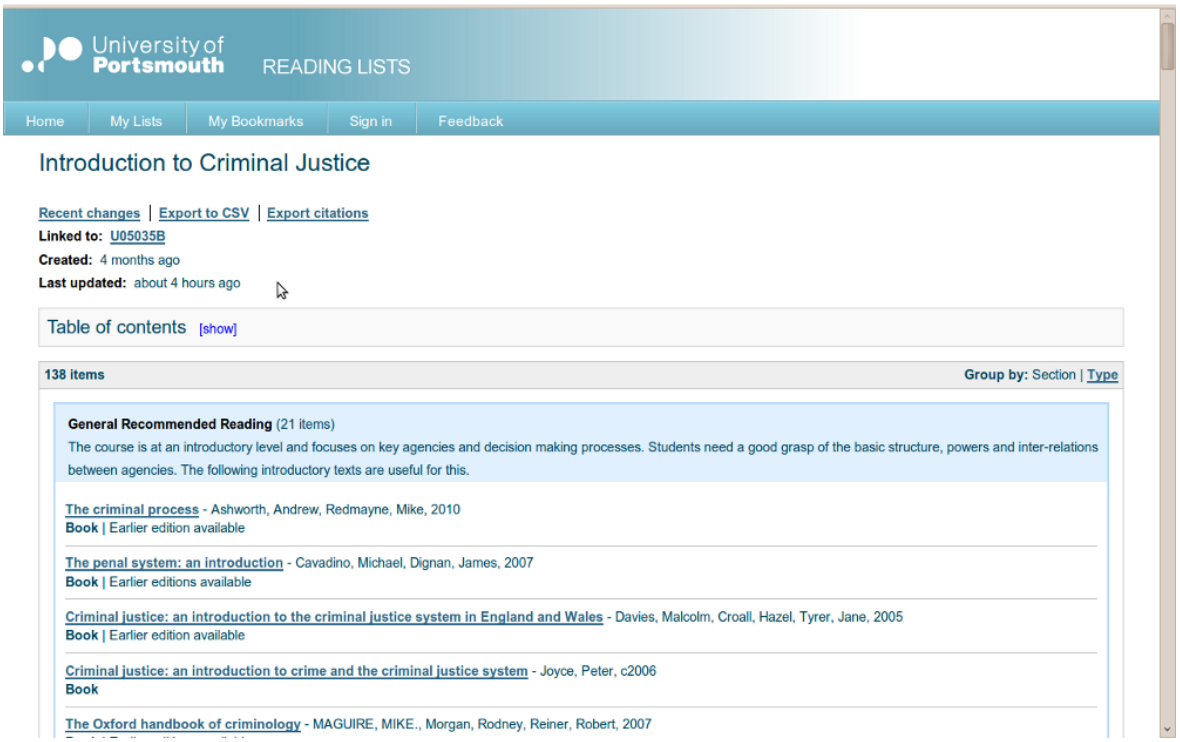
Slika 29: Prikaz rezultatov aplikacije Eurostat Company Profiles v obliki grafa

Seznam aplikacij in držav, ki si z aplikacijami povezanih podatkov želijo večjo transparentnost je na voljo na *data.gov*[23]. Za nas pomembnejši so podatki na portalu *European Union Data Portal*[24].

4.2.2 Talis Aspire

Med specifične aplikacije spada aplikacija *Talis Aspire,* ki je namenjena ustvarjanju in upravljanju virov izobraževalnega gradiva, kot so knjige, članki ali spletne strani. Talis Aspire uporablja na tisoče študentov iz številnih fakultet. Napisana je v PHP jeziku, na Talis platformi za povezane podatke, podatki pa so shranjeni v RDF formatu. Vsak vir, avtor, izdajatelj ima dodeljen URI naslov, kar omogoča globalno povezovanje podatkov in s pomočjo ostalih tehnologij, pridobivanje večjega nabora informacij v izobraževanju. Interakcija je zato omogočena prek spletnega vmesnika.

[23] Spletni naslov: http://www.data.gov/opendatasites
[24] Spletni naslov: https://open-data.europa.eu/en/data



Slika 30: Spletni prikaz poizvedbe v Talis Aspire[25]

4.2.3 Faviki

Faviki[26] je orodje, ki omogoča družbeno označevanje strani. Vsak, ki se prijavi lahko izbrani spletni strani doda semantično značko. Pri dodajanju značke spletni strani vpišemo značko, ki bi najbolje opisala vsebino strani. Faviki nam nato ponudi možne semantike izraza, npr. če vpišem "jaguar" nam Faviki ponudi značke Jaguar kot znamka avtomobilov ali jaguar kot žival. Pri tem izberemo semantični pomen značke, ki najbolje opisuje vsebino strani. Značko je možno dodati tudi dereferencabilnem URI naslovu, s katerim je mogoče pridobiti dodatne podatke. Po vseh zaznamkih, ki jih skupnost ustvari je moč iskati z značkami. Npr. če izberemo značko Jaguar kot avtomobil so rezultati iskanja vse strani, katerih vsebina je povezana z avtomobili Jaguar. Faviki predstavlja ključni del semantičnega spleta in semantičnega iskanja.

4.2.4 BBC Music

Michael Hausenblas je v članku *Exploiting Linked Data For Build Web Application*[27] podal zelo lep primer uporabe povezanih podatkov in povezave med tehnologijami RDF/XML, SPARQL in uporabo slovarjev. BBC Music je spletna stran, ki z uporabo

[25] Vir: http://linkeddatabook.com/editions/1.0/images/TalisAspirePortsmouth_small.png
[26] Spletni naslov: http://www.faviki.com/pages/welcome/
[27] Spletni vir: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.216.8004&rep=rep1&type=pdf

zgoraj naštetih tehnologij o glasbenikih omogoča pridobivanje množico različnih informacij. Čeprav je stran v HTML jeziku je mogoče s pomočjo curl orodja pridobiti RDF obliko pogleda na podatke. Primer poizvedbe je prikazan spodaj

```
curl -H "Accept: application/rdf+xml"
http://www.bbc.co.uk/music/artists/79239441-bfd5-4981-a70c-55c3f15c1287
```

Odgovor iz strani serverja

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
xmlns:foaf = "http://xmlns.com/foaf/0.1/"
xmlns:mo = "http://purl.org/ontology/mo/"
xmlns:owl = "http://www.w3.org/2002/07/owl#">
<mo:SoloMusicArtist rdf:about="/music/artists/79239441-bfd5-4981-a70c-
55c3f15c1287#artist">
<foaf:name>Madonna</foaf:name>
<owl:sameAs rdf:resource="http://dbpedia.org/resource/Madonna_(singer)"
/>
<mo:image rdf:resource="/music/images/artists/7col_in/79239441-bfd5-
4981-a70c-55c3f15c1287.jpg" />
<foaf:page rdf:resource="/music/artists/79239441-bfd5-4981-a70c-
55c3f15c1287.html" />
<mo:musicbrainz rdf:resource="http://musicbrainz.org/artist/79239441-
bfd5-4981-a70c-55c3f15c1287.html" />
<mo:homepage rdf:resource="http://www.madonna.com/" />
<mo:fanpage rdf:resource="http://www.mad-eyes.net" />
<mo:fanpage rdf:resource="http://www.allaboutmadonna.com/" />
...
</mo:SoloMusicArtist>
</rdf:RDF>
```

V zgornjem odgovoru je razvidno, da so uporabljeni slovarji FOAF, ki opisuje ljudi, MO, ki opisuje glasbenike in OWL, ki opisuje osnovne lastnosti znotraj RDF struktur. FOAF npr. določa lastnost name oz. ime neke osebe, MO pa npr. fanpage oz. spletno stran oboževalcev glasbenika.

Za še lažje razumevanje povezav med podatki si lahko ogledamo primer poizvedbe SPARQL nad zgornjimi podatki.

```
PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?fanpage
FROM <http://triplr.org/rdf/www.bbc.co.uk/music/artists/79239441-bfd5-
4981-a70c-55c3f15c1287>
WHERE {
?artist a mo:SoloMusicArtist ;
owl:sameAs <http://dbpedia.org/resource/Madonna_(singer)> ;
mo:fanpage ?fanpage .
}
```

Torej do spletne strani oboževalcev pridemo tako, da določimo glasbenika in lastnost, ki nas zanima in je opisana v slovarju.

4.2.5 Facebook Graph API

Velik korak pri razvoju aplikacij povezanih podatkov, ki so namenjene široki množici je naredil Facebook. Facebook Graph API je aplikacija, ki omogoča vsakomur prebiranje in objavo podatkov iz Facebook oblaka. Prvotno je Facebook Graph API ponujal podatke v JSON formatu, leta 2011 pa so se usmerili tudi v svet semantičnega spleta, s ponujanjem enakih podatkov v RDF formatu. Facebook ima čez milijardo medseboj povezanih uporabnikov, ki imajo osebne podatke, fotografije, priljubljene dogodke in še veliko več, zato si težko predstavljamo kako ogromen je Facebookov oblak povezanih podatkov in kako velik prispevek prinašajo spletu podatkov.

V sledečih vrsticah si bomo ogledali kako je potekala pretvorba aplikacije Graph API iz klasičnega podatkovnega formata JSON v Turtle format semantičnega spleta, kateri so bili izzivi, omejitve in kateri so dosežki.

Projekt je imel tri bistvene zahteve, čim manjše spremembe v kodi aplikacije, zagotoviti delovanje, ker bi bile potrebe po človeškem poseganju čim manjše in izogib XML formata. Našteto je bilo doseženo z implementacijo robustne in prilagodljive tranzicije JSON izhoda v Turtle izhod, ki uporablja mehanizma HTTP pogajanja.

Prvi korak v projektu je bila pretvorba iz JSON format v RDF format. JSON format podatkov v praksi sestavljata dva elementa, ključ in vrednost. Tipi podatkov pa so lahko števila, nizi, booleanove vrednosti, polja, ničelne vrednosti in objekti, ki so množice ključev in vrednosti. Ker je bilo v projektu določeno, da se ne sme uporabiti XML jezika je tudi RDF/XML neprimeren format za ta primer. Odločili so se za alternativni format Turtle. Pretvorba se je vršila relativno enostavno. Objektom v JSON formatu so bili dodeljeni URI identifikatorji, ti so predstavljali subjekte v trojicah. Ključem v objektu so bili dodeljeni URI in predstavljajo predikat. RDF objekt pa predstavljajo vrednosti, katere je bilo potrebno smiselno pretvoriti v literale, URI identifikatorje ali prazna vozlišča.

Pri dodeljevanju URI identifikatorjev je bilo potrebno nekaj pazljivosti in skrbi, saj so morali biti identifikatorji dereferencabilni skladni z načeli httpRange-14[28], ki jih določa tip podatka na katerega kaže URI identifikator na osnovi odgovora na poizvedbo po identifikatorju. Natančneje če "http" vir odgovori z 2xx na poizvedbo GET, identifikator predstavlja neko informacijo, če odgovori z 303 identifikator lahko predstavlja karkoli, odgovor 4xx pomeni, da je tip neznan.

Vsaka instanca v grafu ima unikatni identifikator, ki je lahko v obliki pozitivnega celega števila (identifikatorji enostavnega tipa) ali niz števil s podčrtaji vmes (sestavljeni identifikatorji). Prvim določimo URI identifikator tako, da enostavno identifikatorju v

[28] Več na: http://www.garshol.priv.no/blog/125.html

grafu dodamo fragmentni identifikator[29] v obliki #. Vsaka instanca v grafu z indetifikatorjem enostavnega tipa je v RDF obliki dosegljiva na URI naslovu /identifikator#. Pri določanju URI identifikatorjev za instance sestavljenih identifikatorjev je zadeva nekoliko bolj kompleksna. Če bi dodelili URI naslov v obliki /identifikator# bi z dereferenciranjem dobili v JSON odgovor "false" s kodo 200 OK. Ker se lahko v aplikaciji odpravlja samo minimalne popravke je bila najboljša rešitev uvedba URI značk (ang. Tag URI). Instanci s sestavljenim identifikatorjem i je tako dodeljen URI oblike tag:graph.facebook.com,2011:/id_i, kjer je graph.facebook.com naslov za dostop do povezanih podatkov. Obstajajo posebni primeri JSON objektov, ki ne predstavljajo instance, v takšnih primerih jih pri tranziciji v RDF pretvorimo v prazna vozlišča.

Vsaka instanca v družbenem grafu ima določene tipe, ki opisujejo lastnosti. Pri tranziciji iz JSON formata v RDF je velik izziv ontološko vrednotenje lastnosti. Prvi korak pri ontološkem vrednotenju lastnosti je objava ontoloških shem. Za vsak tip instance je bila objavljena shema, ki je dostopna na naslovu /schem/class, kjer class predstavlja tip instance. Tip instance je pri JSON objektih opredeljen s ključem "type". Med tipi lahko najdemo razrede user, page, album, photo, status itd. Tip "c" je možno identificirati z URI identifikatorjem c:type, lastnost "p" pa z identifikatorjem c:p. Primer je prikazan spodaj.

```
user:type a rdfs:Class;
        rdfs:label "user".
```

Če je pri pretvorbi objekta iz JSON v RDF format mogoče najti tip instance, se v shemi preveri za vsak ključ, če zanj obstaja semantična razlaga. V takšnem primeru se določi URI za specifično lastnost. V primerih, ko JSON objekt nima ključa, ki bi opredelil tip instance se pri pretvorbi lastnostim določi generični tip identificiran z URI oblike :p, kjer p predstavlja ključ lastnosti. Primer generične lastnosti.

```
:name a rdf:Property;
        rdfs:label "name";
        rdfs:comment "A tag having no ..."
```

Lastnosti generičnega tipa imajo enako semantično vrednoto kot JSON ključ. Iz tega sledi, da je specificirana lastnost dejansko podlastnost generične lastnosti z enakim imenom, kar je prikazano na spodnjem primeru.

```
user:name a rdf:Property;
        rdfs:label "name";
        rdfs:comment "The usewr's full..";
        rdfs:domain user:type;
        rdfs:subPropertyOf :name.
```

[29] Več na: http://en.wikipedia.org/wiki/Fragment_identifier

V skupini generičnih lastnostih imamo še dve vrsti posebnih lastnosti. Prva je lastnost, s ključem "key", ki je pri tranziciji v RDF, za razliko z ostalimi generičnimi lastnostmi, semantično obogatena. Primer je prikazan spodaj.

```
:id a owl:InverseFunctionalProperty;
       rdfs:label "name"
       rdfs:comment "The user's full ..."
       rdfs:range xsd:string.
```

V drugo skupino spadajo ključi nenegativnih celih števil, za katere je v RDFS uporabljeno indeksiranje. Primer je prikazan spodaj.

```
:_0 a rdf:Property;
       rdfs:label "_0";
       rdfs:comment "A tag having no ...";
       rdfs:subPropertyOf api:has;
       api:index 0.
```

V zgornjem primeru je treba vedeti, da api:has ustreza rdfs:member in lastnost api:index eksplicitno določa numerično indeksiranje lastnosti.

S tem smo zaključili z opisom objave povezanih podatkov za aplikacijo Graph API.

Aplikacija dinamično ustvarja povezane podatke s pretvorbo JSON izhodov v Turtle format, ki je semantično obogaten z uporabo RDF shem in OWL slovarjev. URI dereferenciranje pa je ustvarjeno v skladu z načeli povezanih podatkov. Gaph API ponuja izjemno veliko količino podatkov, zato je njegova povezava v splet podatkov izredno pomemben prispevek [20].

4.3 Izdelava LOD aplikacij

Pri razvoju aplikacij se moramo usmerjati po principih povezanih podatkov iz poglavja 3, vendar principi ne določajo kako dosežemo aplikacijo, ki učinkovito izkorišča povezane podatke, ampak zato potrebujemo dodatne razlage in postopke, ki jih bomo obravnavali v sledečih vrsticah.

Vprašanje, ki si ga lahko postavimo je sledeče. Kako bomo razvili aplikacijo, ki bo v najboljši meri izkoristila prednost povezanih podatkov. Lahko si predstavljamo, da imamo spletno stran ali blog na naslovu *http://example.org/cw/*, ki obravnava tematiko hladne vojne. Zdaj, ko smo ugotovili kaj so povezani podatki, si želimo vsebino naše strani obogatiti z uporabo takšne tehnologije. Da bi dosegli zastavljen cilj moramo narediti dva

osnovna koraka, pripraviti naše podatke in izbrati ciljne podatkovne baze s katerimi bomo obogatili našo spletno stran.

4.3.1 Priprava naših podatkov

V skladu s povezanimi podatki lahko začnemo tako, da ustvarimo URI identifikatorje. Npr. če imamo za uporabljen spletni format naslovni prostor *http://example.org/cw/resource/conflict* ustvarimo za RDF format naslov *http://example.org/cw/rdf/conflict* in za html format *http://example.org/cw/html/conflict*.[30]

Naslednji korak je izbira primernih slovarjev. V sekciji 3.1. smo že obravnavali izbiro slovarjev in pomen uporabe obstoječih slovarjev pri globalizaciji podatkov in njihovo uporabo v aplikacijah.

Za izbiro primernih slovarjev je najprej potrebno analizirati naše podatke oz. definirati tematike podatkov. To dosežemo s tematskim določanjem entitet in relacij med podatki. V primeru naše spletne strani, ki govori o hladni vojni je jasno, da imamo opravka s podatki, ki govorijo o ljudeh in dogodkih, ki so se zgodili na določeni lokaciji v določenem času. Tipična slovarja, ki prideta pri tem v poštev sta slovar FOAF, ki je slovar, ki definira ljudi in slovar Event Ontology, ki definira ontologijo za opis dogodkov.

Sledeč korak v našem projektu je določitev načina objave RDF podatkov. Imamo naslednje možne oblike: enostaven RDF/XML format, XHTML+RDFa, ki omogoča integracijo RDF grafov v XHTML datoteki ali SPARQL končno točko, vmesnik ki omogoča povpraševanje po RDF strukturah. Če želimo popolno programsko rešitev za objavo relacijskih podatkov imamo na voljo orodja, kot so D2R ali Triplifty, ki omogočata enostavno preslikavo podatkov v RDF format. Na drugi strani se v poslovne namene večinoma uporabljata platformi Virtuoso in Talis, katerih zagon zahteva nekoliko več znanja. V primerih, kot je naš, se dobro obnese uporaba ARC2, PHP knjižnice, ki zna rokovati z RDFjem in ga je možno uporabiti z xAMPP platformo, ki združuje komplet orodij za upravljanje spletnih strani, tudi CMS sistemov, kot so Joomla ali Worldpress.

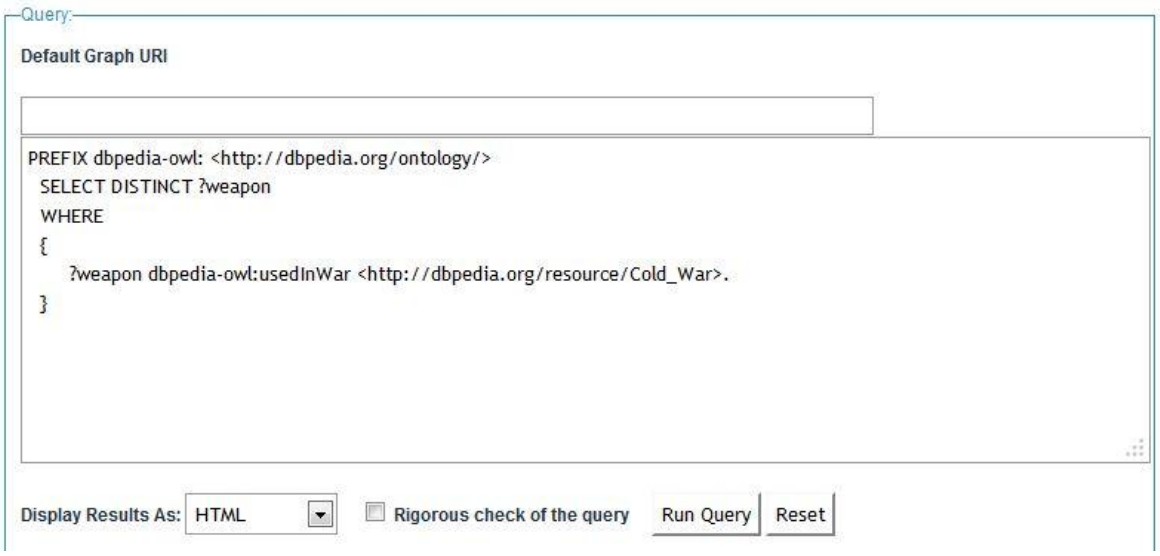
4.3.2 Izbira in uporaba baz povezanih podatkov

V trenutku, ko smo dosegli skladnost s spletom podatkov lahko nadaljujemo s selekcijo podatkovnih virov, ki bodo služili za obogatitev naše strani. Iskati in raziskati baze, ki bi nam lahko služile ni tako enostavno, saj zahteva nekaj intuitivnosti, dela in časa. V veliko pomoč so nam tako imenovani semantični indeksi, kot je npr. *Sindice*. Sindice zbira semantične vsebine iz spleta in nam omogoča vrsto različnih načinov pri raziskovanju

[30] Več o ustvarjanju URI identifikatorjev in izbiri slovarjev je mogoče prebrati v vodiču "How to Publish Linked Data on the Web" (http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/).

vsebine virov. Ponuja nam možnost splošnega iskanja z nizi, po lastnostih npr. lahko določimo lastnost foaf:mbox ali uporabimo množico filtrov. Dodatno Sindice omogoča SPARQL poizvedbe in SIGMA pregled vsebin virov.

Slika 21 prikazuje primer SPARQL poizvedbe za naš primer, iščemo vsa orožja, ki so bila uporabljena v hladni vojni.



Slika 31: Primer SPARQL poizvedbe v Sindice

Dodatna možnost, ki se je pojavila za lažje odkrivanje spleta podatkov, je tako imenovan voiD. VoiD prinaša razrede in lastnosti, ki nam omogočajo opis vsebine in medbaznih povezav. Lahko določimo tip in število povezav, naprimer lahko določimo, da je 120k povezav tipa foaf:depiction iz baze A v bazo B. Čeprav je tehnologija, kot je voiD zelo uporabna pri raziskovanju podatkov, je upravljalcev virov, ki se odloča za njeno uporabo malo, zato je takšen način raziskovanja nerazširjen.

Če se vrnemo k našem primeru je intuitivno, da bomo potrebovali vire splošnih informacij, kot je DBpedia in npr. Geonames za opis geografskih lokacij. Obe bazi sta del oblaka povezanih podatkov, kar pomeni, da s povezavo naše baze o hladni vojni postane z zgoraj omenjenima bazama tudi del povezanih podatkov.

Kot upravljalci podatkov lahko podatke izpostavimo z uporabo vmesnika XHTML+RDFa-ja ali SPARQL končne točke, na strani uporabnika podatkov je izbira najprimernejšega formata. HTML format je odgovor, če bodo podatki predstavljeni neposredno človeku. Če bo podatke procesiral agent je primeren format sam RDF.

Koraki, ki smo jih zgoraj omenili zadostujejo, če želimo našo aplikacijo obogatiti z uporabo obstoječih podatkov. Če bi želeli stran "hladna vojna" objaviti je to mogoče na široko uporabljeni platformi Drupal, ki omogoča module za integracijo LOD podatkov.

Če gre za komercialno aplikacijo imamo kot razvijalci še dodatne korake, moramo zagotoviti verodostojnost podatkov, zanesljivost delovanja, preverjati učinkovitost in zagotoviti dobro skalabilnost.

5 ZAKLJUČEK

Svetovni splet je omogočil globalni informacijski prostor povezanih dokumentov, ki so razumljivi samo človeku, računalnik je imel le nalogo posrednika, ki nam postreže dokumente v čitljivi obliki. Izziv, ki se je pri tem pojavil, je bil ustvariti splet v katerem računalnik ni samo posrednik dokumentov, ampak lahko dostopa do podatkov in jih razume, da bi lahko z njimi logično rokoval. Ta izziv imenujemo splet podatkov.

V zaključni nalogi smo si ogledali osnovne tehnologije spleta povezanih podatkov, kot so RDF model, URI identifikatorji, SPARQL povpraševalni jezik in slovarji. Ogledali smo si pomen LOD zbirk in razvoj odprtih povezanih podatkov skozi čas. Prehod iz spleta dokumentov v splet podatkov je odprl praznino v naboru uporabnih aplikacij, ne samo zato, ker aplikacije povezanih podatkov temeljijo na specifičnih tehnologijah, ampak ker povezani podatki omogočajo aplikacijam popolnoma nov spekter uporabe. Zato sem v zaključni nalogi predstavil nekaj aplikacij, ki nam pomagajo pri ustvarjanju povezanih podatkov, torej pri pripravi podatkov, njihovi objavi in njihovi uporabi. Naštel in predstavil sem nekaj najbolj razvitih in poznanih aplikacij, ki uporabljajo povezane podatke in nekaj priporočenih praks pri razvoju novih aplikacij, ki izkoriščajo prednosti povezanih podatkov.

Med raziskovanjem povezanih podatkov sem prišel do zaključka, da takšen splet ponuja izjemen potencial pri razvoju aplikacij, saj je možno videti nešteto novih konceptov in spektrov uporabe, vendar je potrebno vedeti, da je celoten koncept spleta povezanih podatkov ni še popolnoma razvit. Pojavlja se veliko ovir pri samem razvoju in uporabi aplikacij. Pri uporabi aplikacij sem večkrat zasledil, da podatki niso bili ažurni ali celo dostopni in mislim, da je to največja težava s povezanimi podatki, saj veliko upravljalcev, po objavi podatkov v obliki povezanih podatkov teh več ne uporablja oz. ureja, ampak uporabljajo popolnoma ločen sistem s klasičnimi podatkovnimi bazami. Pri razvoju aplikacije je potrebno veliko pozornosti posvetiti raziskovanju virov.

6 LITERATURA IN VIRI

[1] *Linked Open Data*, http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData (31.3.2014).

[2] *Triplestore*, http://en.wikipedia.org/wiki/Triplestore (31.3.2014).

[3] D. Beckett, B. McBride, *RDF/XML Syntax Specification*, http://www.w3.org/TR/REC-rdf-syntax/ (31.3.2014).

[4] T. Berners-Lee, *Linked Data*, http://www.w3.org/DesignIssues/LinkedData.html, (31.3.2014).

[5] T. Berners-Lee, D. Connolly, L. Kagal, Y. Scharf, J. Hendler, *N3Logic: A logical framework for the World Wide Web*, Theory and Practice of Logic Programming 8, (2008), 249−269.

[6] T. Berners-Lee, *Primer: Getting into RDF & Semantic Web using N3*, http://www.w3.org/2000/10/swap/Primer.html (31.3.2014).

[7] T. Berners-Lee, J. Hollenbach, K. Lu, J. Presbrey, E. Pru d'ommeaux, mc schraefel, *Tabulator Redux: Writing Into the Semantic Web* (2007)

[8] T. Berners-Lee, J. Hendler, and O. Lassila, The Semantic Web, Scientific American, (2001)

[9] C. Bizer, R. Cyganiak, T. Heath, *How to Publish Linked Data on the Web*, http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/LinkedDataTutorial/

[10] R. Cyganiak, C. Bizer, *Pubby a Linked Data Frontend for SPARQL Endpoints*, Freie Universität Berlin, http://wifo5-03.informatik.uni-mannheim.de/pubby/ (31.3.2014).

[11] R. Cyganiak, D. Wood, M. Lanthaler, *RDF 1.1 Concepts and Abstract Syntax*, http://www.w3.org/TR/rdf11-concepts/, (31.3.2014).

[12] M. Grobelnik, *Semantični splet - več kot le hype!*, Jožef Stefan Institute http://videolectures.net/kp07_grobelnik_spvkh/ (31.3.2014).

[13] T. Heath , C. Bizer, *Linked Data: Evolving the Web into a Global Data Space*, 2011, Synthesis Lectures on the Semantic Web: Theory and Technology, 1:1, 1−136. Morgan & Claypool

[14] F. Manola, E. Miller, B. McBride, *RDF 1.1 Primer*, http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140225/ (31.3.2014).

[15] M. Nilsson, A. Powell, P. Johnston, A. Naeve, *Expressing Dublin Core metadata using the Resource Description Framework (RDF)*, http://dublincore.org/documents/2006/05/29/dc-rdf/ (31.3.2014).

[16] E. Prud'hommeaux, A. Seaborne, *SPARQL Query Language for RDF*, http://www.w3.org/TR/rdf-sparql-query/ (31.3.2014).

[17] M. R. Koivunen, E. Miller, *W3C Semantic Web Activity*, Semantic Web Kick-Off in Finland, HIIT Publications, Helsinki (2002), 35−36.

[18] F. Sansalone , *Linked Data: End-User Applications?*, http://artofsystems.blogspot.com/2009/03/linked-data-end-user-applications.html (31.3.2014).

[19] I. Savnik, *RDF*, http://osebje.famnit.upr.si/~savnik/predmeti/PMJ/rdf.pdf (31.3.2014).

[20] J. Weaver, P. Tarjan, *Facebook Linked Data via the Graph API*, Semantic Web, Information Technology, Artificial Intelligence and Theory of Computation, IOS Press (2012), 245−250.