Izbor materijala temeljen na ontologiji povezanih podataka

Cundeković, Marko

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Marko Cundeković

Zagreb, 2014.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:
Prof. dr. sc. Mario Essert, dipl. ing.

Student:
Marko Cundeković

Zagreb, 2014.

Izjavljujem da sam ovaj rad izradio samostalno koristeći stečena znanja tijekom studiranja i navedenu literaturu.

Zahvaljujem se svom mentoru prof. dr. sc. Mariu Essertu na pruženoj stručnoj pomoći i savjetima pri izradi ovog diplomskog rada.

Također se zahvaljujem prof. dr. sc. Tomislavu Filetinu, dr. sc. Veri Rede i dr. sc. Ireni Žmak na pruženim savjetima i literaturi.

Zahvaljujem se i svojoj obitelji, svim kolegama i prijateljima koji su mi bili podrška tijekom studiranja.

Marko Cundeković

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

DIPLOMSKI ZADATAK

Student: **MARKO CUNDEKOVIĆ** Mat. br.: 0035157307

Naslov rada na hrvatskom jeziku: **Izbor materijala temeljen na ontologiji povezanih podataka**

Naslov rada na engleskom jeziku: **Material Selection Based on the Ontology of Linked Data**

Opis zadatka:

Izbor materijala jedno je od temeljnih strojarskih disciplina, jer ovisno o uspjehu tog izbora ovisi konstrukcija i uporaba proizvoda. Primjena računala u izboru materijala odavno je postala nezaobilazna točka na tom putu, a nove tehnologije svakim danom poboljšavaju mogućnosti računalne primjene. Jedna od njih je i mrežna (web) tehnologija povezanih podataka (LOD – linked open data) uz pomoć koje se izbor materijala provodi s daleko više upita (kriterija svojstava materijala) nego do sada. Uz to, za njegovo korištenje dovoljan je mrežni preglednik (browser), pa korisnici mogu biti na bilo kojoj strani svijeta.

U ovom radu potrebno je istražiti i provesti:

1. Osnove 'linked data' paradigme, te njenog mjesta u semantičkom webu,
2. Opisati vrste i značajke materijala za koje će se stvarati ontologijski model,
3. Opisati uloguVirtuoso mrežnog poslužitelja i njegove 'triplestore' baze LOD trojaca,
4. Implementirati ontologijski model u on-line rješenje,
5. Projektirati i izvesti mrežno sučelje preko kojeg će korisnik moći postavljati složene (SPARQL) upite, koji će kao rezultat davati numeričke i grafičke rezultate.

Zadatak zadan:	Rok predaje rada:	Predviđeni datum obrane:
25. rujna 2014.	27. studenog 2014.	3., 4. i 5. prosinca 2014.

Zadatak zadao: Predsjednik Povjerenstva:

Prof.dr.sc. Mario Essert

Prof. dr. sc. Franjo Cajner

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

SAŽETAK

Cilj ovog rada je izraditi i organizirati bazu podataka, te odgovarajuće web sučelje preko kojeg će biti moguće vršiti izbor inženjerskih materijala prema željenim kriterijima, odnosno pretraživati materijale prema njihovim vrstama i svojstvima, te rezultate vizualno prikazati grafom.

U prvome dijelu je predstavljen i analiziran teoretski dio semantičkog web-a, osnovni pojmovi i načini pretrage povezanih podataka, te alati za pohranu (OpenLink Virtuoso) i kreiranje (Protégé) povezanih podataka.

Drugi dio objašnjava praktični dio. Opisuje strukturu baze te klase i relacije kojima su prikazani tipovi materijala i njihova svojstva. Također se navodi lista upita u SPARQL-u koji su potrebni za pretraživanje te kako općenito koristiti web aplikaciju uz odgovarajuća objašnjenja.

Ključne riječi: povezani, podaci, semantički, web, sparql, rdf, materijal, izbor, vrsta, svojstva

SUMMARY

The goal of this work is to develop and organize a database, and corresponding web interface through which will be possible to search and make a selection of engineering materials according to desired criteria, material types and properties, and present the results with chart.

The first part presents the theoretical part and analysis of semantic web, basic concepts and methods, tool for storage (OpenLink Virtuoso) and tool for creation (Protégé) of linked data.

The second part is explanation of the practical part. It shows the structure of the database, classes and relations that represent the types and properties of materials and SPARQL queries that are used to search the same. It also shows how to use web application with the corresponding explanations.

Keywords: linked, data, semantic, web, sparql, rdf, material, selection, type, property

1. Uvod

1.1. Semantički web

Razvoju semantičkog web-a uvelike je doprinijelo širenje dosadašnjeg weba i povećanje količine informacija koje se na njemu nalaze. Dolazi do problema velikog ponavljanja istih informacija u kojima je teško pronaći onu pravu.

Nakon statičkog Web 1.0 i dinamičkog Web 2.0 načina funkcioniranja interneta, može se reći da je semantički web ustvari Web 3.0 koji kao glavnu značajku ima povezivanje informacija semantičkim vezama, odnosno davanje značenja vezama koje povezuju podatke. U dosadašnjem web-u su podaci povezani preko HTML linkova koji predstavljaju samo pokazatelj na mjesto gdje se drugi podatak, informacija, video, audio ili tekst nalazi. Na taj način iste informacije zapisane na različitim internet stranicama ne mogu biti prepoznate kao iste od strane računalnih algoritama, a ponekad ni od samih korisnika.

Semantički web uvodi način, koristeći postojeću infrastrukturu weba, na koji točno definira određen pojam iz stvarnog svijeta. Pri tome omogućuje opisivanje tog istog pojma na više internet stranica koje zajedno čine globalnu mrežu informacija koje opisuju taj pojam. Više nije potrebno da svaka internet stranica posjeduje sve informacije o pojedinom pojmu zbog čega i dolazi do ponavljanja, već cijela mreža funkcionira kao jedna globalna baza podataka i zbog toga doprinosi puno širim informacijama o određenom pojmu. [1]

Tako na primjer, semantički web omogućuje spajanje informacija o materijalima koji se koriste u proizvodnji sa više izvora. Više nije potrebno da svaka stranica na kojoj se spominje pojedini materijal ima sve podatke o istom. Semantički web omogućuje kreiranje oznake za pojedini materijal koja ga jednoznačno definira, te sve stranice na kojima se on spominje povezuju svoje podatke o njemu preko njegove indetifikacije.

Na taj način sam proizvođač pojedinog materijala može ga povezati sa cijenom i njegovim svojstvima, kao i laboratorij za testiranje materijala koji ga može povezati sa svojim rezultatima testiranja, odnosno njegovim svojstvima. Razne tvrtke koje izrađuju proizvode od njega ga mogu povezati sa svojim proizvodima. Kemičari ga mogu povezati sa kemijskim elementima sadržanim u njemu. Geografija ga može povezati sa zastupljenosti po područjima planete. Za svaki pojam postoji ogroman broj područja u kojem se oni detaljno analiziraju na različite načine, a upravo semantički web objedinjuje sve informacije o nekom pojmu te na taj način proširuje i pretvara Internet u globalnu bazu znanja.

1.2. Izbor materijala

Izbor materijala dolazi u ranoj fazi izrade proizvoda već pri samom procesu projektiranja i konstruiranja proizvoda gdje se određuje i do 70% troškova izrade proizvoda. Danas je konstruktorima na raspolaganju više stotina tisuća inženjerskih materijala razvrstanih u 4 osnovne klase: metali, polimeri, keramike i kompoziti. Zbog tako velikog broja postojećih i velikog broja novih materijala koji se pojavljuju, inženjeri moraju biti u mogućnosti brzo doći do kvalitetnih podataka, samim time povećati kvalitetnu proizvoda, smanjiti cijenu i na taj način ostvariti prednost pred konkurencijom. [3]

Izbor materijala u grubo se može podijeliti u 4 osnovna koraka koje su naveli J.R. Dixon i C. Poli [4].

- Na temelju kritičnih svojstava, odrediti da li će dio biti izrađen od metala, polimera, keramike ili kompozita.
- Odrediti da li će metalni dio biti izrađen deformacijom ili lijevanjem; za polimere odrediti njegovu vrstu.
- Suziti kategorije materijala. Metali mogu biti podijeljeni na ugljične čelike, nehrđajuće čelike, legure… Polimeri mogu biti podijeljeni u plastomere, duromere…
- Odabrati odgovarajući materijal prema odabranom razredu i zahtjevima proizvoda.

Materijali se izabiru pri izradi novih proizvoda, ali i pri poboljšanju postojećih. Kod izrade novih proizvoda konstruktor ima najveću slobodu pri izboru i sva svojstva koja proizvod zahtjeva se detaljno analiziraju. Kod prilagođavanja ili izradi varijanti postojećih dozvoljena je promjena oblika, raspodjela dijelova, dimenzija itd., ali ne osnovnih funkcionalnih rješenja. Razlozi zbog kojih dolazi do ponovnog izbora materijala su: [3]

- Razmatranje prednosti uvođenja novog materijala.
- Poboljšanje uporabnih karakteristika proizvoda.
- Promjena uvjeta rada u uporabi proizvoda.
- Kvarovi u uporabi uzrokovani karakteristikama materijala.
- Pojava novih uputa, propisa, normi i zakona.
- Sniženje troškova i postizanje bolje konkurentnosti.

Postoji više načina na koji se mogu izabirati materijali. Najduži i najnespretniji način za konstruktora je traženje i uspoređivanje podataka o materijalima u tiskanim izdanjima kao što su katalozi proizvođača, priručnici, knjige i časopisi. Veliki nedostatak je što se ne može pronaći velik broj materijala na jednom mjestu kako bi ih se međusobno usporedilo, već je obično pojedino izdanje vezano uz pojedinu vrstu materijala ili manji broj više vrsta materijala.

Sljedeći način je pretraga kompjutorskih baza podataka koje se nekada isporučuju na CD-u, a razvojem interneta dolazi i do razvoja web aplikacija koje u pozadini sadrže takve baze podataka pristupačne globalno. U ovom slučaju je moguće pristupiti većem broju materijala odjednom, te usporediti njihove karakteristike na brži i efikasniji način, uspoređujući više željenih svojstava materijala sa onima u bazi odjednom.

Postoje i CAMS (Computer Aided Materials Selection) sustavi koji također u pozadini sadrže svojstva materijala zapisanih u bazu podataka, ali i bazu znanja koja pomaže pri odlučivanju pri odabiru materijala. Pojedini sustavi imaju mogućnost direktnog prebacivanja karakteristika materijala u CAD/CAM sustav, te na taj način omogućavaju brzu simulaciju i analizu proizvoda.

Kao najrazvijeniji računalni sustavi za izbor materijala su ekspertni sustavi. Oni mogu zamijeniti stručnjake, do određene granice, pri izboru materijala koristeći logiku odlučivanja uz bazu podataka i bazu znanja. U novijim verzijama se počinje primjenjivati i umjetna inteligencija za potpuno odlučivanje umjesto ljudi.

Kod svih načina koje koriste bazu podataka kao izvor informacija o materijalima, od kritične je važnosti u startu kreirati bazu na način da je ona proširiva novim materijalima i omogućiti nadopunjavanje i prepravljanje postojećih.

Zbog toga je potrebno iza svake kvalitetne baze podataka uvijek imati osobu, odnosno administratora baze podataka, koji je zaslužan za kreiranje, održavanje, kontroliranje i nadopunjavanje baze podataka novim materijalima i podacima. [3], [5]

2. Povezani podaci

2.1. Osnovni elementi povezanih podataka

2.1.1. Resurs, URI (Uniform Resource Identifier) i namespace

Osnovni element povezanih podataka čini resurs. Resurs predstavlja bilo koji pojam iz stvarnog svijeta koji se može identificirati i imenovati. Osim pojmova koji se žele povezati u semantičkom webu, pod resursima se podrazumijevaju i same relacije, odnosno veze između tih pojmova. [1], [2]

Resursi semantičkog weba vezani za materijale mogu biti:

- sam pojam *materijal*
- kemijski elementi: *željezo, aluminij, titan…*
- grupe materijala: *metal, nemetal, čelik, aluminijeva legura…*
- individualni materijali poput aluminijevih legura: *AlMg3, AlSi12…*
- svojstva materijala: *vlačna čvrstoća, gustoća, temperatura taljenja…*
- mjerne jedinice: *Pascal, Newton, metar, centimetar…*

Svaki resurs se jednoznačno određuje dodjeljivanjem URI-a (Uniform Resource Identifier). Na taj način se razlikuju resursi sa istim imenom, ali različitim značenjem poput homonima. Pascal je na primjer mjerna jedinica, ali i osoba.

URI je vrlo sličan URL-u (Uniform Resource Locator) svojim izgledom i strukturom. Za razliku od URL-a koji vodi do određene lokacije na internetu bilo to audio, video, tekstualne datoteke ili web stranice, URI može, ali i ne mora iza sebe imati sadržaj. Iako je poželjno da se odlaskom na određen URI prikaže internet stranica sa objašnjenjem i prikazom veza sa resursom, URI i bez toga ima funkciju. Njegova osnovna funkcija je točno definiranje resursa o kojem se radi, za razliku od URL-a koji je bez sadržaja u pozadini beskoristan link bez funkcije.

Upravo zbog lakšeg opisa URI-a i mogućnosti dodavanja internet stranica koje stoje iza njih, pri izradi samih URI-a valja pripaziti da oni sadržavaju samo znakove engleske abecede, brojeve i eventualno (donju) crtu kao zamjenu za prazno mjesto. Na taj način se osigurava da URI može u bilo kojem trenutku postati URL ako to već nije i da se samim otvaranjem URI-a kao internet adrese dolazi do prikaza povezanih podataka o željenom resursu. Zbog tog razloga se u nastavku ovoga rada koristi URI na engleskom jeziku. [1], [2]

Sljedeći primjer prikazuje nekoliko URI-a resursa materijala, svojstava materijala i mjernih jedinica:

http://www.materialsdb.eu/resource#AluminiumAlloy
http://www.materialsdb.eu/resource#AlMg3
http://www.materialsdb.eu/property#hasTensileStrength
http://www.materialsdb.eu/property#hasMeltingPoint
http://www.materialsdb.eu/unit#Pascal
http://www.materialsdb.eu/unit#Kelvin

Kako bi se u zapisivanju i pretraživanju URI-a u ovome radu i bazama podataka općenito izbjeglo ponavljanje domene pomoću koje je definiran URI, uvode se *prefix*, *namespace*, *qname* i *local name* koji pomažu u kompaktnosti i preglednosti zapisa Uri-a, te smanjenju veličine baze podataka i datoteka.

Dio URI-a koji se ponavlja naziva se *namespace* i on obično predstavlja domenu na kojoj se nalazi skup sličnih resursa. Ponavljanje domene, odnosno *namespacea*, u zapisu svakog URI-a se zamjenjuje proizvoljno definiranim prefiksom, tj. skraćenicom.

Tako se dio URI-a *http://www.materialsdb.eu/resource#* može zamijeniti sa prefiksom *res, http://www.materialsdb.eu/property#* se može zamijeniti prefiksom *pro,* a *http://www.materialssdb.eu/unit# s*a prefiksom *unit.* Prefiks se piše odvojen dvotočkom od imena resursa. Ime resursa, odnosno *local name* u navedenom primjeru predstavljaju *AluminiumAlloy, AlMg3, hasTensileStrength, hasMeltingPoint, Pascal* i *Kelvin*. Prefiks i ime resursa odvojeni dvotočkom zajedno čine *qname*.

Gore navedeni URI-i se mogu zapisati na sljedeći način pomoću *qname-a*:

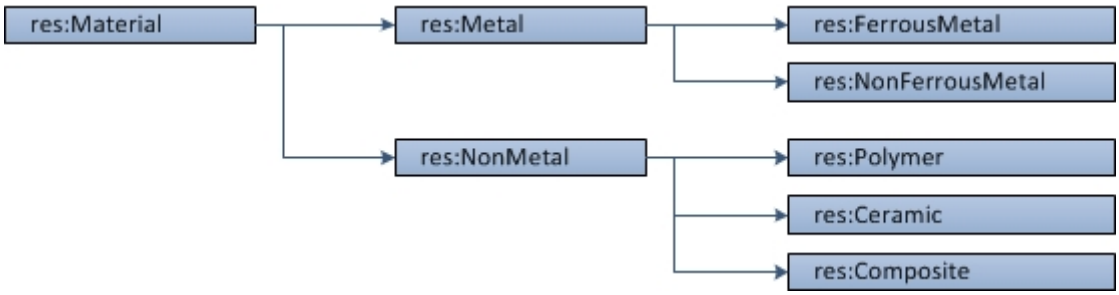
res:AluminiumAlloy
res:AlMg3
pro:hasTensileStrength
pro:hasMeltingPoint
unit:Pascal
unit:Kelvin

2.1.2. Klase, relacije i individue

Klase u modelu povezanih podataka predstavljaju grupe sličnih resursa, odnosno individua. Klase mogu sadržavati druge klase, odnosno podklase koje predstavljaju detaljnije grupe resursa.

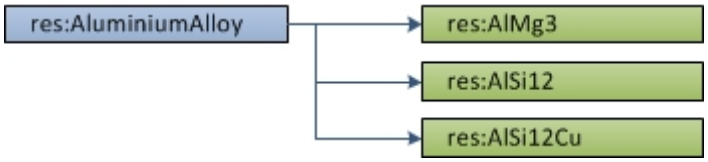
Za jednostavan primjer se može uzeti klasa *res:Material* koja predstavlja najvišu klasu. Ona može sadržavati sve ostale podklase, odnosno podgrupe materijala. U njoj tako mogu biti sadržane klase *res:Metal* i *res:NonMetal*. Nadalje, klasa *res:Metal* može sadržavati podklase *res:FerrousMetal* i *res:NonFerrousMetal*. Klasa *res:NonMetal* može sadržavati podklase *res:Polymer*, *res:Ceramic* i *res:Composite*. Na taj se način može stvoriti cijela hijerarhija grupa i podgrupa materijala do željene i potrebne razine.

Slika 1. ilustrira primjer hijerarhije vrsta i podvrsta, odnosno klasa i podklasa materijala.



Slika 1. - Klase i podklase materijala

U klasama materijala smještene su individue, odnosno krajnji resursi, a to su u ovom slučaju konkretni materijali definirani jednom od standardnih oznaka i sa poznatim karakteristikama. Tako će u podklasi *res:AluminiumAlloy* biti smještene individue, odnosno vrste aluminijevih legura: *res:AlMg3, res:AlSi12, res:AlSi12Cu...* Slika 2. ilustrira primjer tri navedene individue u odgovarajućoj klasi.



Slika 2. - Materijali kao individue u odgovarajućoj klasi

Na isti način se resursi mjernih jedinica i svojstva materijala mogu grupirati i razvrstati u klase. Tako najviša klasa mjernih jedinica *unit:Unit* može sadržavati podklasu *unit:Pressure* koja nadalje može sadržavati individue *unit:Pascal, unit:Bar…* U klasi svojstava materijala osnovnu klasu može predstavljati klasa *pro:hasMaterialProperty* i sastojati se od podklasa *pro:hasMechanicalProperty, pro:hasThermalProperty, pro:hasElectricalProperty…* Nadalje klasa *pro:hasMechanicalProperty* može sadržavati *pro:hasTensileStrength, pro:hasYieldStrength, pro:hasYoungsModulus…*

2.1.3. RDF (Resource Description Framework)

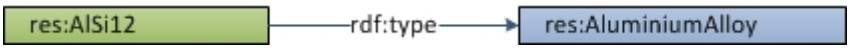
RDF je idejni podatkovni model semantičkog weba koji nije striktno vezan sa određenim formatom zapisa, već koristi jednu od vrsta serijalizacije. On služi za povezivanje i širenje mreže podataka pomoću svog osnovnog elementa RDF trojca. Osnovni elementi RDF trojca su subjekt, relacija i objekt. [1], [2]

Subjekt i objekt predstavljaju dva resursa ili resurs i doslovnu vrijednost, te su povezani relacijom. Ta veza za razliku od relacijskih baza podataka ne predstavlja samo općenitu povezanost, već daje i semantiku RDF trojcu, odnosno opisuje vrstu veze. Postoje relacije koji vežu klase i individue međusobno i one se nazivaju O*bject properties*. Relacije koji vežu klase i individue sa doslovnim vrijednostima zovu se *Datatype properties*. Slika 3. ilustrira općeniti model RDF trojca.

Slika 3. - Model RDF trojca

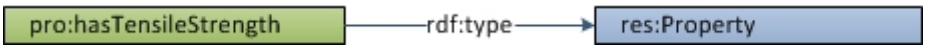
Namespace osnovnog RDF vokabulara je *http://www.w3.org/1999/02/22-rdf-syntax-ns#* i sastoji se od nekoliko osnovnih klasa i relacija za stvaranje mreže povezanih podataka. RDF *namespace* se uobičajeno zamjenjuje sa prefiksom *rdf.*

Individue se dodjeljuju u klase RDF modela pomoću relacije *rdf:type.* Slika 4. ilustrira RDF trojac koji dodjeljuje individuu *res:AlSi12* u klasu *res:AluminiumAlloy.*



Slika 4. - Dodjela individue u odgovarajuću klasu

Pomoću relacije *rdf:type* je moguće definirati pojedini resurs kao relaciju, odnosno dodati ga u RDF klasu *rdf:Property*. Iako zbog samog povezivanja to nije obavezno, definiranje relacija se preporuča kako bi im se mogli dodijeliti podaci o podacima, te kako bi se korisnicima i algoritmima olakšala pretraga RDF trojaca i baza podataka. Slika 5. ilustrira RDF trojac koji deklarira resurs kao relaciju.



Slika 5. - Definiranje resursa kao relacije

Na mjestu objekta, umjesto URI-a resursa može stajati i doslovna vrijednost. Slika 6. ilustrira RDF trojac koji veže *res:AlSi12* preko relacije *pro:hasTensileStrength* sa doslovnom vrijednosti *"150 MPa - 230 MPa".*

Slika 6. - Povezivanje resursa sa doslovnom vrijednosti

Doslovnim vrijednostima kojima nije definirana vrsta nazivaju se *plain literal*. Kod takvih vrijednosti algoritam za pretraživanje RDF trojaca pretpostavlja tip vrijednosti, no može doći do grešaka. Da bi se izbjegli problemi kod operacija nad tim vrijednostima, a ujedno i olakšalo pretraživanje, moguće je definirati vrstu doslovne vrijednosti. Tada se ona naziva *typed literal*. Vrijednosti je moguće definirati preko standardnih koje su obuhvaćene u *XML Schema Datatype* ili kreirati vlastite.

Globalni *namespace* XML Scheme je *http://www.w3.org/2001/XMLSchema#*, a obično se zamjenjuje prefiksom *xsd*. Na taj način algoritam može provjeriti da li je moguće vršiti računske operacije, operacije nad znakovima, vremenom, datumom… Slika 7. ilustrira RDF trojac sa definiranom doslovnom vrijednosti kao cijeli broj, odnosno *integer*.

Slika 7. - Definiranje vrste doslovne vrijednosti

Doslovnim tekstualnim vrijednostima je moguće definirati jezik na kojem su napisane. Na taj način je moguće spremiti višejezične nazive i opise pojedinog resursa, odnosno pojma koji se nalazi iza određenog URI-a. Slika 8. ilustrira RDF trojce sa zajedničkim subjektom *res:Iron*, i nazivom resursa na više jezika.

Slika 8. - Definiranje doslovne vrijednosti na više jezika

Na mjestu objekta u RDF trojcu mogu stajati i prazni čvorovi odnosno *blank nodes*. Ako u RDF trojcu postoji više objekata sa istom relacijom, koji uz to moraju biti grupirani tada se uvode prazni čvorovi. Oni nemaju trajno značenje već je njihova jedina funkcija povezivanje više vrijednosti sa zajedničkim subjektom i/ili relacijom. Slika 9. ilustrira jedan takav slučaj gdje aluminijeva legura *res:AlSi12* sadrži više vrijednosti vlačne čvrstoće (minimalnu i maksimalnu) i pri tome se svaka vrijednost sastoji od doslovne vrijednosti i individue, odnosno iznosa i mjerne jedinice.

Slika 9. - Povezivanje resursa praznim čvorovima

2.1.4. Serijalizacija RDF trojaca

Pod terminom serijalizacija se podrazumijeva pohranjivanje RDF trojaca u običnu tekstualnu datoteku pomoću neke od standardiziranih sintaksi, kako bi ih računala mogla pročitati. Prvi i najjednostavniji način pohrane RDF trojaca u tekstualne datoteke, odnosno način serijalizacije, je N-Triples. Kod njega se subjekt, relacija i objekt pišu u istoj liniji i nije dozvoljeno pisanje jednog RDF trojca u više linija.

Na temelju N-Triples načina serijalizacije, nastao je sličan, ali prošireni oblik imena Turtle (Terse RDF Triple Language). Ovaj način će biti objašnjen i koristiti će se u daljnjim primjerima ovoga rada zbog svog kompaktnog i razumljivog načina zapisa.

Za razliku od N-Triples načina zapisa, prazna mjesta i novi redovi ne utječu na zapis RDF trojaca u Turtle sintaksi. U njoj je pravilo da se potpuni URI uvijek piše unutar izlomljenih zagrada, bilo za deklaraciju prefiksa ili u samim trojcima.

Prefiksi se deklariraju na samom početku datoteke. Osnovna sintaksa deklaracije prefiksa je pisanje ključne riječi *@prefix* nakon koje se dodjeljuje ime prefiksa odvojeno dvotočkom od *namespace-a* koji se piše u izlomljenim zagradama, te se na kraju deklaracije pojedinog prefiksa piše točka. [1], [2]

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns# > .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix mat: <http://www.materialsdb.eu/resource#> .
@prefix pro: <http://www.materialsdb.eu/property#> .
@prefix unit: <http://www.materialsdb.eu/unit#> .
```

Postoji i osnovni prefiks, odnosno *base prefix*. On se obično koristi za najčešće korišteni prefiks jer kod njegova pozivanja nije potrebno upisivati njegovo ime, već samo dvotočku što doprinosi još većoj kompaktnosti zapisa. On se definira ključnom riječi *@base* nakon koje ne dolazi ime već samo *namespace* u izlomljenim zagradama i točka na kraju. Osnovni prefiks se može definirati i sa ključnom riječi *@prefix*, a u tom slučaju se upisuje samo dvotočka, bez imena, te *namespace* u izlomljenim zagradama i točka na kraju.

```
@base <http://www.materialsdb.eu/resource#> .
@prefix : <http://www.materialsdb.eu/resource#> .
```

U ovom slučaju za resurs *res:AlSi12* više nije potrebno pisati prefiks već je dovoljno pisati *:AlSi12.*

Ako se želi označiti jezik na kojem je doslovna vrijednost napisana tada nakon same vrijednosti u navodnim znakovima dolazi znak @ i oznaka jezika prema ISO 639 standardu poput *hr*, *en*, *fr*…

```
"željezo"@hr
"iron"@en
"fer"@fr
```

Doslovne vrijednosti u Turtle serijalizaciji se pišu unutar navodnih znakova. Ova sintaksa dopušta i definiranje tipa doslovne vrijednosti prema *XML Schema Datatype*, upisujući nakon same vrijednosti znakove ^^ i tip vrijednosti bilo skraćenim imenom preko unaprijed definiranog prefiksa ili pomoću punog URI-a.

```
"2650"^^xsd:integer
"0.002"^^<http://www.w3.org/2001/XMLSchema#decimal>
"željezo"^^<http://www.w3.org/2001/XMLSchema#string>
"2014-06-10T00:19:45"^^xsd:dateTime
"true"^^xsd:boolean
```

Kompaktan način zapisa RDF trojaca u Turtle serijalizaciji se postiže na način da se elementi RDF trojca koji se ponavljaju ne moraju pisati svaki put, već se njihovi različiti elementi nižu odvojeni zarezom ili točkom-zarez jedni od drugih.

U slučaju zapisa više RDF trojaca koji imaju zajednički subjekt, a različite relacije i objekte, njihove relacije i objekti se nižu odvojeni točkom-zarez, a nakon zadnjega se piše točka.

```
:AlSi12     rdfs:label                  "AlSi12"xsd:string ;
            pro:hasDensity              "2650"^^xsd:integer ;
            pro:hasMeltingTemperature   "580.12"^^xsd:decimal .
```

U slučaju ponavljanja subjekta i relacije, a različitih objekata RDF trojaca, objekti se nižu i odvajaju zarezom.

```
:Iron       rdfs:label      "željezo"@hr ,
                            "iron"@en ,
                            "fer"@fr .
```

Na ove načine je zapisano tri različita RDF trojca.

Prazni čvorovi se u Turtle sintaksi pišu pomoću donje crte umjesto prefiksa, te nakon dvotočke privremenim proizvoljnim imenom, ali ih je moguće i kompaktnije zapisati tako što se zamijene uglatim zagradama unutar kojih se pišu relacije i objekti sljedećeg RDF trojca.

Sljedeći primjeri prikazuju zapis istih RDF trojaca na dva navedena načina.

```
:AlSi12     pro:hasTensileStrength     _:ts1 .
_:ts1       rdf:value                  "2650"^^xsd:integer .
_:ts1       unit:hasUnit               unit:megaPascal .


:AlSi12     pro:hasTensileStrength
            [ rdf:value     "2650"^^xsd:integer ;    unit:hasUnit    unit:megaPascal  ]
```

2.2. Ontologije

2.2.1. RDFS (Resource Description Framework Schema)

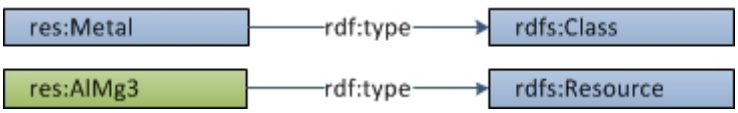
RDFS je prva semantička nadogradnja na idejni RDF model semantičkog weba. Ona omogućava detaljnije opisivanje, strukturiranje i povezivanje hijerarhije klasa i relacija samog RDF modela i njegovih osnovnih klasa i relacija. Globalni *namespace* RDFS rječnika je *http://www.w3.org/2000/01/rdf-schema#,* a uobičajeni prefiks koji se dodjeljuje je *rdfs.* [2]

Na sličan način kao što se pomoću RDF-a resurs definira kao relacija, preko relacije *rdf:type* i klase *rdf:Property*, pomoću RDFS se definiraju klase i sami resursi. Za to služe klase *rdfs:Class* i *rdfs:Resource.*

Na sljedećem primjeru je resurs *res:Metal* definiran kao klasa, a *res:AlMg3* kao resurs.

```
mat:Metal     rdf:type     rdfs:Class .
mat:AlMg3     rdf:type     rdfs:Resource .
```

Slika 10. ilustrira definiranje klase i resursa.

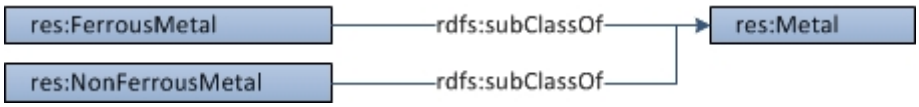


Slika 10. - Definiranje klase i resursa

Proširenje hijerarhija klasa i relacija se izvodi korištenjem *rdfs:subClassOf* i *rdfs:subPropertyOf* relacijama. Tako je moguće definirati klase *res:FerrousMetal* i *res:NonFerrousMetal* kao podklase klase *res:Metal.* Ovime se zapisuje da je svaka individua koja će se naći u podklasama *res:FerrousMetal* i *res:NonFerrousMetal* ujedno i član klase *res:Metal*.

```
res:FerrousMetal      rdfs:subClassOf     res:Metal .
res:NonFerrousMetal   rdfs:subClassOf     res:Metal .
```

Slika 11. ilustrira podklase *res:FerrousMetal* i *res:NonFerrousMetal* dodijeljene klasi *res:Metal*.

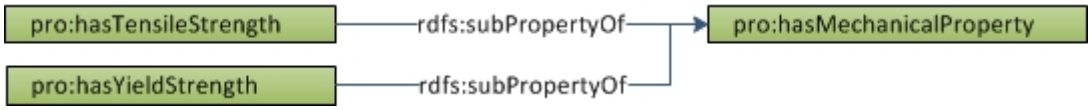


Slika 11. - Definiranje klasa kao podklase

Slično kao i stvaranje hijerarhije klasa, stvara se i hijerarhija relacija. Relacije *pro:hasTensileStrength* i *pro:hasYieldStrength* mogu pripadati grupi relacija *pro:hasMechanicalProperty*. Tako svrstane relacije definiraju da dva resursa koja su povezana relacijom *pro:hasTensileStrength* su ujedno povezana i relacijom *pro:hasMechanicalProperty*.

```
pro:hasTensileStrength    rdfs:subPropertyOf    pro:hasMechanicalProperty .
pro:hasYieldStrength      rdfs:subPropertyOf    pro:hasMechanicalProperty .
```

Slika 12. ilustrira relacije *pro:hasTensileStrength* i *pro:hasYieldStrength* dodijeljene relaciji *pro:hasMechanicalProperty* kao podrelacije.



Slika 12. - Definiranje relacija kao podrelacije

Još dvije bitne relacije u RDFS rječniku koje služe za svrstavanje resursa u klase su *rdfs:domain* i *rdfs:range.* Koristeći relaciju koja je definirana sa prethodno navedene dvije, resursi RDF trojca pod subjektom (*rdfs:domain*) i objektom (*rdfs:range*) se indirektno svrstavaju u klase.

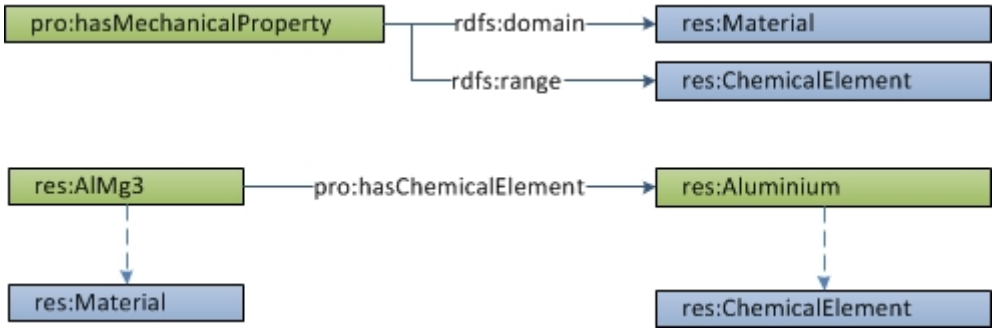
Neka postoje klase *res:Material* i *res:ChemicalElement,* te individue *res:AlMg3* i *res:Aluminium* koje nisu dodijeljene u nijednu klasu. Relacija koja veže navedene individue je *pro:hasChemicalElement*.

```
res:AlMg3    pro:hasChemicalElement    res:Aluminium .
```

Toj relaciji se mogu definirati *rdfs:domain* i *rdfs:range*.

```
pro:hasChemicalElement    rdfs:domain     res:Material ;
                          rdfs:range      res:ChemicalElement .
```

Iako individue *res:AlMg3* i *res:Aluminium* nigdje nisu prethodno dodijeljene na standardni način u odgovarajuće klase, samim korištenjem relacije *rdfs:domain* i *rdfs:range* RDFS procesor automatski dodjeljuje resurs *res:AlMg3* u klasu *res:Material*, a resurs *res:Aluminium* u klasu *res:ChemicalElement.* Slika 13. ilustrira takav primjer.

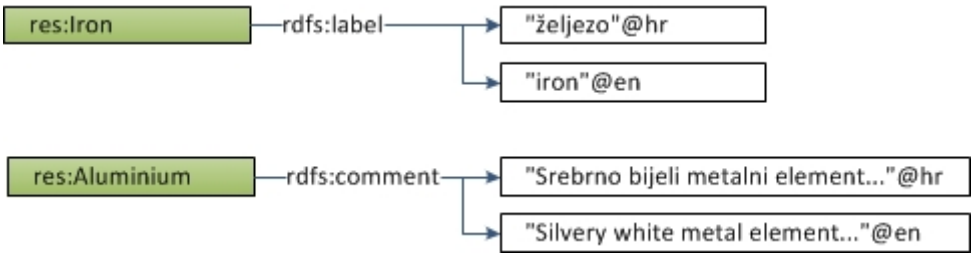


Slika 13. - Indirektno svrstavanje resursa u klase

Od ostalih često korištenih RDFS relacija postoje *rdfs:label* i *rdfs:comment*. Te relacije služe za naziv i opis resursa koje su lako čitljive ljudima s obzirom da se iz URI-a ponekad ne može saznati koji točno pojam predstavlja. Objekti ovih dviju relacija su doslovne vrijednosti i njima je poželjno uvijek dodijeliti jezik na kojem su napisane. Algoritmi i ljudi obično traže resurse upravo preko tih relacija.

mat:Iron	rdfs:label	"željezo"@hr , "iron"@en .
mat:Aluminium	rdfs:comment	"Srebrno bijeli metalni element..."@hr , "Silvery white metal element..."@en .

Slika 14. ilustrira relacije *rdfs:label* i *rdfs:comment*.



Slika 14. - Opisne relacije

2.2.2. OWL (Web Ontology Language)

Web Ontology Language je nadogradnja na RDFS, odnosno dodatni set relacija kojima se opisuju resursi i definiraju same relacije. Glavna značajka OWL-a je što dodaje mogućnosti za definiranje unija, presijecanja klasa, dodavanje karakteristika relacijama, te ograničavanja iznosa vrijednosti kao i ograničavanje broja individua sa kojima resurs može biti vezan.

Prva verzija je izdana 2004. godine i sastoji se od 3 profila. OWL Lite, OWL DL i OWL Full. OWL Lite je najjednostavniji način opisivanja podataka i stvaranja hijerarhije, sa najmanje dodanih relacija. Upravo iz tog razloga daje i najmanje semantike podacima, ali je najlakši za implementaciju i procesorsko odlučivanje. OWL DL (OWL Description Logic) je kompleksniji profil koji omogućava procesorima izvlačenje maksimalan broj indirektno definiranih veza iz RDF trojaca, zadržavajući pri tome mogućnost svih procesorskih odluka. OWL Full je najkompleksniji profil bez ograničenja. Zbog toga zna doći do problema kod procesorskog odlučivanja, poput beskonačnih petlji.

Svaki set relacija u pojedinom OWL profilu je sadržan i kompatibilan sa prethodnim. OWL Full sadrži OWL DL, a OWL DL sadrži OWL Lite. I rezultat svakog procesorskog odlučivanja u OWL Lite je jednak i u OWL DL. Isto tako, svaki rezultat u OWL DL je jednak i u OWL Full.

2009. godine je izdana nova verzija OWL2 koja proširuje osnovne OWL profile sa dodatnim mogućnostima. Ova verzija se dijeli na OWL 2 DL i OWL 2 Full. OWL 2 DL je ograničena verzija OWL 2 Full profila koja kao i u osnovnoj verziji olakšava implementaciju i povezivanje podataka za korisnika koji izrađuje ontologiju, a isto tako i za procesorsko odlučivanje.

Kao i u prvoj verziji postoje određeni profili: OWL 2 EL, OWL 2 RL i OWL 2 QL. Iako su ovi profili sadržani u OWL 2 Full verziji, oni nisu međusobno sadržani jedan u drugome za razliku od osnovne verzije i njenih profila.

Treba napomenuti da svi ovi profili predstavljaju standardne setove relacija i načine za stvaranje ontologije, ali nisu obavezni. Moguće je stvoriti vlastite profile, odnosno koristiti samo relacije koje su potrebne za izradu konkretne aplikacije ili baze podataka, kao što će biti u ovome radu. Ovi načini služe za olakšavanje čitanja i donošenja odluka procesorima pri izvlačenju indirektnih relacija iz RDF trojaca i ontologija. [2] Ovdje će biti objašnjen dio relacija i klasa iz OWL Lite profila koje će se pri izradi samog rada i koristiti.

Globalni *namespace* OWL rječnika je *http://www.w3.org/2002/07/owl#*, a uobičajeni prefiks koji se dodjeljuje je *owl*.

Ono što u RDFS-u predstavljaju *rdfs:Class* i *rdfs:Resource*, to u OWL Lite ontologiji predstavljaju *owl:Class* i *owl:NamedIndividual*. Na sličan način preko relacije *rdf:type* je moguće definirati resurs kao klasu ili individuu.

```
res:Metal       rdf:type    owl:Class .
res:AlMg3       rdf:type    owl:NamedIndividual .
```

Vrste i karakteristike relacija

Pomoću OWL klasa je moguće odrediti vrstu relacije. Postoje dvije klase, odnosno dvije vrste relacija, *Object property* i *Datatype property*. Ako se relacija svrsta u klasu *owl:ObjectProperty* tada ona ima funkciju povezivanja klasa i/ili individua međusobno, a svrstana u klasu *owl:DatatypeProperty* ima funkciju povezivanja klasa i/ili individua sa doslovnim vrijednostima. Na sljedeći način se relacije dodjeljuju u odgovarajuću klasu i pri tome definiraju.

```
pro:hasChemicalElement    rdf:type    owl:ObjectProperty .
pro:hasSymbol             rdf:type    owl:DatatypeProperty .
```

Relacija *pro:hasChemicalElement* će vezati dvije individue poput *res:AlMg3* i *res:Aluminium*, dok će relacija *pro:hasSymbol* vezati resurse poput individue *res:Iron* sa doslovnom vrijednosti *"Fe"*.

Osim definiranja vrsta pojedinih relacija, OWL omogućuje i dodjeljivanje karakteristika relacijama također ih svrstavajući u određene klase kako bi OWL procesor mogao izvući dodatnu semantiku i povezanost iz RDF trojaca. Tako postoje relacija *owl:inverseOf*, i klase *owl:TransitiveProperty*, *owl:SymmetricProperty*, *owl:FunctionalProperty* i *owl:InverseFuncionalProperty*.

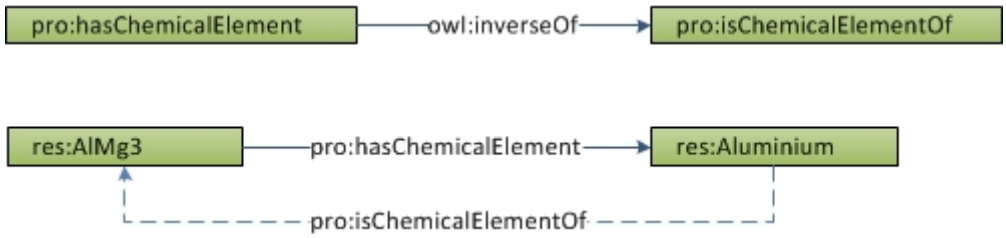
Relacijom *owl:inverseOf* je moguće nekoj relaciji dodijeliti inverznu relaciju. Neka postoje relacije *pro:hasChemicalElement* i *pro:isChemicalElementOf* pomoću kojih se povezuje određen materijal sa njegovim sastavnim kemijskim elementima i obrnuto, kemijski element koji se nalazi u određenim materijalima. Relaciji *pro:hasChemicalElement* se inverzna relacija *pro:isChemicalElementOf* definira na sljedeći način:

```
pro:hasChemicalElement    owl:inverseOf    pro:isChemicalElementOf .
```

Sada kod zapisanog RDF trojca koji govori da legura *res:AlMg3* sadrži *res:Aluminium*, OWL procesor može zaključiti i da je *res:Aluminium* sadržan u leguri *res:AlMg3* iako to nigdje u bazi nije direktno zapisano preko RDF trojca.

```
res:AlMg3       pro:hasChemicalElement    res:Aluminium .
res:Aluminium   pro:isChemicalElementOf   res:AlMg3 .
```

Slika 15. ilustrira *owl:inverseOf* relaciju i njenu funkciju.



Slika 15. - Definiranje inverzne relacije

Suprotno od inverzne relacije, postoji simetrična klasa *owl:SymmetricProperty* u kojoj je moguće definirati simetrične relacije. Iako u ovome radu neće biti korištena, valja je spomenuti jer je ona jedna od glavnih značajki OWL-a. Simetričnom relacijom vežući subjekt sa objektom RDF trojca, ujedno se povezuje objekt sa subjektom preko iste te relacije.

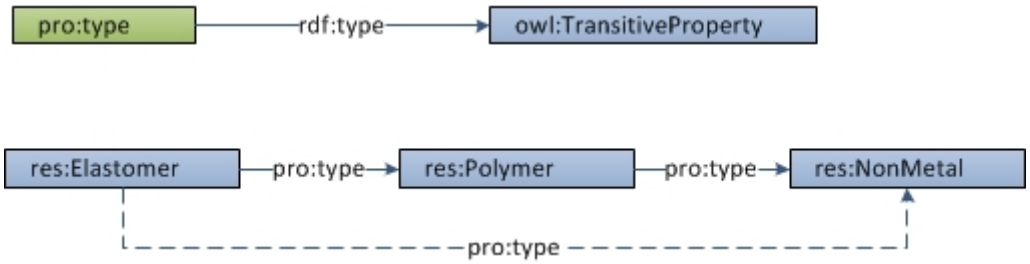
owl:TransitiveProperty je klasa u koju se dodaju relacije koje OWL procesor prepoznaje kao prijenosne. Neka postoji relacija *pro:type* koja je definirana kao prijenosna.

```
pro:type        rdf:type        owl:TransitiveProperty .
```

Preko te relacije je moguće povezati klasu *res:Elastomer* i klasu *res:Polymer*, te isto tako klasu *res:Polymer* i klasu *res:NonMetal*.

```
res:Elastomer   pro:type        res:Polymer .
res:Polymer     pro:type        res:NonMetal .
```

Ovime je indirektno povezan podatak da je klasa *res:Elastomer* vezana preko relacije *pro:type* sa klasom *mat:NonMetal*, odnosno OWL procesor prepoznaje da su elastomeri ujedno vrsta nemetalnih materijala iako to nigdje nije direktno zapisano preko RDF trojca. Slika 16. ilustrira takvu vezu.



Slika 16. - Definiranje kaskadne relacije

Dodajući relaciju u klasu *owl:FunctionalProperty,* ta relacija tada može biti povezana sa maksimalno jednom vrijednosti, odnosno maksimalno jednim resursom. Neka je relacija *pro:hasAtomicNumber* svrstana u klasu *owl:FunctionalProperty*.

```
pro:hasAtomicNumber    rdf:type    owl:FunctionalProperty .
```

Nakon toga navedena relacija *pro:hasAtomicNumber* može sadržavati samo jednu vrijednost.

```
mat:Iron    pro:hasAtomicNumber    "26"^^xsd:integer .
```

Postoji i klasa *owl:InverseFunctionalProperty* koja kao i *owl:SymmetricProperty* neće biti korištena, ali ju također treba spomenuti. Relacija unutar te klase ima isto svojstvo kao i relacije unutar *owl:FunctionalProperty*, ali sa inverznom karakteristikom, što znači da resurs na mjestu objekta može biti povezan maksimalno sa jednim subjektom.

Jednakost i nejednakost

Pomoću OWL relacija se mogu definirati iste klase, relacije i individue koje se pojavljuju unutar vlastite baze i drugih baza, odnosno između različitih ontologija.

Prva i osnovna relacija kojom je moguće definirati da su dvije individue različitog imena ustvari isti resurs je *owl:sameAs*. Neka postoje dva različita URI-a za željezo kao kemijski element: *res:Iron* i *res:Iron_(ChemicalElement)*. Pomoću relacije *owl:sameAs* je moguće reći OWL procesoru da se ustvari radi o istom pojmu. Tada je moguće izvući sve ostale relacije vezane uz jedan i drugi URI, te ih prikazati kao zajedničke.

```
res:Iron    owl:sameAs    res:Iron_(ChemicalElement) .
```

Ova relacija u OWL Lite profilu vrijedi samo za individue. Kada je potrebno striktno definirati različitost individua, koristi se relacija *owl:differentFrom*. Tako je na primjer moguće razlikovati resurs *item:Iron* koji predstavlja glačalo i *res:Iron* koji predstavlja željezo. Kako bi bilo strogo definirano da se ne radi o istom resursu, koristi se relacija *owl:differentFrom*.

```
item:Iron    owl:differentFrom    res:Iron .
```

Za definiranje jednakosti klasa koristi se relacija *owl:equivalentClass*, dok za definiranje jednakosti relacija koristi se *owl:equivalentProperty*. Treba napomenuti da se ovdje ne deklariraju klase kao isti pojmovi, već klase sa istim individuama, a u stvarnom svijetu mogu biti dva različita pojma, odnosno resursa.

```
res:AlAlloy    owl:equivalentClass       res:AluminiumAlloy .
pro:hasCE      owl:equivalentProperty    pro:hasChemicalElement .
```

Valja spomenuti i ostale relacije za ograničavanje drugih relacija koje se koriste OWL-u. *owl:allValuesFrom*, *owl:someValuesFrom*, *owl:minCardinality*, *owl:maxCardinality* i *owl:cardinality*, ali u OWL Lite profilu mogu poprimiti samo vrijednosti 0 i 1 pa se u ovome radu neće koristiti.

2.2.3. Izrada ontologije

Svaka ontologija se izrađuje prema namjeni. Koliko daleko u hijerarhiji klasa i podjeli resursa se želi ići ovisi o namjeni aplikacije koja će tu ontologiju koristiti. Isto tako količina relacija koje se izrađuju za opis resursa nisu ograničene s tehničke strane.

Jedna od glavnim značajki i ideja povezanih podataka je ta da su oni naknadno proširivi i to ne samo u vlastitoj ontologiji već globalno. Zbog toga pri modeliranju vlastite ontologije, dovoljno je izraditi hijerarhiju klasa i smjestiti individue u njima koje su značajne za vlastitu aplikaciju. Druge aplikacije i korisnici mogu uzeti krajnje individue, njih preko relacije poput *owl:sameAs* poistovjetiti sa novom klasom, te nakon toga proširiti, detaljizirati i opisati za njihove potrebe. Na taj način se širi mreža povezanih podataka.

Za izradu ontologije, s tehničke strane, moguće je koristiti običan tekst editor poput Notepad-a, te u njemu preko RDF trojaca definirati klase, individue i relacije, na isti način kao same podatke. Te RDF trojce je moguće pisati na bilo koji od standardiziranih načina serijalizacije. Ovaj način je pogodan kada se ontologiju stvara računalo nekim algoritmom za raščlanjivanje podataka sa postojećih izvora, te kreiranje RDF trojaca i spremanja u bazu podataka. Za čovjeka bi taj postupak bio predugačak pogotovo sa velikim i detaljnim ontologijama kod kojih bi vjerojatno puno stvari ostalo nedefinirano.

Valja spomenuti da za imenovanje resursa ne postoji obavezno pravilo. Jedan od načina je *CamelCase* kod kojeg se sve riječi pišu zajedno, ali svaka riječ počinje velikim slovom. Običaj je da se individue i klase pišu velikim početnim slovom, a relacije malim.

Prije svega je potrebno posjedovati internet domenu na kojoj će se nalaziti ontologija, odnosno koja će biti sadržana u URI-ima koji će predstavljati resurse materijala. U ovome dijelu treba razmišljati o tome da će URI predstavljati pojmove na koje će se u budućnosti vezati druge ontologije drugih aplikacija. Iz tog razloga je poželjno da ta domena bude u stalnom vlasništvu vlasnika same ontologije kako bi imao pod kontrolom sadržaj iza nje. Čak i da je domena neaktivna, ta adresa, odnosno URI će za ontologije i dalje predstavljati određen resurs. [2]

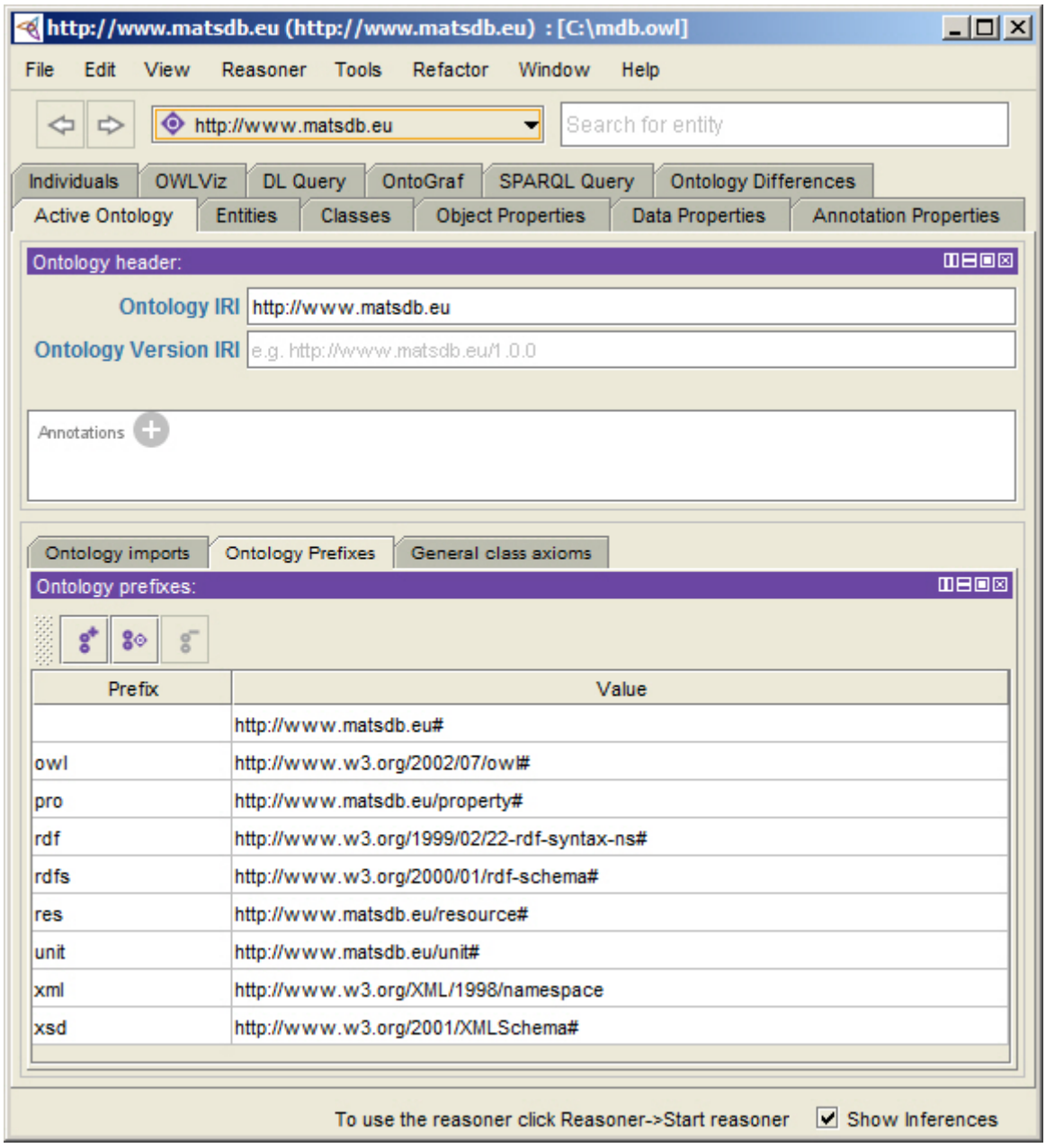
2.2.4. Protégé

Postoje računalni programi kod kojih se preko grafičkog korisničkog sučelja na jednostavan i brz način mogu izrađivati, modificirati i vizualizirati ontologije. U takvim programima je dosta stvari automatizirano, te samu izradu nadgleda i provjerava sam program koji i javlja greške, odnosno nedefinirane elemente ontologije.

Protégé je jedan od takvih računalnih programa koji je nastao na Stanford University-u i besplatan je za korištenje. Potpuno je kompatibilan sa OWL 2 ontologijom, te omogućava korisnicima izradu, modificiranje, provjeru i vizualizaciju novonastalih ontologija i baza podataka. Također podržava implementaciju dodataka koje korisnici sami mogu kreirati ili instalirati već postojeće.

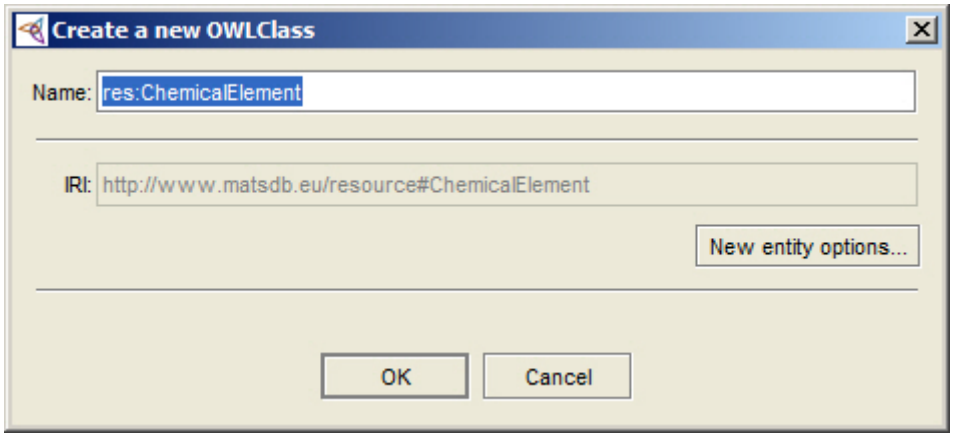
Ovdje će kroz par primjera iz ontologije materijala biti objašnjena općenita izrada ontologije pomoću Protégé-a, a u drugom dijelu rada će biti navedena cijela ontologija materijala izrađena na načine prikazane kroz ove primjere.

Odmah pri otvaranju Protégé-a u glavnom prozoru je moguće deklarirati glavnu domenu i prefikse koji će se koristiti u ontologiji. Standardne prefikse koji su sadržani u OWL ontologiji kao što su *rdf*, *rdfs*, *owl, xml* i *xsd* Protégé automatski dodjeljuje. Slika 17. prikazuje glavni prozor Protégé-a u kojem su dodijeljeni prefiksi i domena.



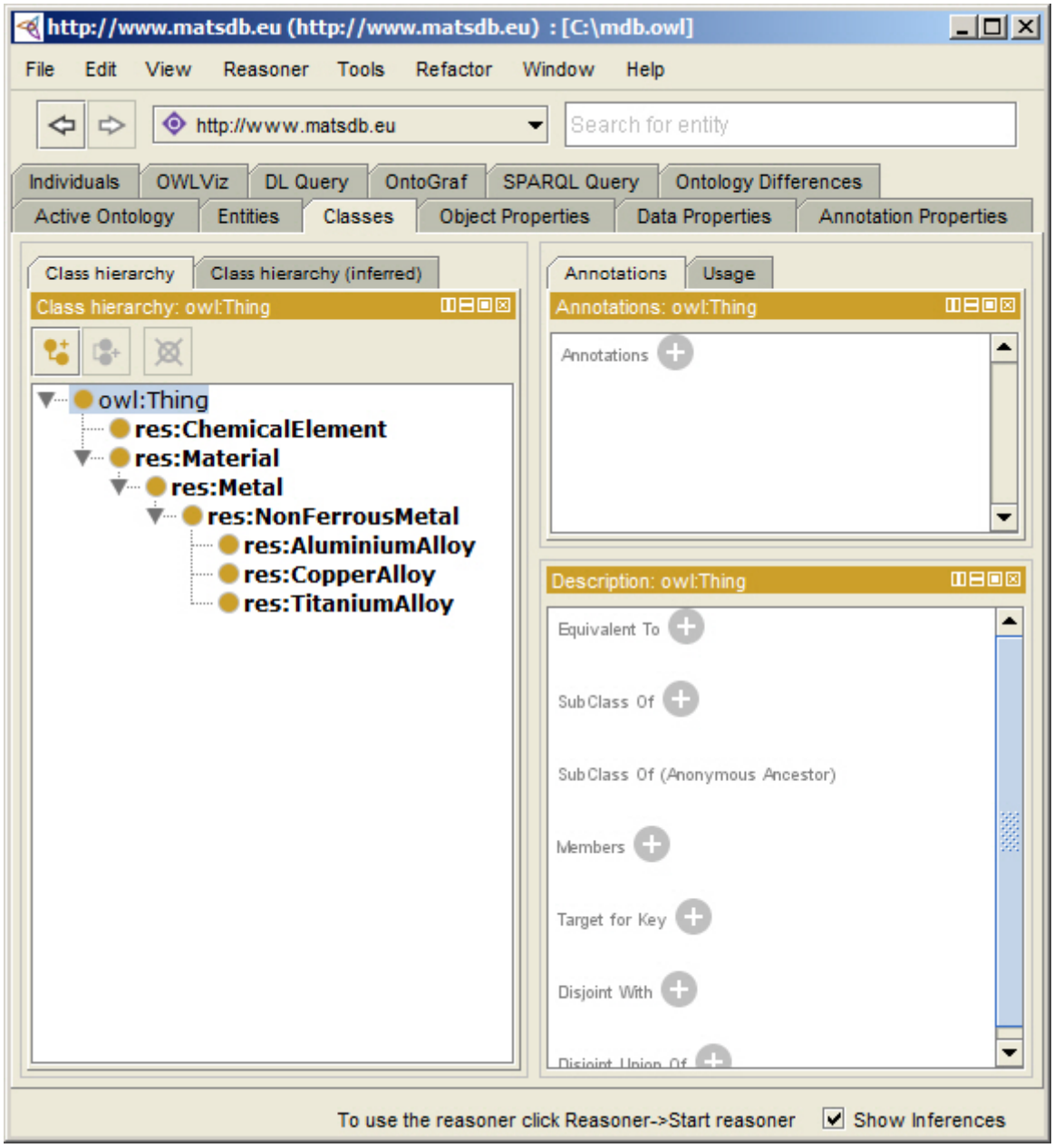
Slika 17. - Glavni prozor Protégé-a

Nakon toga je potrebno izraditi klase i podklase koje će sadržavati individue raznih materijala. One se izrađuju u kartici *Classes*. U OWL ontologiji vrh hijerarhije predstavlja klasa *owl:Thing*, a sve ostale klase, uključujući glavne klase i podklase su ustvari podklase *owl:Thing*. Klikom na gumb *Add subclass* se otvara novi prozor u koji se upisuje URI koji će predstavljati novu klasu. S obzirom da su u glavnom prozoru zadani prefiksi *res*, *pro* i *unit*, moguće je umjesto cijelog URI-a pisati prefiks i ime klase odvojene dvotočkom, odnosno *qname*. Slika 18. prikazuje prozor za kreiranje nove klase.



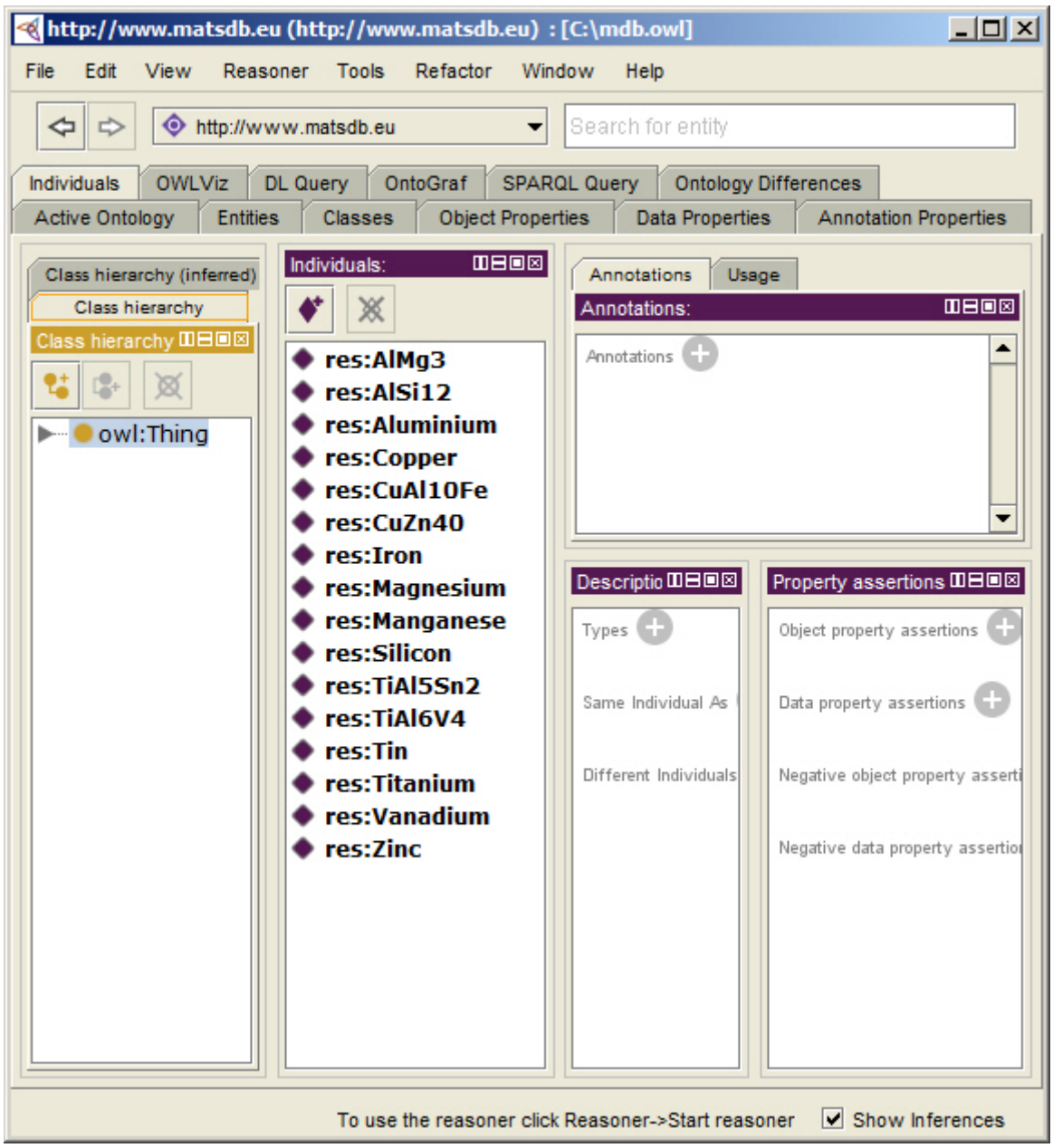
Slika 18. - Kreiranje nove klase u Protégé-u

Slika 19. prikazuje par klasa i podklasa iz ontologije materijala kreiranih u kartici *Classes*.



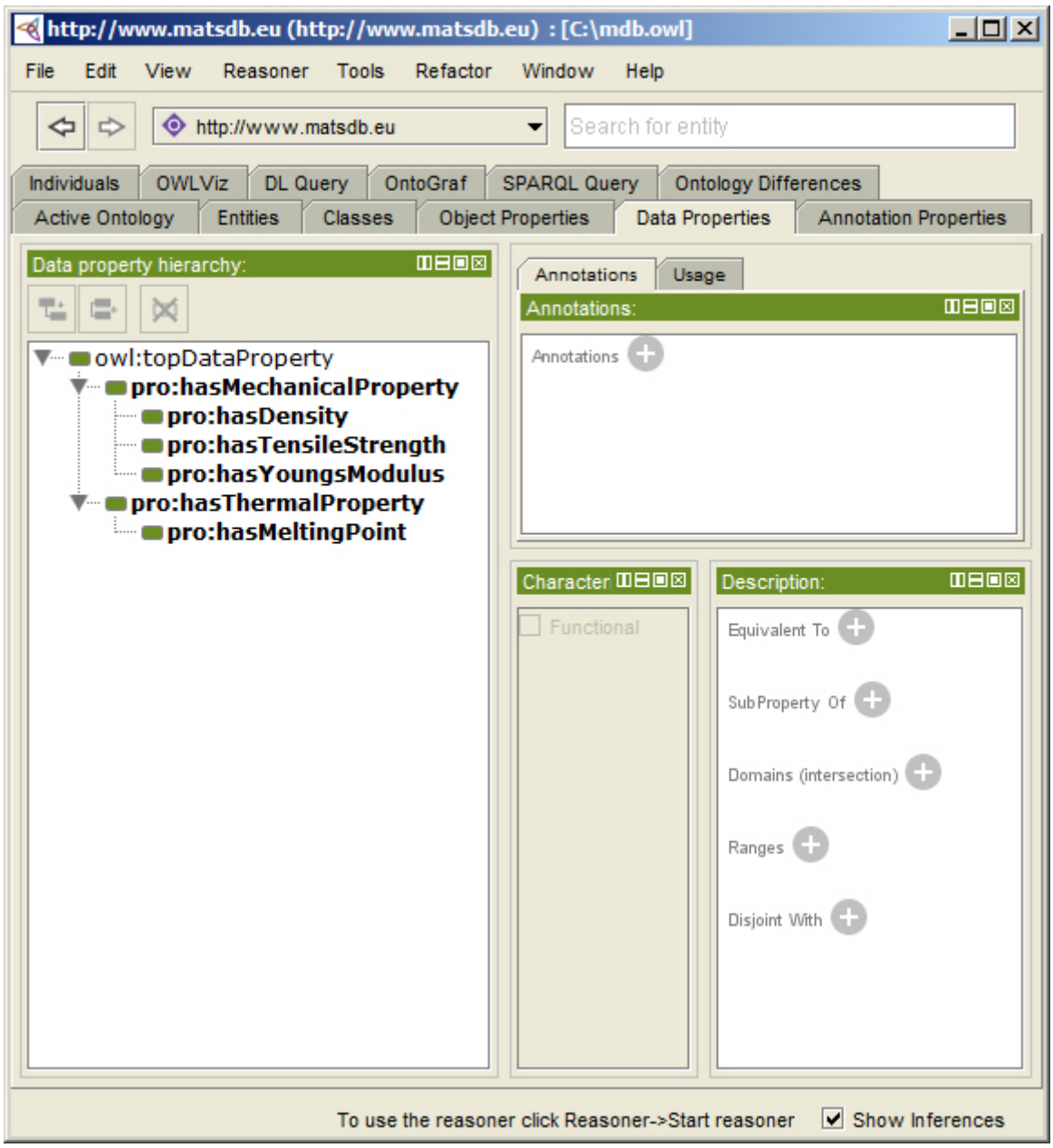
Slika 19. - Prikaz kreiranih klasa u Protégé-u

Nakon kreiranih klasa, moguće je dodati individue u njih. To se radi u kartici *Individuals*. Klikom na gumb *Add individual* otvara se novi prozor u kojem se na isti način kao i za klasu upisuje URI ili *qname* pojedine individue. Slika 20. prikazuje dio individua iz ontologije materijala kreiranih u kartici *Individuals*.



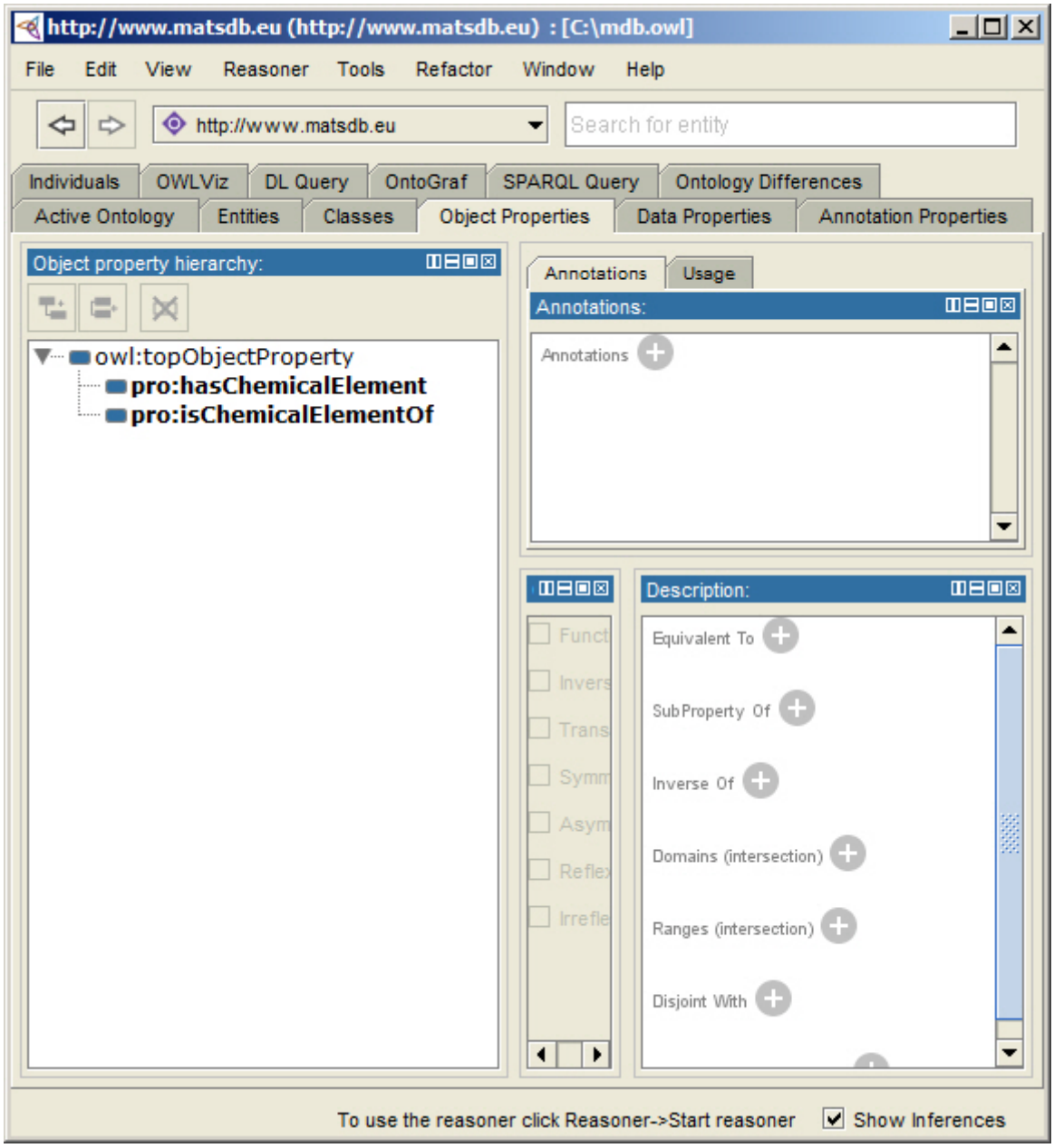
Slika 20. - Prikaz kreiranih individua u Protégé-u

U karticama *Object Properties* i *Data Properties* definiraju se relacije koje će povezivati klase, individue i doslovne vrijednosti. U kartici *Object Properties* se definiraju relacije koje će povezivati klase i individue međusobno, a u *Data Properties* kartici se deklariraju relacije koje će povezivati klase i individue sa doslovnim vrijednostima. Kao i sa klasama, moguće je stvoriti hijerarhiju podrelacija. Vrh hijerarhije relacija u OWL ontologiji će predstavljati relacije *owl:topObjectProperty* i *owl:topDataProperty*. Klikom na gumb *Add sub property* otvara se novi prozor. Kao za klase i individue u njega se upisuje URI ili *qname* nove relacije. Slika 21. prikazuje dio relacija iz ontologije materijala kreiranih u kartici *Data Properties*.



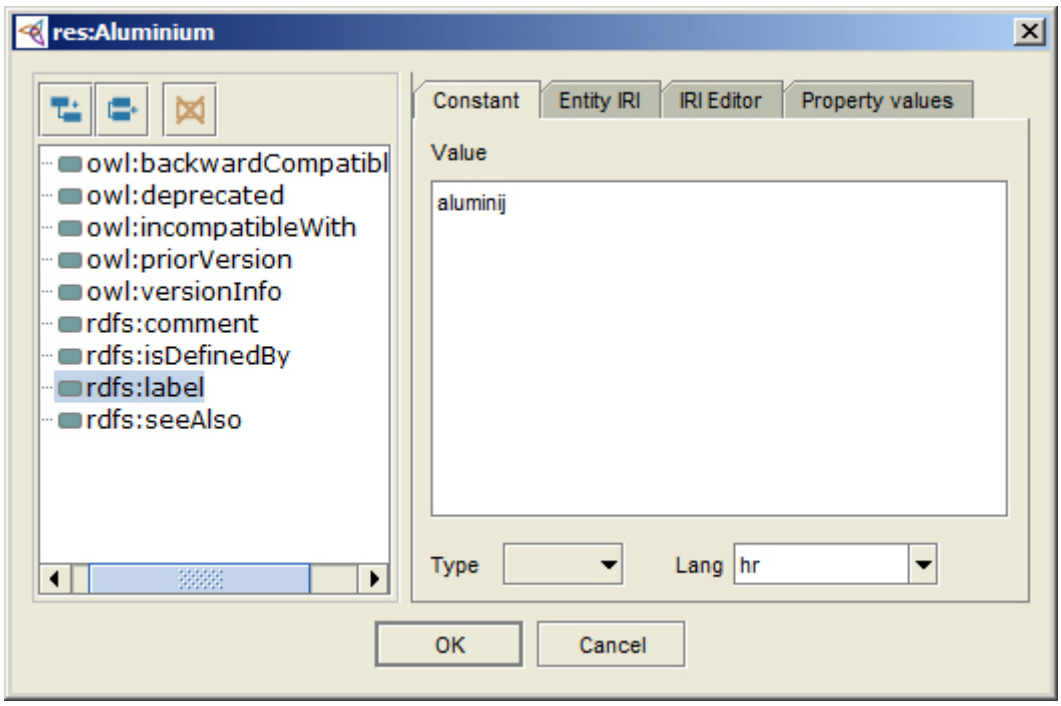
Slika 21. - Prikaz kreiranih datatype relacija u Protégé-u

Slika 22. prikazuje dio relacija iz ontologije materijala kreiranih u kartici *Object Properties*.



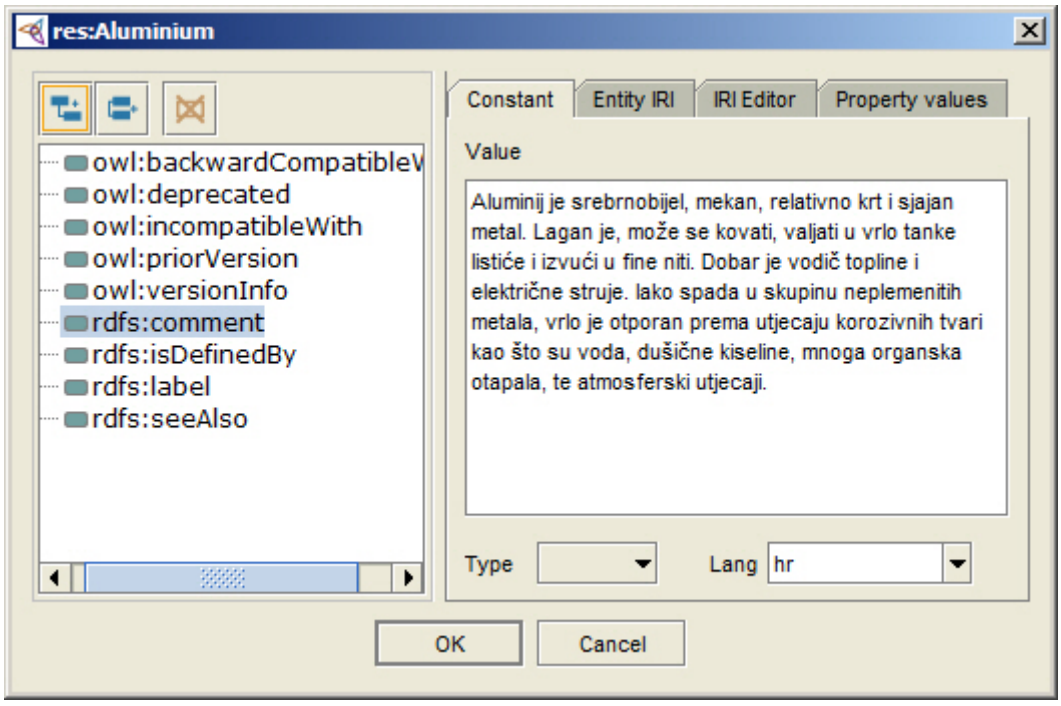
Slika 22. - Prikaz kreiranih object relacija u Protégé-u

Nakon kreiranja svih klasa, individua i relacija moguće im je dodati lako čitljiva imena za čovjeka, odnosno doslovne vrijednosti preko *rdfs:label* i *rdfs:comment* relacija. Klikom na gumb *Annotations* otvara se novi prozor u kojem je potrebno odabrati *rdfs:label*, te pod *Value* upisati doslovnu vrijednost. Pri tome je moguće odrediti i jezik na kojem je napisana, te vrstu doslovne vrijednosti. Slika 23. prikazuje taj prozor.



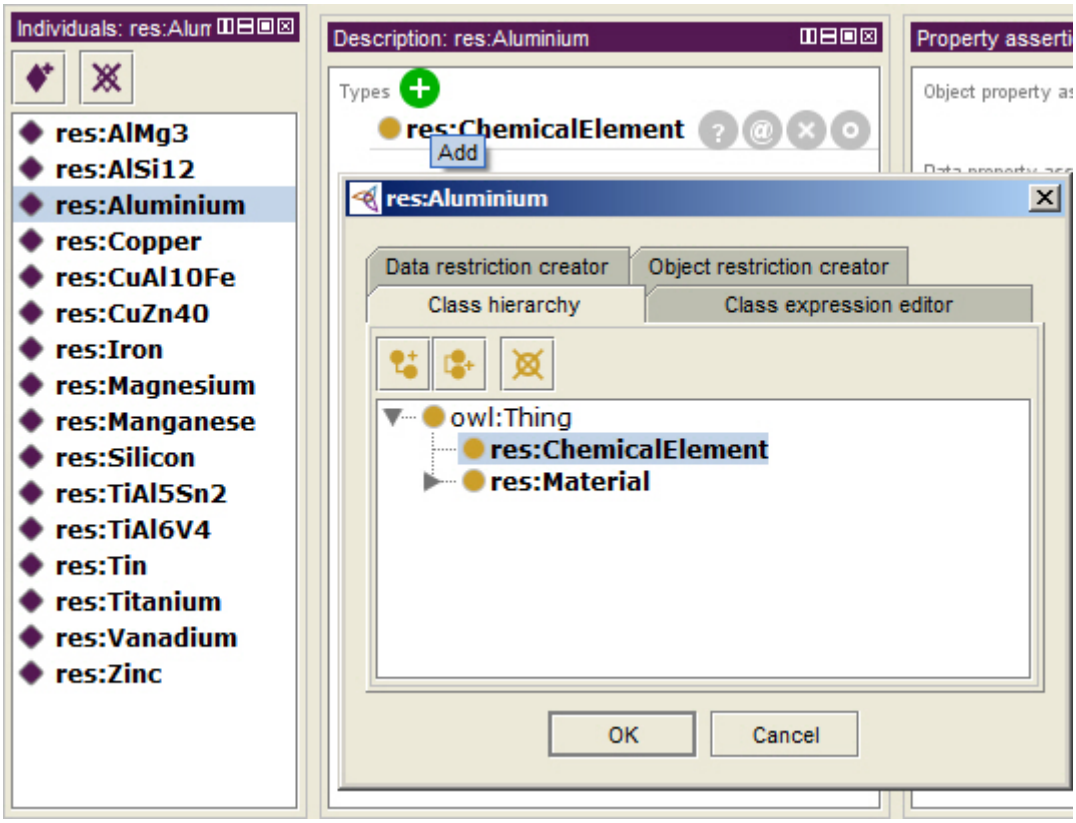
Slika 23. - Dodavanje opisne relacije naziva u Protégé-u

Na isti način, ali odabirom relacije *rdfs:comment* je moguće upisati komentar o resursu, odnosno dodatno ga opisati na čovjeku razumljiv način. Slika 24. prikazuje taj prozor.



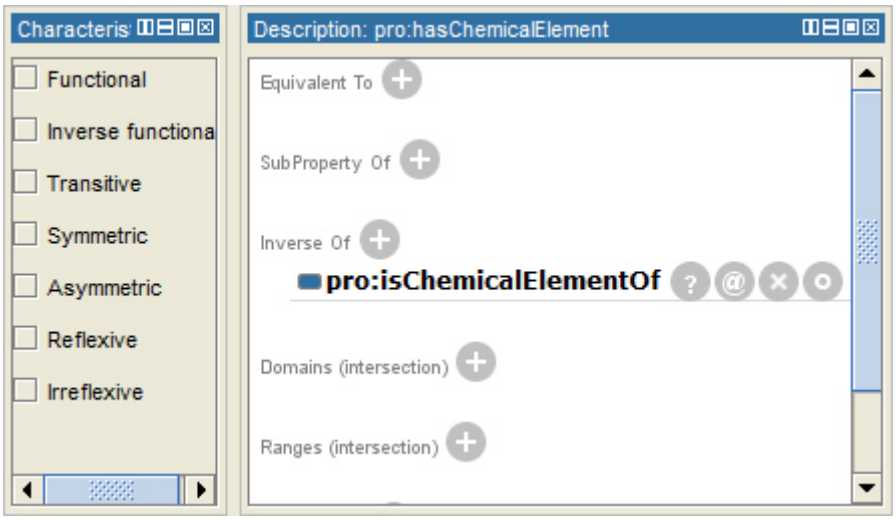
Slika 24. - Dodavanje opisne relacije komentara u Protégé-u

Nakon toga je potrebno individue dodati u željene klase. Klikom na gumb *Types* otvara se novi prozor u kojem je potrebno odabrati klasu kojoj se pridružuje individua. Slika 25. prikazuje dodavanje individue *res:Aluminium* u klasu *res:ChemicalElement*.



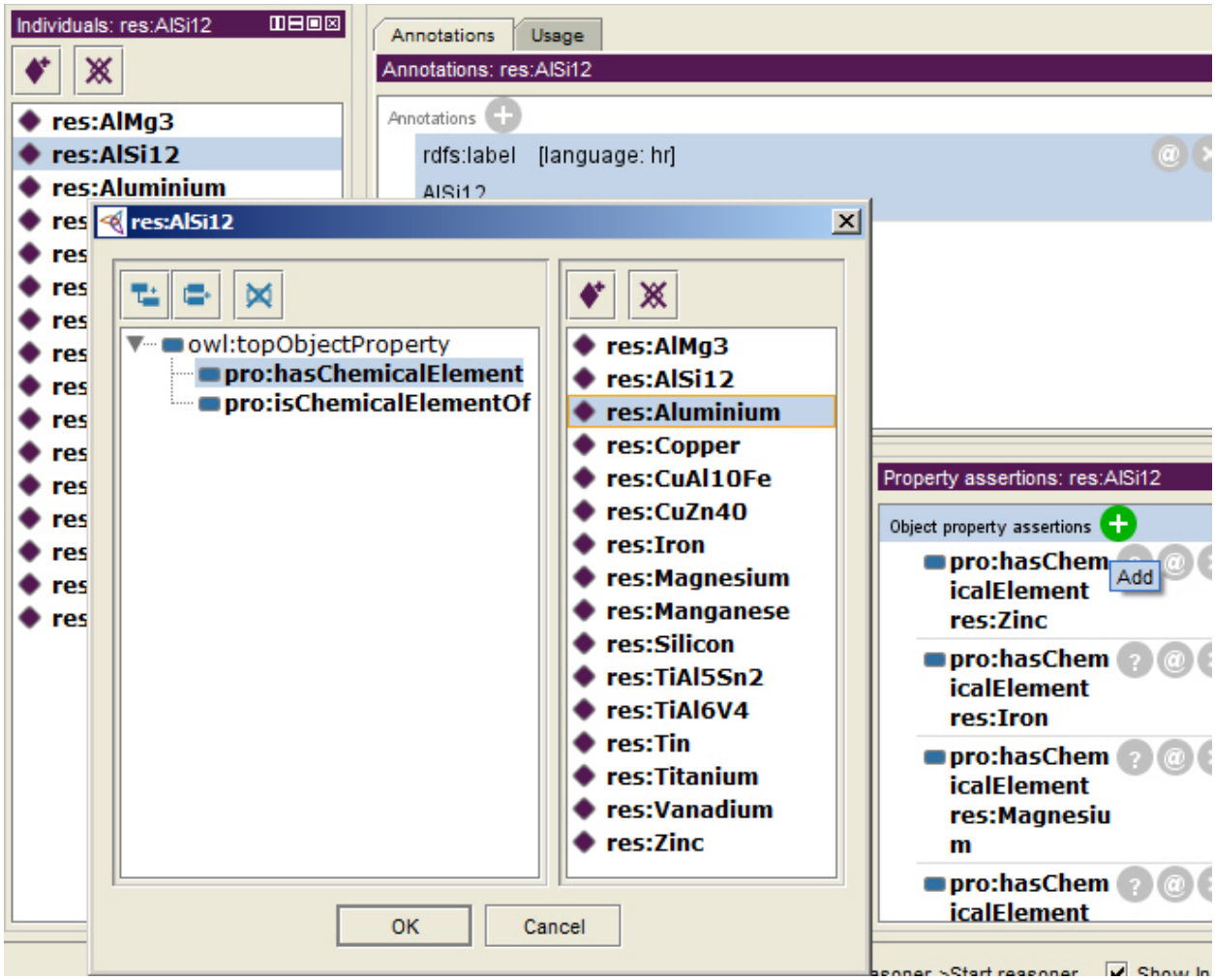
Slika 25. - Dodavanje individue u odgovarajuću klasu u Protégé-u

Kada su upisane sve klase, individue i relacije, prije samog povezivanja potrebno je još odrediti i karakteristike relacija prema OWL 2 ontologiji. U karticama za *Object Properties* relacije postoje opcije za određivanje karakteristika *Functional*, *Inverse functional, Transitive, Symmetric, Asymmetric, Reflexive i Irreflexive*, dok u kartici za *Data Properties* relacije, postoji samo *Functional*. U ovim je opcijama također moguće odrediti *rdfs:domain* i *rdfs:range* relacije. Slika 26. prikazuje definiranje relacije *pro:hasChemicalElement* kao inverzne relacije od *pro:isChemicalElementOf*.



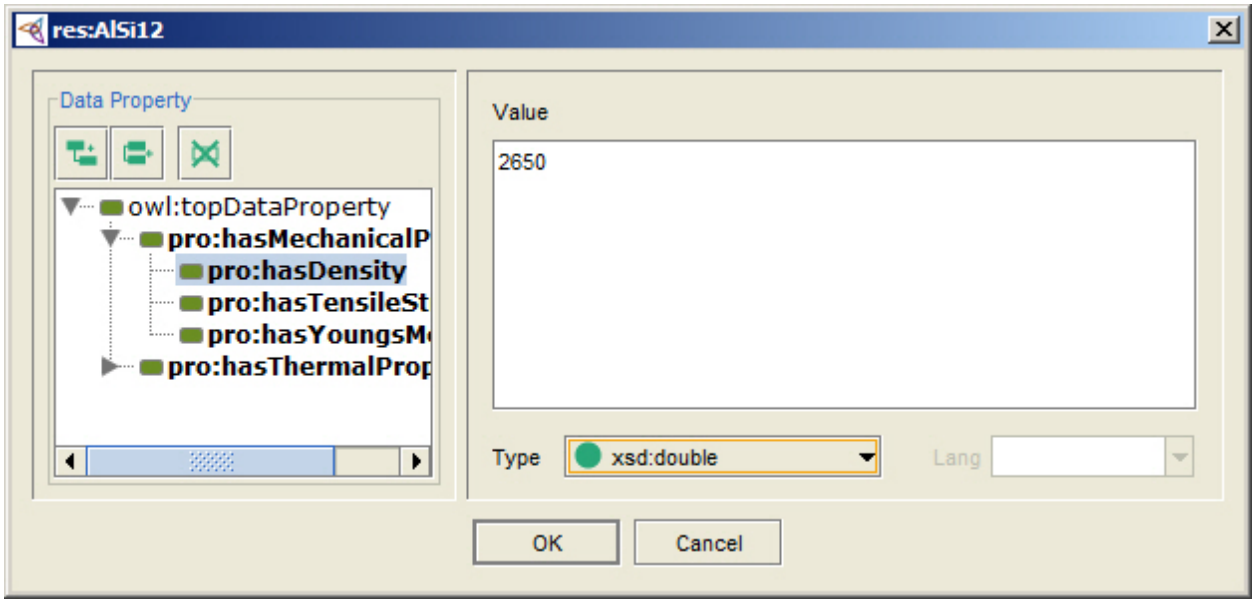
Slika 26. - Definiranje dodatnih karakteristika relacija u Protégé-u

U sljedećem koraku je potrebno povezati podatke i dodijeliti im doslovne vrijednosti. U kartici *Individuals* odabirom pojedine individue moguće ju je povezati sa drugom individuom ili joj dodijeliti doslovnu vrijednost. Nakon toga u dijelu prozora *Property assertions* klikom na gumb *Object property assertions* otvara se novi prozor u kojem se odabire relacija i individua sa kojom se želi povezati odabrana individua. Slika 27. prikazuje taj postupak.



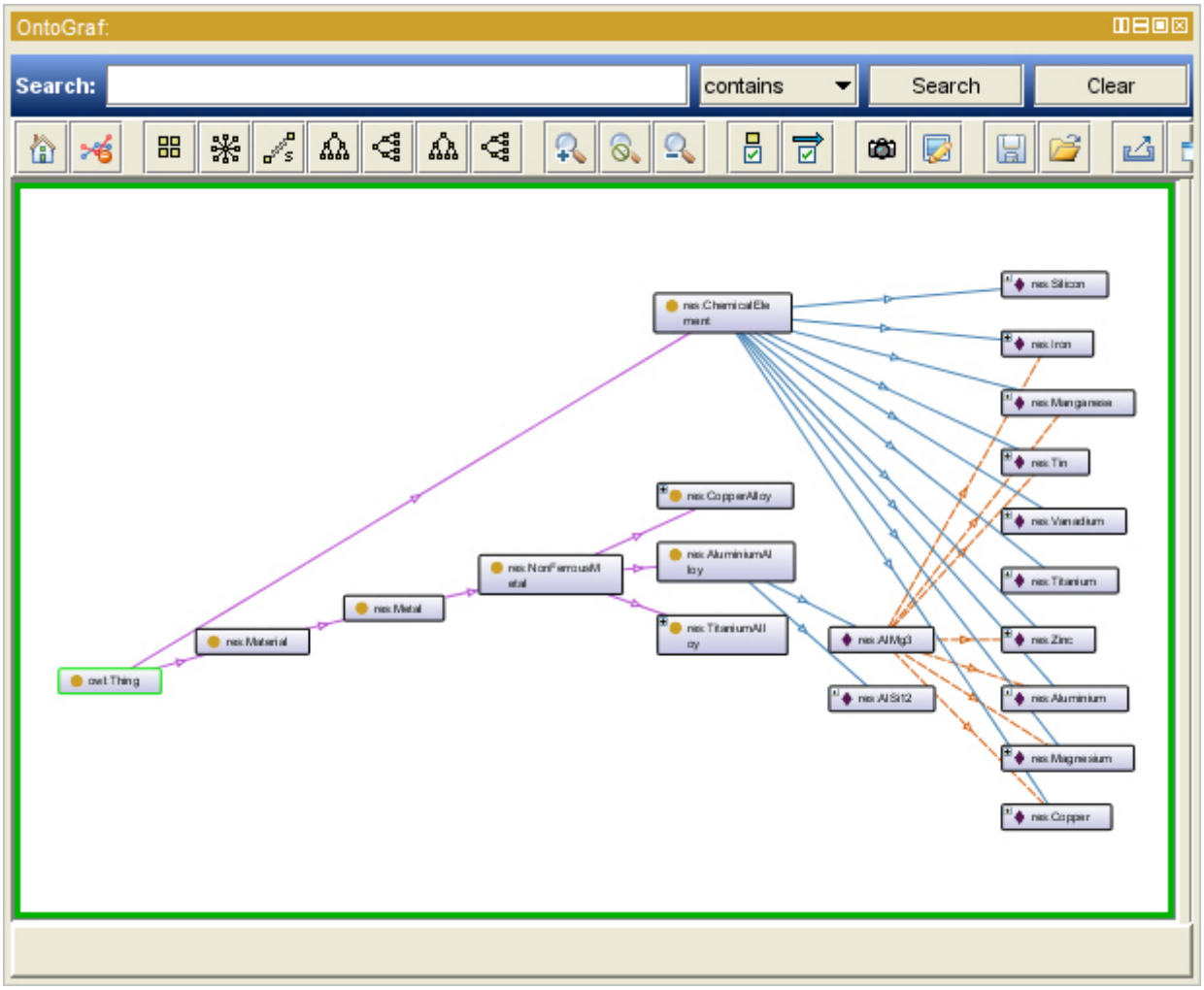
Slika 27. - Povezivanje individua u Protégé-u

Na isti način klikom na gumb *Data property assertions* dodaju relacije i upisuje doslovna vrijednost koja se želi povezati sa odabranom individuom. Slika 28. prikazuje taj prozor.



Slika 28. - Dodavanje doslovnih vrijednosti u Protégé-u

Za vizualizaciju ontologije se može koristiti već ugrađena opcija u Protégé-u OntoGraf ili se može instalirati jedan od dodataka kao što je OWL Viz. Slika 29. prikazuje vizualizaciju pomoću OntoGraf-a.



Slika 29. - Vizualizacija ontologije pomoću OntoGraf-a

Protégé također ima ugrađene neke od procesora za zaključivanje odnosno *reasoner*. Aktivacijom jednog od njih Protégé povezuje podatke koji nisu direktno definirani u ontologiji, već su nove relacije i RDF trojci izvedeni iz semantičkih veza postojećih.

Na kraju je novu ontologiju moguće spremiti preko neke od ponuđenih načina serijalizacije. Preporuka je Turtle format jer je jednostavan i najrazumljiviji čovjeku, dok je s druge strane kompatibilan sa svim ostalim aplikacijama.

3. SPARQL (SPARQL Protocol and RDF Query Language)

Standardni jezik za pretraživanje povezanih podataka je SPARQL. Ono što SQL jezik predstavlja za relacijske baze podataka, to SPARQL predstavlja za RDF model podataka. Prva verzija SPARQL 1.0 je postala standard 2008. godine od strane W3C. Nakon toga se razvija nova verzija SPARQL 1.1 koji je 2013. proglašen standardnim jezikom za pretraživanje RDF modela podataka.

SPARQL omogućuje pretraživanje podataka na temelju djelomičnog ili kompletnog podudaranja RDF trojaca u bazi i upitu koristeći pri tome unije upita, vezanjem više različitih upita i postavljanje neobaveznih upita. Pomoću njega je moguće filtriranje, grupiranje i sortiranje rezultata. U novoj verziji SPARQL 1.1 dodane su mogućnosti za manipulaciju podacima poput ubacivanja novih, te brisanja i ažuriranja postojećih podataka.

Vrlo bitna karakteristika SPARQL jezika je to što uz pretraživanje lokalnih baza podataka na vlastitom serveru, postoji mogućnost pretraživanja javnih i otvorenih baza podataka na drugim serverima, kojima inače ne bi imali pristup, kroz SPARQL *endpoint*. Na taj način RDF trojce iz vlastite baze je moguće vezati sa drugim bazama i podacima te dobiti kombinaciju zajedničkih rezultata sa više izvora. Za to nije potreban nikakav poseban program, već je pretraživanje moguće direktno na web stranici na kojem se SPARQL *endpoint* nalazi, kao i iz vlastite aplikacije.

Sintaksa koja se koristi za postavljanje SPARQL upita temeljena je na Turtle sintaksi i vrlo je slična. Unatoč tome, SPARQL nije ograničen samo na pretraživanje Turtle strukture, već je moguće pretraživati i ostale načine zapisa RDF trojaca. Štoviše, SPARQL je u mogućnosti pretraživati i relacijske baze podataka uz neke dodatke. [6]

Sljedeći primjer prikazuje opću strukturu SPARQL upita.

```
PREFIX
PREFIX

SELECT

WHERE {}

ORDER BY
LIMIT
OFFSET
```

Na samom početku, odnosno zaglavlju upita je mjesto predviđeno za deklaraciju prefiksa koji se koriste u upitima i time doprinose kompaktnosti samog zapisa. Za razliku od izvorne Turtle sintakse, u SPARQL upitu ispred riječi PREFIX ne postoji znak @, a na kraju deklaracije pojedinog prefiksa ne dolazi točka. Ime samog prefiksa se piše nakon ključne riječi PREFIX i odvojeno je dvotočkom od *namespace*-a koji se piše unutar izlomljenih zagrada. Za najčešće ponavljane prefikse je moguće deklarirati prefiks pomoću ključne riječi BASE umjesto ključne riječi PREFIX.

Na taj način se deklarira prefiks bez imena, odnosno moguće ga je pozvati upisivanjem same dvotočke što doprinosi još većoj kompaktnosti upita.

Nakon deklariranih prefiksa dolazi naredba koja određuje vrstu upita. Postoje SELECT, ASK, CONSTRUCT i DESCRIBE naredbe koje služe za dohvaćanje podataka i INSERT, DELETE, CLEAR i DROP naredbe kojima se preko upita manipulira podacima.

Pod klauzulu WHERE, između vitičastih zagrada, se upisuje jedan ili grupa trojaca koje će SPARQL procesor usporediti sa RDF trojcima u bazi podataka. Ovaj dio sadrži barem jedan trojac, jer bez ijednoga upit za pretraživanje ne bi imao na temelju čega pretražiti bazu podataka. Izuzetak su upiti za manipuliranje podacima jer kod njih u određenim slučajevima, poput umetanja novih, nije potrebno pronalaziti postojeće RDF trojce pa stoga oni ne zahtijevaju WHERE klauzulu. Unutar vitičastih zagrada mogu biti dodatni neobavezni uvjeti, unije uvjeta i filtri rezultata koji se pozivaju klauzulama OPTIONAL, UNION i FILTER.

Zadnji dio je predviđen za modificiranje rezultata upita, odnosno njihovo sortiranje, limitiranje, grupiranje i filtriranje, te provjeravanje postojećih i nepostojećih vrijednosti. Ovdje dolaze naredbe ORDER BY, LIMIT, OFFSET, GROUP BY, HAVING i VALUES.

RDF trojci zadani u WHERE klauzuli imaju sličnu sintaksu kao u Turtle strukturi. Zarezom se odvajaju i nižu objekti sa zajedničkim subjektom i relacijom, dok se točkom-zarez nižu relacije i objekti sa zajedničkim subjektom.

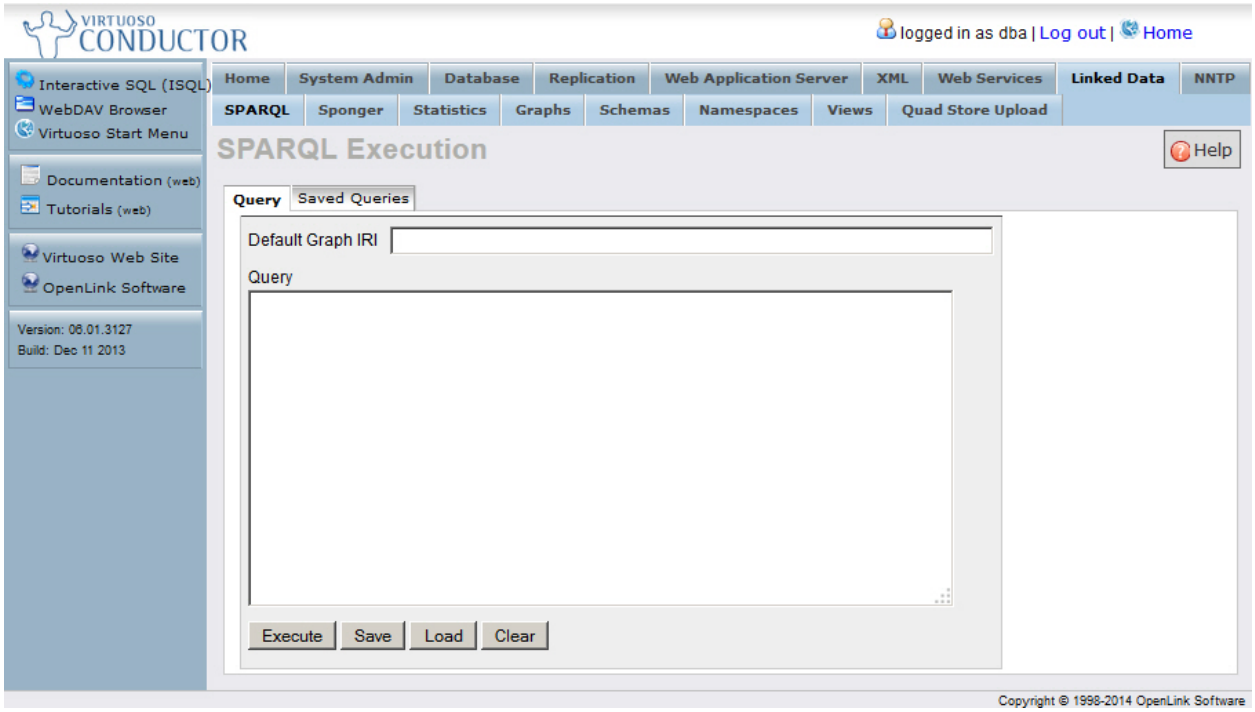
Razlika naspram Turtle strukture je u tome što na mjestu subjekta, relacije i/ili objekta mogu stajati varijable. Varijable se formiraju željenim imenom ispred kojeg stoji znak upitnik. One se deklariraju direktno na željenom mjestu u RDF trojcima.

Pretraživanje baze se tada temelji na pronalaženju RDF trojaca kojima se podudaraju elementi trojaca u WHERE klauzuli i RDF trojaca u bazi podataka, a na mjestu elemenata trojaca u WHERE klauzuli gdje se nalaze varijable, u RDF trojcima u bazi mogu stajati bilo koje vrijednosti koje će biti spremljene u te varijable.

3.1. Pretraživanje podataka

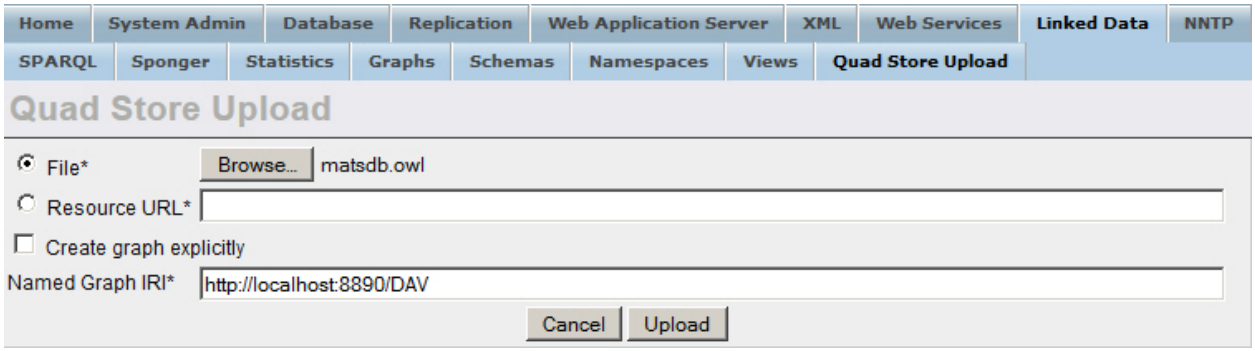
Kroz sljedeće primjere će biti objašnjene vrste SPARQL upita, te korištene funkcije upita pri izradi web aplikacije za izbor materijala. Treba napomenuti da će biti korišten samo dio baze materijala jer bi prikazivanje svih rezultata pojedinih upita (poput osnovnog upita prikazivanja svih RDF trojaca iz baze) bilo preveliko.

Kao baza podataka, te SPARQL procesor, biti će korišten OpenLink Virtuoso, programski paket koji sadrži *Quad Store* za pohranu RDF trojaca, SPARQL 1.1 jezik za pretraživanje baze, te SPARQL *endpoint* preko kojeg će biti postavljani upiti i prikazani rezultati. Slika 30. prikazuje OpenLink Virtuoso *endpoint*.



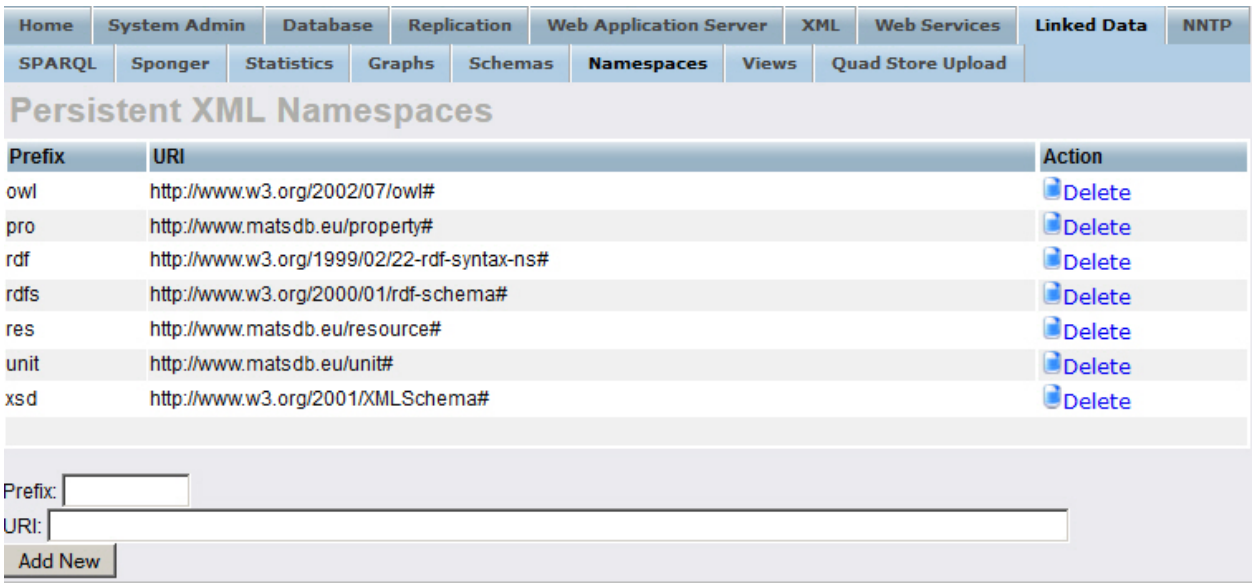
Slika 30. - OpenLink Virtuoso endpoint

Umetanje vlastite ontologije u OpenLink Virtuoso je jednostavno i brzo. Potrebno je u sučelju odabrati *Linked Data* karticu, zatim karticu *Quad Store Upload* te odabrati datoteku ontologije i kliknuti na gumb *Upload*. Slika 31. prikazuje *Quad Store Upload* karticu za umetanje ontologije koja se nalazi u datoteci *matsdb.owl* u bazu podataka.



Slika 31. - OpenLink Virtuoso učitavanje baze iz datoteke

OpenLink Virtuoso podržava zasebno definiranje prefiksa pa ih nije potrebno u svakome upitu posebno pisati što doprinosi kompaktnijim upitima. Slika 32. prikazuje karticu za definiranje prefiksa, kao i same prefikse koji su prethodno upisani *res*, *pro*, *unit*, *rdf*, *rdfs*, *owl* i *xsd*.



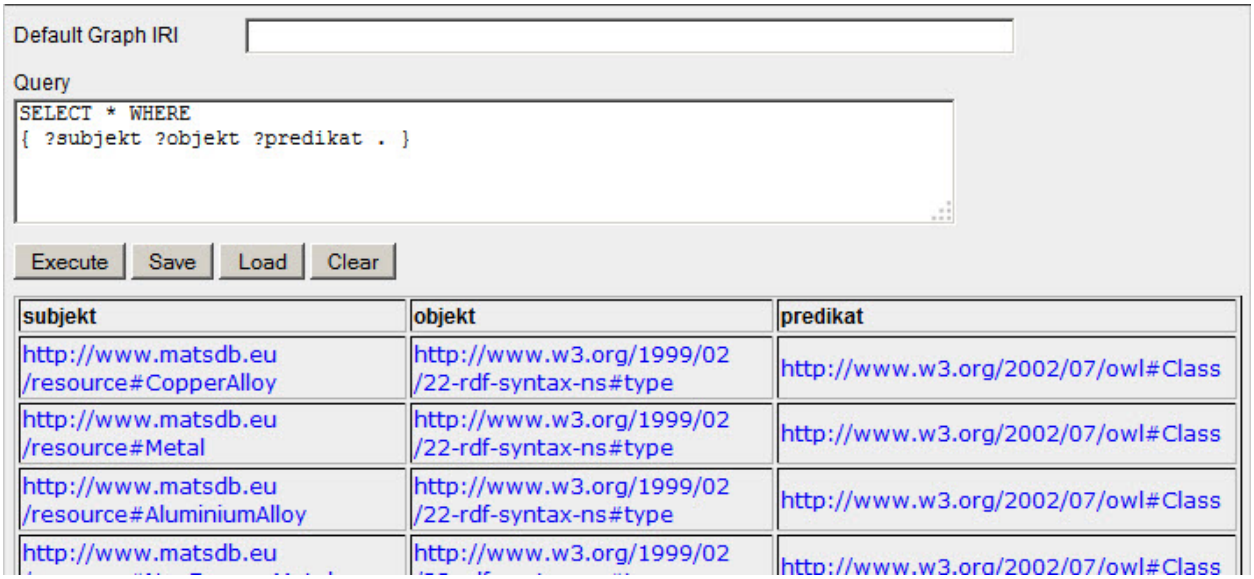
Slika 32. - OpenLink Virtuoso kartica za definiranje prefiksa

3.1.1. Osnovne vrste SPARQL upita [6]

Osnovne vrste SPARQL upita čine naredbe SELECT, ASK, CONSTRUCT i DESCRIBE.

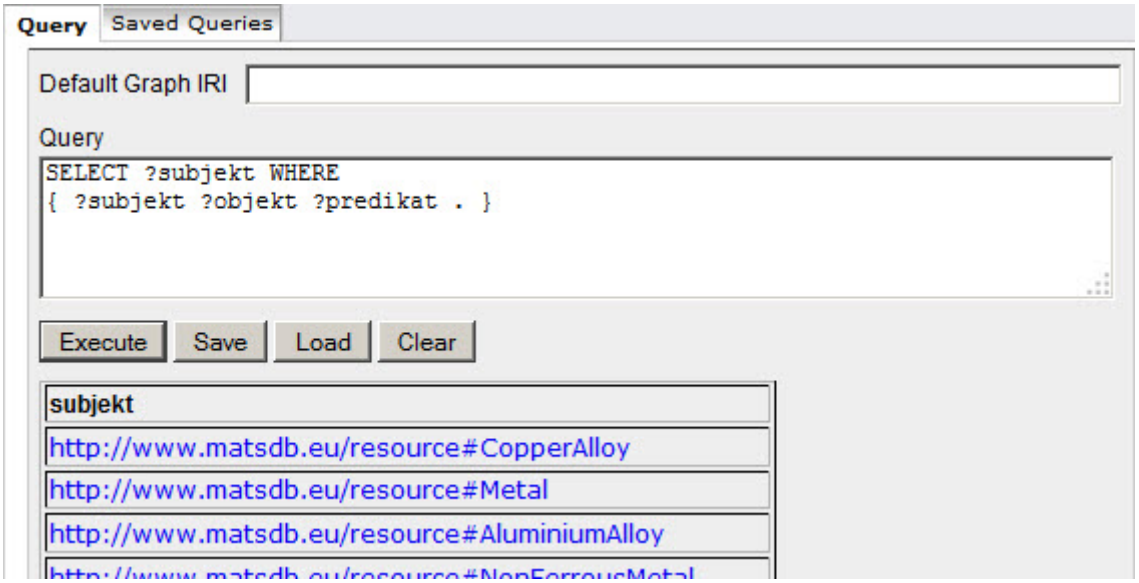
SELECT je naredba SPARQL jezika koja služi za pretragu RDF trojaca, a kao rezultat vraća pronađene rezultate u obliku liste, odnosno tablice podataka. Najjednostavniji upit koji je moguće postaviti prema bazi podataka bi bio dohvaćanje svih RDF trojaca iz nje.

Zvjezdica nakon naredbe SELECT označava da će biti ispisan sadržaj svih varijabli koje su postavljene pod klauzulom WHERE, odnosno sadržaja svih varijabli na mjestu subjekta, relacije i objekta u trojcima prema kojima se vrši pretraga. Rezultat je tablica koja sadržava sve klase, individue, relacije i doslovne vrijednosti u bazi, a nazivi stupaca tablice rezultata su nazivi varijabli. Slika 33. prikazuje SPARQL upit koji dohvaća sve RDF trojce iz baze podataka, te ispisuje sve njihove elemente.



Slika 33. - SPARQL upit i prikaz svih RDF trojaca u bazi podataka

Ako je potrebno prikazati vrijednosti spremljene samo u pojedine varijable tada se nakon SELECT naredbe, umjesto zvjezdice, navode imena varijabli čije se vrijednosti žele prikazati, odvojene praznim mjestom. Slika 34. prikazuje SPARQL upit koji dohvaća sve RDF trojce iz baze podataka, ali kao rezultat prikazuje samo njihove subjekte.



Slika 34. - SPARQL upit i prikaz subjekata RDF trojaca u bazi podataka

Ovi primjeri prikazuju ideju pretraživanja baze podataka pomoću SPARQL jezika kojom se postavljaju upiti za dohvaćanje željenih rezultata preko varijabli u uzorku trojaca.

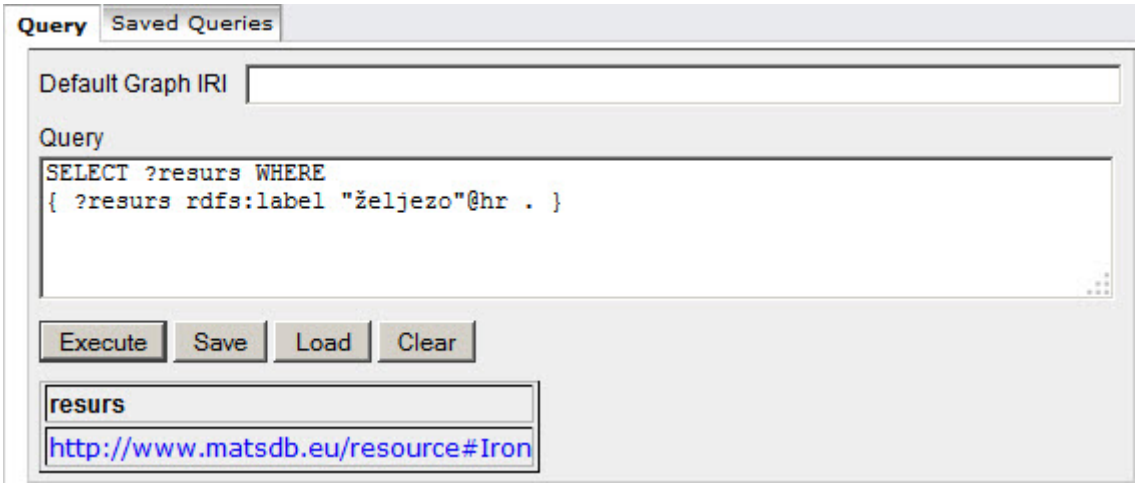
Kod pretrage baza podataka, obično se željeni URI resursa traži preko relacije *rdfs:label* i njezine doslovne vrijednosti. S obzirom da su doslovne vrijednosti uvijek na mjestu

objekta u RDF trojcu, a relacija *rdfs:label* u tom slučaju na mjestu relacije, tada se traži subjekt RDF trojca u bazi podataka.

Neka je za primjer potrebno naći URI resursa koji predstavlja kemijski element željeza. Slika 35. ilustrira RDF trojac koji je potrebno pronaći.

Slika 35. - Ilustracija RDF trojca za kojim se vrši pretraga

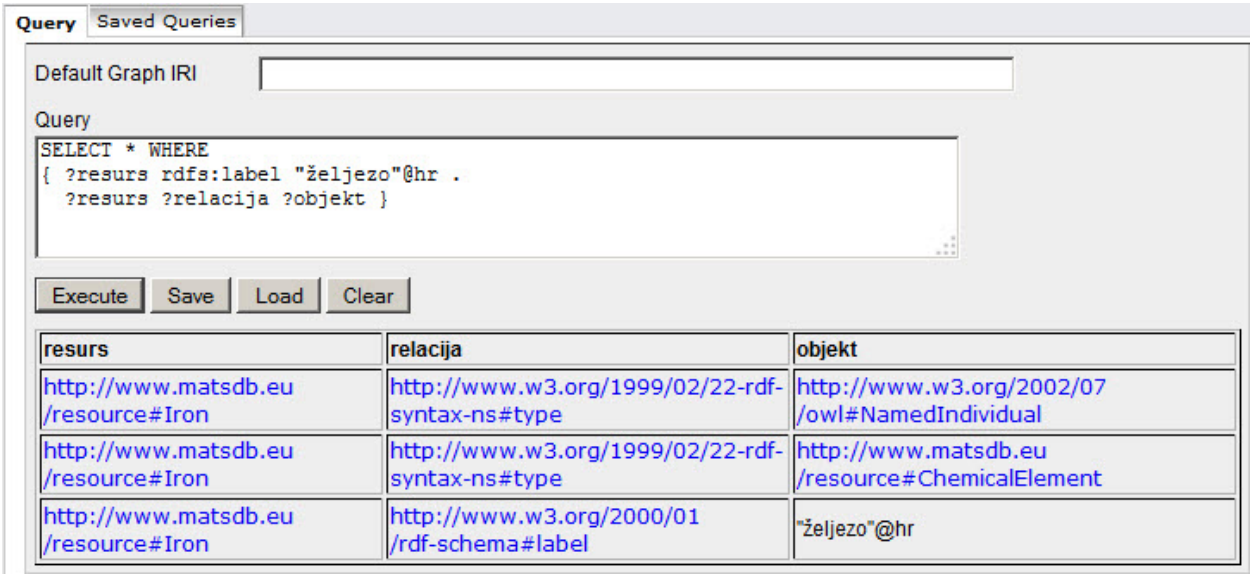
On je preko relacije *rdfs:label* vezan za doslovnu vrijednost *"željezo"@hr* i doslovnu vrijednost *"iron"@en*. Slika 36. prikazuje SELECT upit za dohvaćanje URI-a resursa koji predstavlja željezo, preko doslovne vrijednosti napisane na hrvatskom jeziku.



Slika 36. - SPARQL upit i prikaz resursa pronađenog preko doslovne vrijednosti

Iz ovoga se vidi da već pri izradi ontologije i samog pretraživanja je moguće izraditi višejezičnu aplikaciju, te omogućiti nadogradnju i proširenje u svakom trenutku dodajući relacije na drugim jezicima, bez izmjene programskog koda aplikacije.

Za razliku od prijašnjih upita, u varijabli *?resurs* postoji samo jedan URI resursa. Sada je moguće pretraživati sve klase, relacije, individue i doslovne vrijednosti s kojima je on povezan tako što se u novom redu, odnosno novom trojcu, na mjestu subjekta postavi varijabla u kojoj je prethodno spremljen URI resursa, odnosno varijabla *?resurs*, a na mjestu relacije i objekta se dodaju dvije nove varijable *?relacija* i *?objekt*. Slika 37. prikazuje upit i tablicu rezultata, odnosno relacija i objekata s kojima je URI resursa povezan.

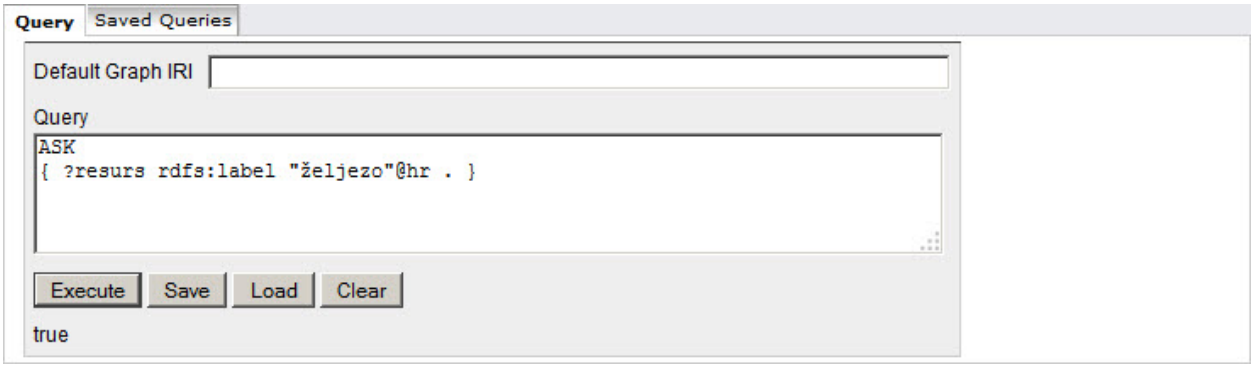


Slika 37. - SPARQL upit i prikaz povezanih podataka sa resursom željeza

Može se reći da je prvi red WHERE klauzule rezerviran za pronalaženje „unatrag“ željenog URI-a resursa koji će biti zapisan u varijablu *?resurs*, dok je drugi red rezerviran za pretraživanje „unaprijed“ i spremanje svih ostalih relacija i objekata u varijable *?relacija* i *?objekt*.

ASK naredba kao rezultat daje isključivo *true* ili *false* vrijednost. Ovu naredbu je moguće koristiti za upite kojima se želi provjeriti postoji li jedan ili više RDF trojaca u bazi podataka koji se poklapa sa uzorkom trojaca u ASK klauzuli.

Slika 38. prikazuje primjer ASK naredbe i postavlja upit da li postoji RDF trojac u bazi podataka koji je preko relacije *rdfs:label* vezan sa doslovnom vrijednosti *"željezo"@hr*.

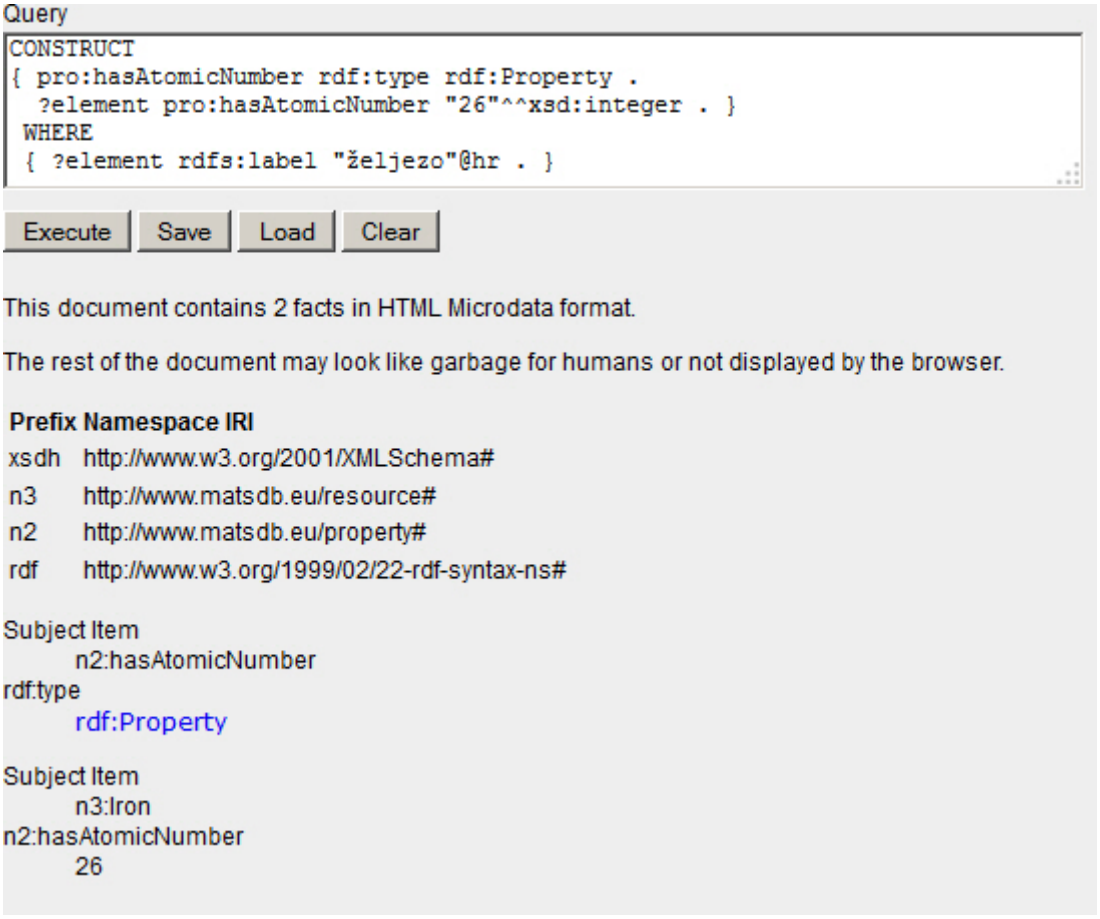


Slika 38. - SPARQL upit i rezultat ASK naredbe

CONSTRUCT naredbom je također moguće dohvatiti podatke iz baze, ali i kreirati nove RDF trojce. Zbog toga za razliku od SELECT naredbe, ona kao rezultat ima RDF trojce. Nove RDF trojce je moguće stvarati koristeći rezultate pretrage baze podataka, direktnim upisivanjem novih, te kombinacijom starih i novih.

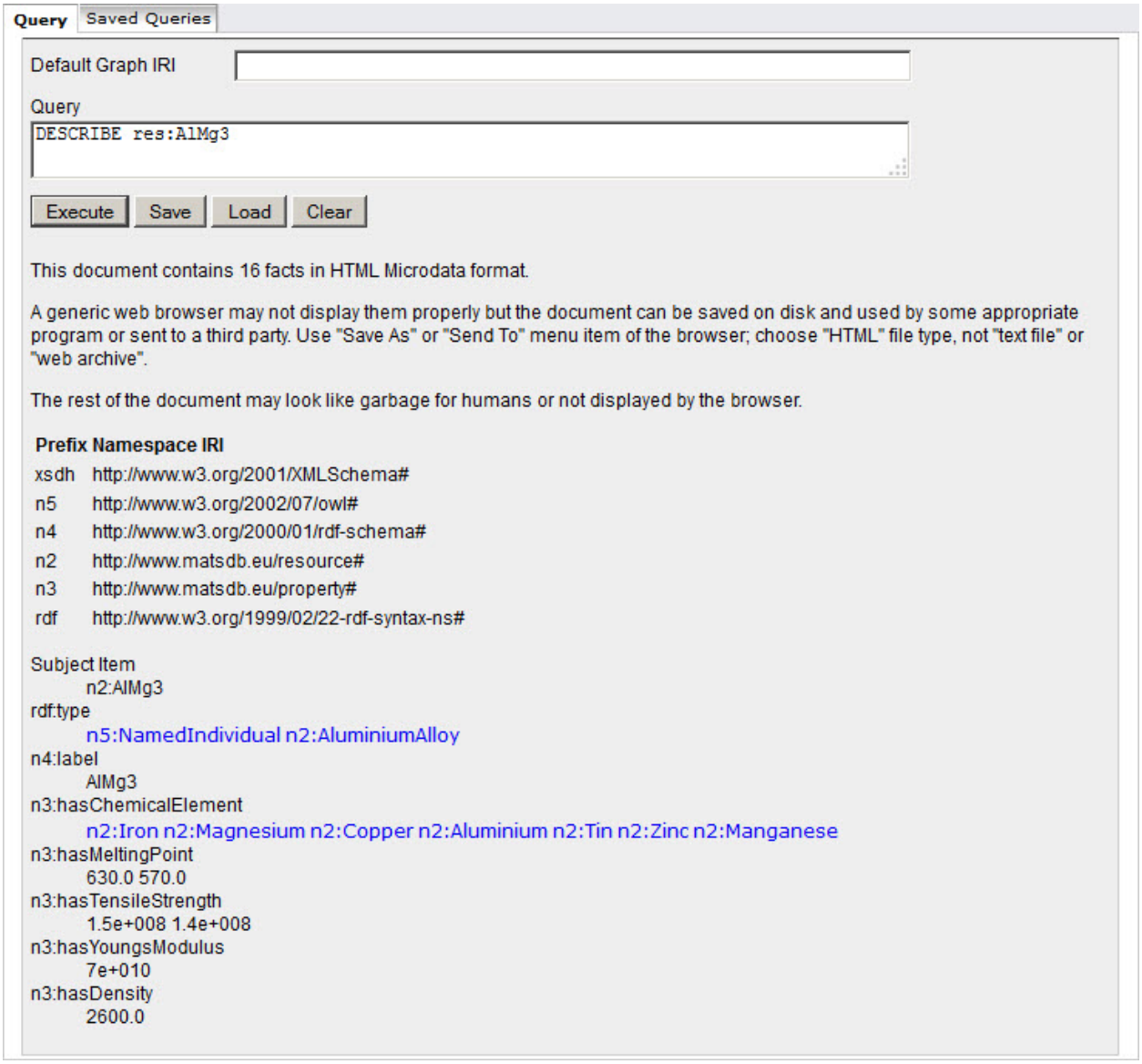
Treba napomenuti da ovom naredbom, kreirani RDF trojci se samo stvaraju kao rezultat, ali se ne upisuju u bazu podataka.

Slika 39. prikazuje primjer koji predstavlja CONSTRUCT upit. On u klauzuli WHERE traži URI resursa koji je preko relacije *rdfs:label* vezan sa doslovnom vrijednosti *"željezo"@hr*. Drugim riječima traži URI željeza. U CONSTRUCT klauzuli se deklarira nova relacija *pro:hasAtomicNumber* kao član klase *rdf:Property*. Nakon toga se stvara novi RDF trojac koji povezuje URI željeza spremljen u varijabli *?element* i doslovnu vrijednost *"26"^^xsd:integer* preko relacije *pro:hasAtomicNumber*.



Slika 39. - SPARQL upit i rezultat CONSTRUCT naredbe

DESCRIBE naredba služi za opis RDF grafa. Njeni rezultati nisu u svim programskim rješenjima jednaki, već ovise o serveru na kojem se SPARQL procesor i baza podataka nalaze tako da se ne koristi često, ali valja je spomenuti jer je jedna od osnovnih naredbi upita i u budućnosti će vjerojatno biti standardizirana, te će njeni upiti na svakom serveru davati iste rezultate. U slučaju OpenLink Virtuosa, ona stvara RDF graf željenog resursa. Slika 40. prikazuje primjer takve naredbe i njenih rezultata, tj. podataka vezanih uz aluminijevu leguru AlMg3, odnosno resurs *res:AlMg3*.



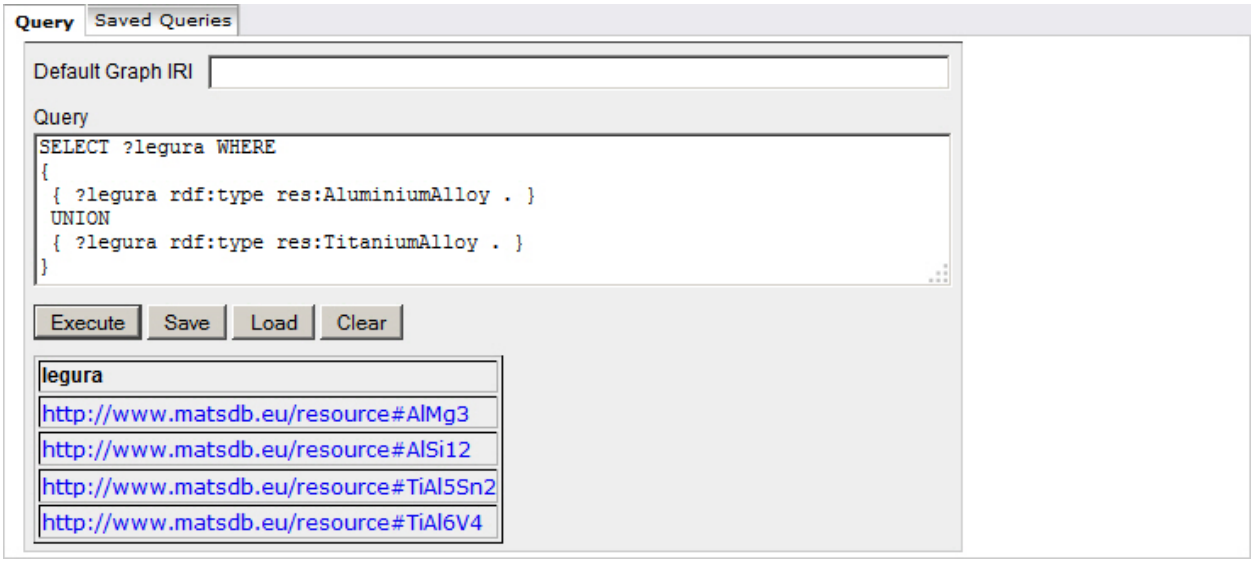
Slika 40. - SPARQL upit i rezultat DESCRIBE naredbe

3.1.2. Dodatni uvjeti upita [6]

Preko dodatnih uvjeta upita UNION i OPTIONAL moguće je pretražiti podatke iz dva različita RDF grafa, odnosno podatke koji mogu ali i ne moraju biti u bazi.

Naredbom UNION je moguće postaviti dvije ili više grupe trojaca od kojih samo jedna grupa mora zadovoljavati RDF trojce u bazi podataka kako bi rezultati bili pohranjeni u varijable. Svaka grupa trojaca za pretraživanje se piše unutar vitičastih zagrada. Grupe trojaca se vežu UNION naredbom. Sve zajedno se piše unutar WHERE klauzule.

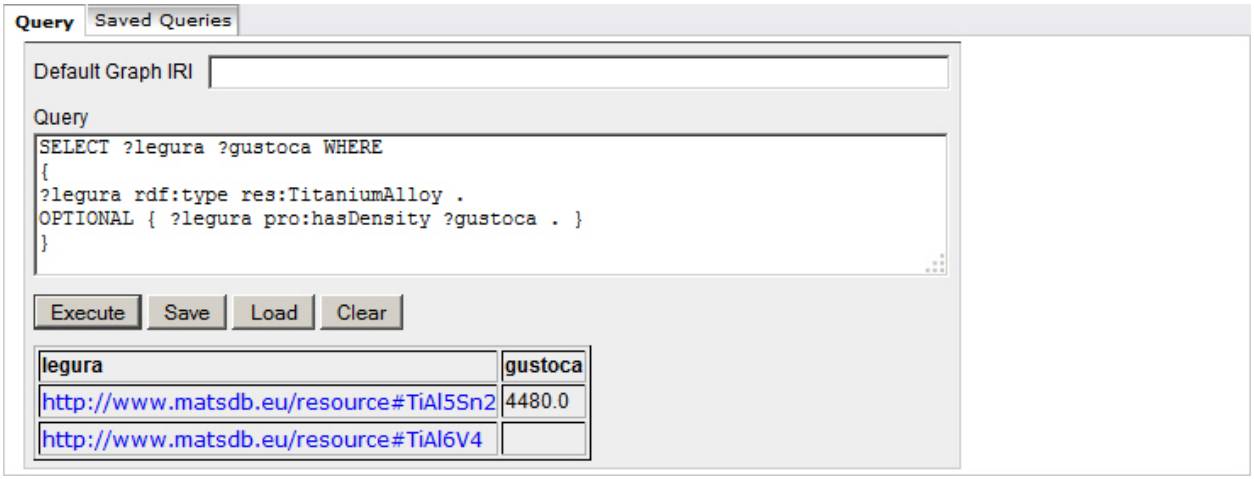
Tako je moguće pretražiti bazu te dohvatiti sve aluminijeve i titanove legure. Preko relacije *rdf:type* jedna grupa trojaca će pretraživati klasu *res:AluminiumAlloy*, a druga *res:TitaniumAlloy*. Slika 41. prikazuje taj upit i rezultate.



Slika 41. - SPARQL upit i rezultati UNION naredbe

Svi dosadašnji upiti pretražuju RDF trojce u bazi na način da njihove elemente spremaju u varijable. Ako dođe do situacije da određen resurs nema upisanu vrijednost preko neke relacije ili samu relaciju, odnosno jedan trojac iz grupe trojaca ne postoji u bazi podataka, tada ta grupa trojaca neće biti uzeta kao rezultat, odnosno pohranjena u varijable.

Naredbom OPTIONAL je moguće izbjeći taj problem. Tako na primjer određeni materijal ne mora imati upisanu gustoću, odnosno doslovnu vrijednost vezanu relacijom *pro:hasDensity*. Slika 42. prikazuje način kada je potrebno izvući sve titanove legure i njihove gustoće (ako postoje).



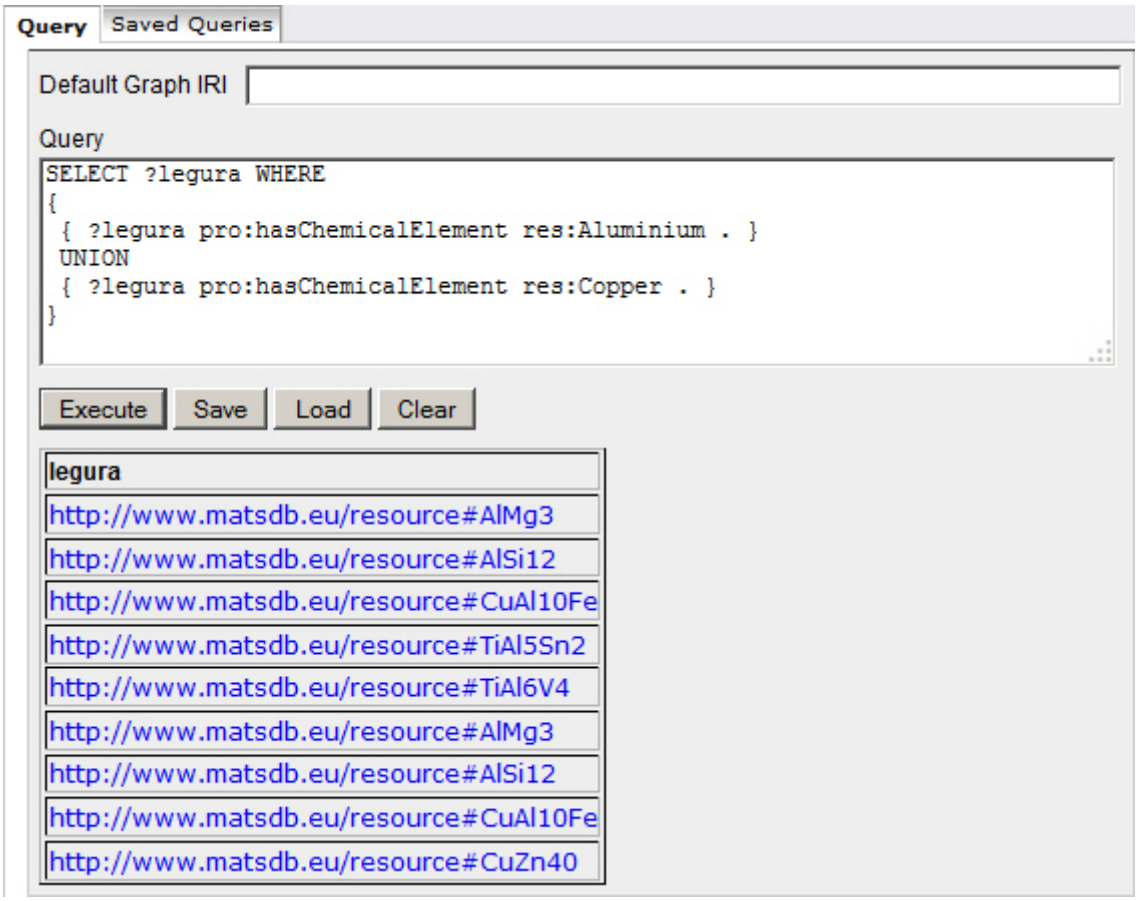
Slika 42. - SPARQL upit i rezultati OPTIONAL naredbe

3.1.3. Ograničavanje, sortiranje i grupiranje rezultata [6]

Za ograničavanje, sortiranje i grupiranje rezultata se u SPARQL jeziku koriste naredbe DISTINCT, ORDER BY, OFFSET, LIMIT i GROUP BY odnosno njihova kombinacija.

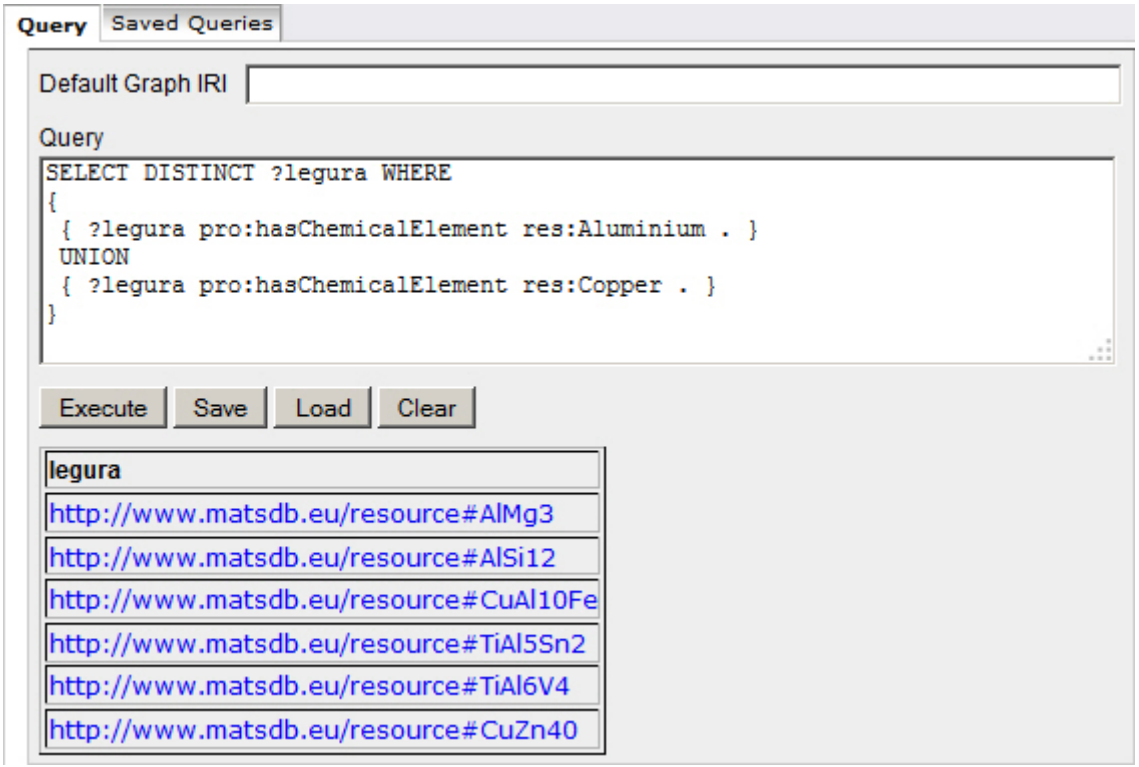
Naredba DISTINCT se piše odmah nakon SELECT naredbe, prije zvjezdice odnosno varijabli koje je potrebno prikazati. Ova naredba služi za uklanjanje ponovljenih rezultata.

Slika 43. prikazuje način gdje je potrebno pronaći legure u kojima se pojavljuju kemijski elementi aluminij ili bakar.



Slika 43. - SPARQL upit i rezultati za materijale sa aluminijom i bakrom

Zbog toga što neke legure sadrže i aluminij i bakar, odnosno zadovoljavaju obje grupe trojaca kod UNION naredbe, događa se da se po dva puta uzimaju kao rezultat. Kako bi se izbjeglo ponavljanje istih rezultata piše se naredba DISCTINCT. Slika 44. prikazuje takav upit sa rezultatima.



Slika 44. - SPARQL upit i rezultati DISTINCT naredbe

ORDER BY, OFFSET i LIMIT naredbe se pišu nakon WHERE klauzule na kraju SPARQL upita.

ORDER BY naredbom je moguće sortirati rezultate prema određenoj varijabli ili više njih. Moguće ih je sortirati prema abecedi i numerički. Također je rezultate moguće sortirati od veće vrijednosti prema manjoj pomoću riječi DESC, te od manje vrijednosti prema većoj pomoću riječi ASC. Slika 45. prikazuje takav primjer.

Query | Saved Queries

Default Graph IRI

Query

```
SELECT * WHERE
{ ?legura pro:hasMeltingPoint ?temperatura . }
ORDER BY ASC(?temperatura)
```

Execute | Save | Load | Clear

legura	temperatura
http://www.matsdb.eu/resource#AlMg3	570.0
http://www.matsdb.eu/resource#AlSi12	574.0
http://www.matsdb.eu/resource#AlSi12	582.0
http://www.matsdb.eu/resource#AlMg3	630.0
http://www.matsdb.eu/resource#CuAl10Fe	1040.0
http://www.matsdb.eu/resource#CuAl10Fe	1060.0
http://www.matsdb.eu/resource#TiAl5Sn2	1549.0
http://www.matsdb.eu/resource#TiAl5Sn2	1649.0
http://www.matsdb.eu/resource#TiAl6V4	1650.0

Slika 45. - SPARQL upit i rezultati ORDER BY naredbe

Naredbom OFFSET je moguće započeti pretragu pomaknutu od početka baze za određen broj RDF trojaca. Naredbom LIMIT je moguće ograničiti broj dohvaćenih rezultata. Kombinacijom tih dviju naredbi moguće je izvući sve rezultate između određenih RDF trojaca. Slika 46. prikazuje primjer gdje su izvučeni četvrti i peti RDF trojac.

Query | Saved Queries

Default Graph IRI

Query

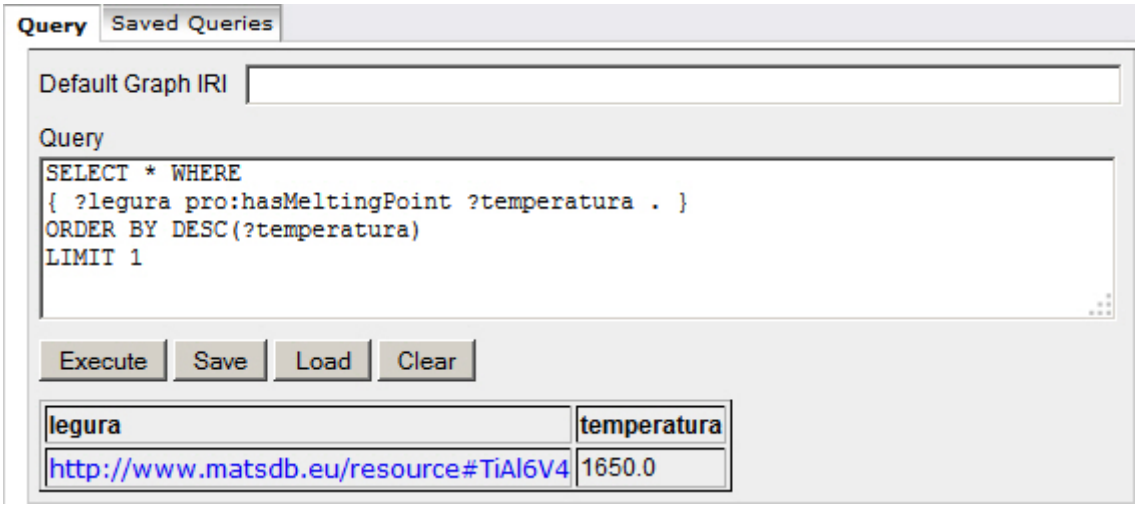
```
SELECT * WHERE
{ ?legura pro:hasMeltingPoint ?temperatura . }
ORDER BY ASC(?temperatura)
OFFSET 3
LIMIT 2
```

Execute | Save | Load | Clear

legura	temperatura
http://www.matsdb.eu/resource#AlMg3	630.0
http://www.matsdb.eu/resource#CuAl10Fe	1040.0

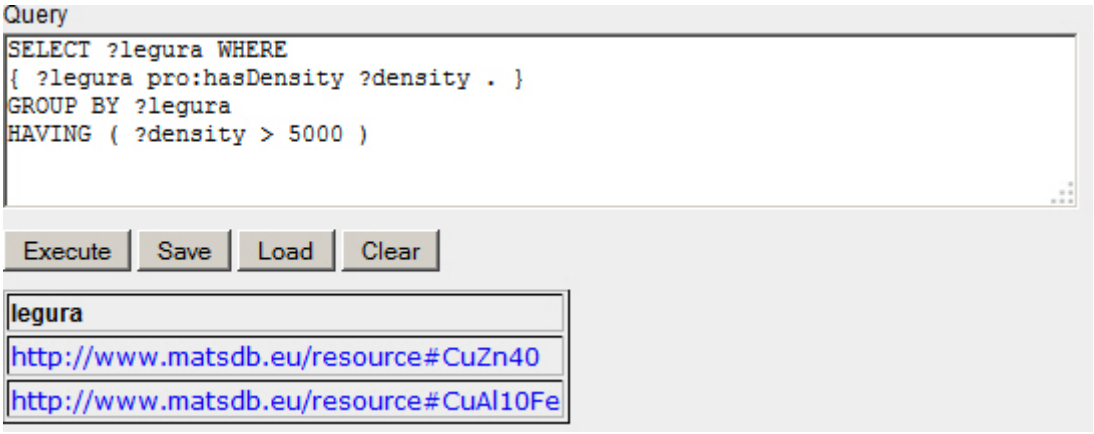
Slika 46. - SPARQL upit i rezultati OFFSET i LIMIT naredbi

Kombinacijom naredbe ORDER BY i LIMIT je moguće pronaći najveću ili najmanju vrijednost neke varijable tako što se rezultate poredaju uzlazno ili silazno, te se ograniče na jedan rezultat. Slika 47. prikazuje takav primjer.



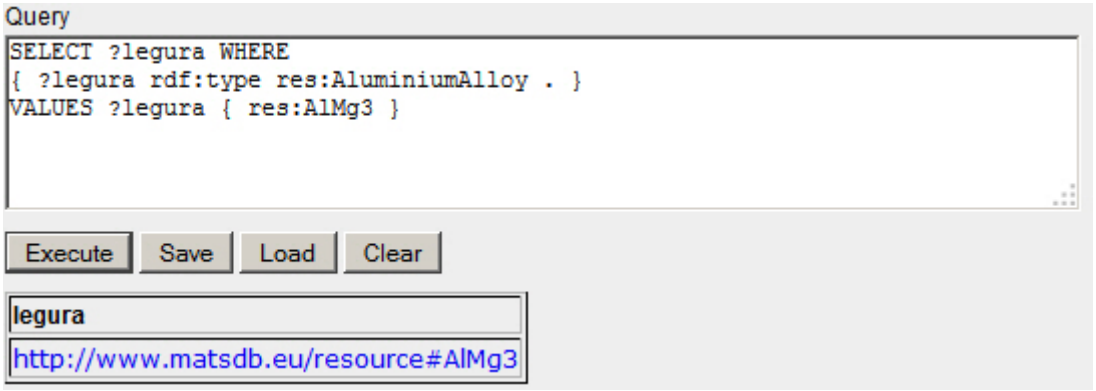
Slika 47. - SPARQL upit i rezultati ORDER BY i LIMIT naredbe

HAVING naredbom u kombinaciji sa GROUP BY naredbom moguće je filtrirati rezultate. Slika 48. prikazuje primjer upita koji pronalazi sve materijale sa gustoćom većom od 5000 kg/m^3 preko ove dvije naredbe.



Slika 48. - SPARQL upit i rezultati HAVING naredbe

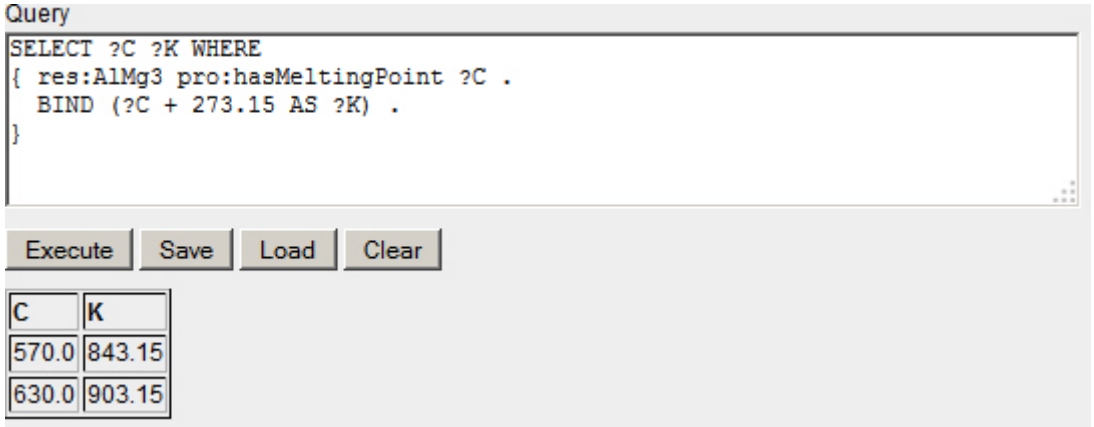
VALUES naredbom je moguće također filtrirati rezultate na način da se navedu vrijednosti iz varijable koje će se zadržati u rezultatima, dok se ostale odbacuju. Slika 49. prikazuje primjer u kojem se zadržava samo aluminijeva legura *res:AlMg3*.



Slika 49. - SPARQL upit i rezultati VALUES naredbe

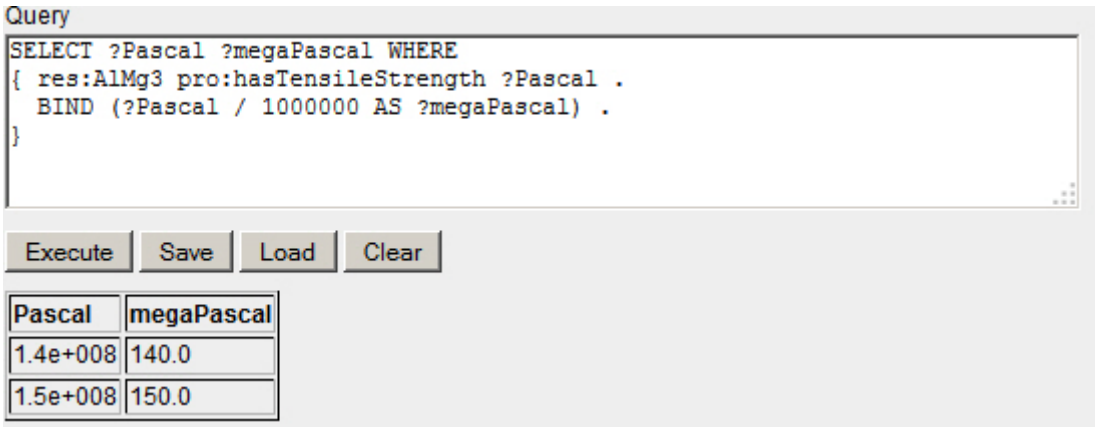
3.1.4. Pridruživanje vrijednosti varijablama [6]

Naredbom BIND se pridružuju doslovne vrijednosti ili vrijednosti spremljene u varijablama u nove varijable. U bazi podataka, doslovne vrijednosti temperatura tališta su zapisane u stupnjevima Celzijevim. Ako je potrebno pretvoriti te vrijednosti u Kelvine, tada je moguće preko BIND naredbe zbrojiti vrijednost u varijabli sa 273,15, te novu vrijednost spremiti u novu varijablu *?Kelvin*. Slika 50. prikazuje korištenje naredbe BIND.



Slika 50. - SPARQL upit i rezultati BIND naredbe

Na sličan način je moguće pronaći vrijednosti vlačnih čvrstoća koje u bazi podataka imaju mjernu jedinicu Pa, te ih podijeliti sa milijun i spremiti u novu varijablu kao MPa. Slika 51. prikazuje taj slučaj.



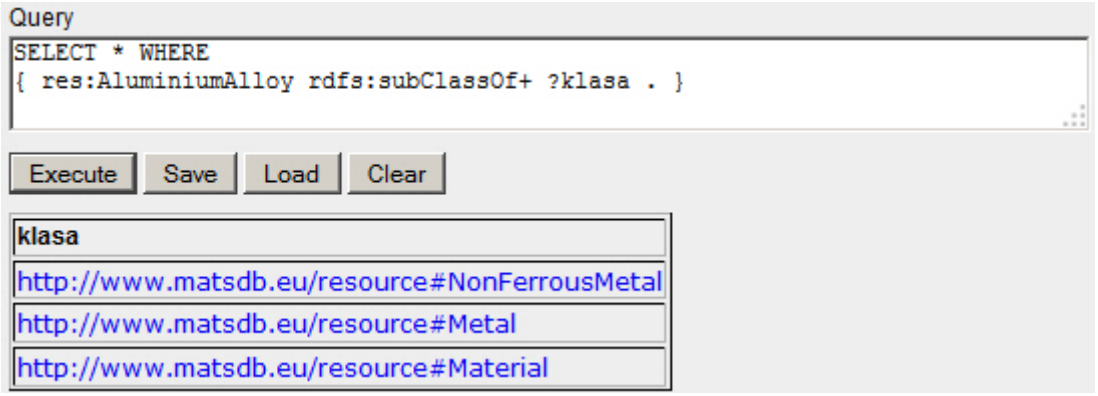
Slika 51. - SPARQL upit i rezultati BIND naredbe 2

.

3.1.5. Putanje relacija [7]

Putanje relacija služe za pronalaženje elemenata RDF trojaca u RDF grafu koji su kaskadno vezani relacijama.

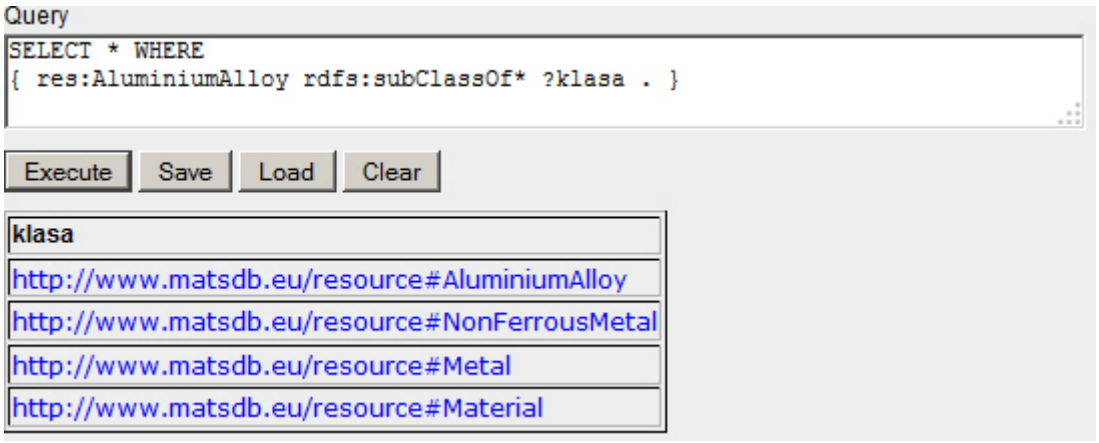
Klasa *res:AluminiumAlloy* je podklasa klase *res:NonFerrousMetal* koja je opet podklasa klase *res:Metal,* a ona je podklasa klase *res:Material*. Sve one su vezane relacijom *rdfs:subClassOf*. Preko putanje je moguće pronaći sve elemente, odnosno klase čijih je klasa *res:AluminiumAlloy* podklasa na način koji prikazuje SLIKA.



Slika 52. - SPARQL putanje

Ovdje znak plus „+“ stoji uz relaciju i govori SPARQL procesoru da traži jedan ili više elemenata kaskadno povezanih preko relacije *rdfs:subClassOf* počevši od subjekta *res:AluminiumAlloy*. U ovom slučaju će pronaći sve klase odnosno podklase kojih je *res:AluminiumAlloy* član.

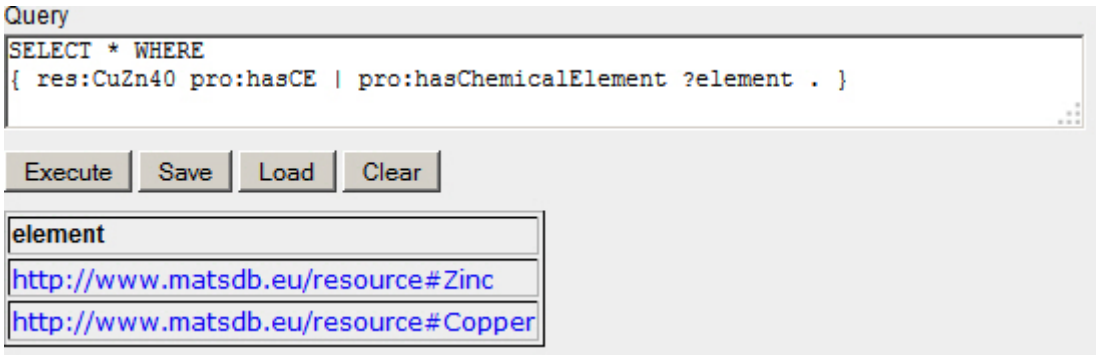
Znak zvjezdica „*“ također pretražuje do zadnjeg pojavljivanja navedene relacije u kaskadi, ali za razliku od prethodnog znaka, kreće od nultog resursa. Drugim riječima uzima i resurs na mjestu subjekta. Slika 53. prikazuje takav primjer.



Slika 53. - SPARQL putanje 2

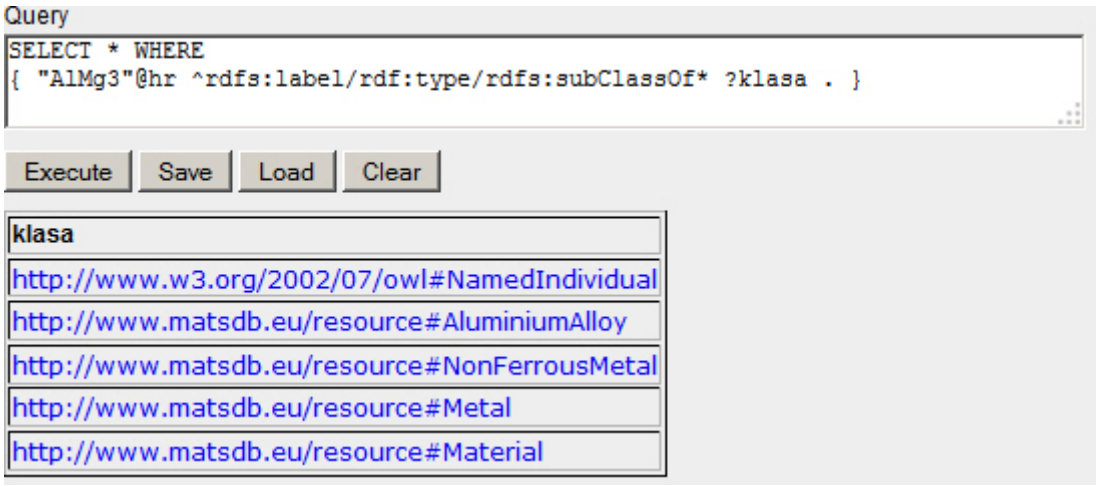
Putanjama se mogu pretraživati i različite, odnosno alternativne relacije.

Neka je resurs *res:CuZn40* vezan preko jedne od relacija *pro:hasCE* odnosno *pro:hasChemicalElement* sa drugim resursima. U slučaju da nije poznato sa kojom su relacijom točno resursi vezani, moguće je postaviti SPARQL upit gdje znak „|“ govori SPARQL procesoru da pronađe sve elemente koji su vezani ili sa prvom ili sa drugom relacijom. Slika 54. prikazuje takav primjer.



Slika 54. - SPARQL putanje 3

Putanje se mogu kombinirati i sa različitim relacijama. Tako je moguće preko relacije *rdfs:label* pronaći individuu, preko relacije *rdf:type* pronaći klasu kojoj ona pripada, a preko *rdfs:subClassOf* sve ostale klase kojih je ona podklasa. Slika 55. prikazuje takav primjer.



Slika 55. - SPARQL putanje 4

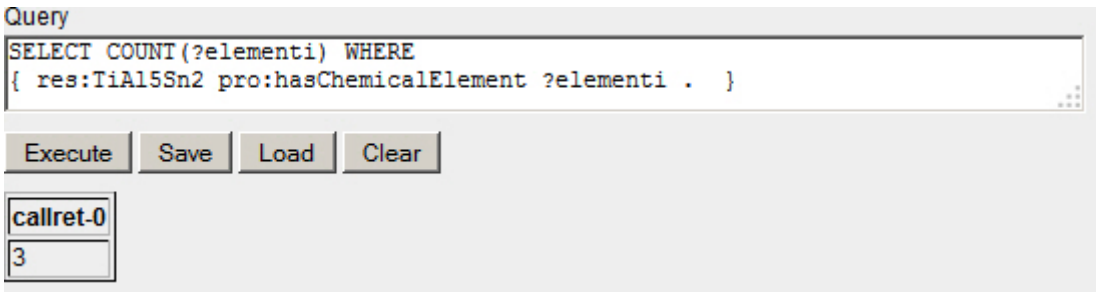
Treba istaknuti da je u ovom primjeru doslovna vrijednost zapisana na mjestu subjekta, a to je zbog toga što kod relacije *rdfs:label* stoji znak ^ koji označava inverznu pretragu, odnosno pretragu unatrag.

3.1.6. Algebra grupe rezultata [6]

Naredba COUNT služi za prebrojavanje rezultata spremljenih u varijabli, a naredbe SUM, MIN, MAX i AVG služe za zbrajanje, nalaženje minimuma, maksimuma i srednje vrijednosti nad brojčanim vrijednostima. One se pišu iza naredbe SELECT uz varijable koje se žele prikazati u rezultatima.

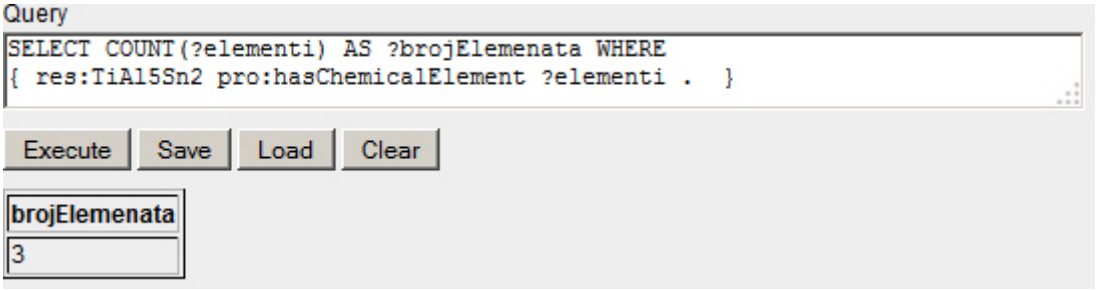
Ove naredbe se koriste sa naredbom GROUP BY pomoću koje se vrše navedene operacije nad pojedinim grupama rezultata.

Neka je potrebno prebrojati koliko se kemijskih elemenata nalazi u titanovoj leguri *res:TiAl5Sn.* To je moguće napraviti tako što se prvo pronađu svi elementi preko relacije *pro:hasChemicalElement*, te se spreme u varijablu *?elementi*. Tada pomoću naredbe COUNT se prebroje svi rezultati spremljeni u varijablu *?elementi.* Slika 56. prikazuje taj primjer.



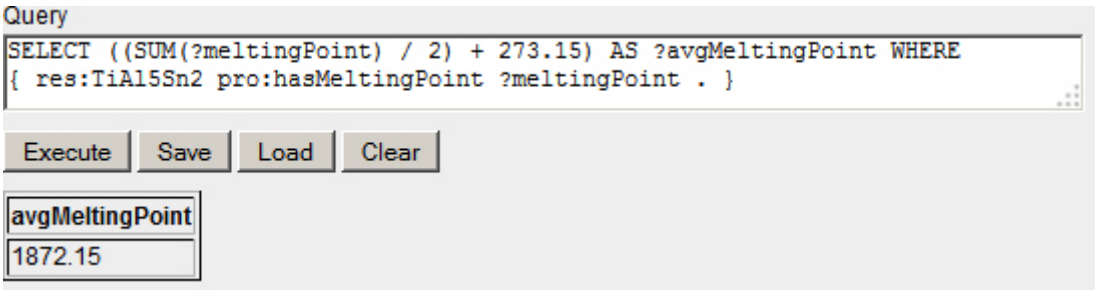
Slika 56. - SPARQL upit i rezultati COUNT naredbe

Vidljivo je da naziv stupca više nije ime varijable odnosno *?elementi*, već je vrijednost koju je izračunala naredba COUNT spremljena u nedeklariranoj varijabli *callret-0*. Kako bi se to izbjeglo i omogućilo daljnje korištenje same varijable, moguće je izračunatu vrijednost dodijeliti u novu varijablu *?brojElemenata.* Slika 57. prikazuje taj slučaj.



Slika 57. - SPARQL dodjeljivanje vrijednosti novoj varijabli

Neka titanova legura *res:TiAl5Sn2* sadrži minimalnu i maksimalnu temperaturu taljenja u stupnjevima Celzijevima. Pomoću naredbe SUM je moguće zbrojiti te dvije temperature, a zatim je podijeliti sa brojem temperatura kako bi se dobila srednja temperatura. Uz to je moguće zbrojiti 273.15 kako bi se srednja temperatura izrazila u Kelvinima. Slika 58. prikazuje navedeni primjer.



Slika 58. - SPARQL upit i rezultati SUM naredbe

Nalaženje srednje vrijednosti od vrijednosti pohranjenih u varijabli je moguće na jednostavniji način naredbom AVG. Slika 59. prikazuje način za pronalaženje srednje vrijednosti temperature taljenja svih legura.

Query

```
SELECT ?legura (AVG(?meltingPoint)) AS ?avgMeltingPoint WHERE
{ ?legura pro:hasMeltingPoint ?meltingPoint . }
```

Execute Save Load Clear

legura	avgMeltingPoint
http://www.matsdb.eu/resource#TiAl5Sn2	1599
http://www.matsdb.eu/resource#AlMg3	600
http://www.matsdb.eu/resource#AlSi12	578
http://www.matsdb.eu/resource#TiAl6V4	1650
http://www.matsdb.eu/resource#CuAl10Fe	1050

Slika 59. - SPARQL upit i rezultati AVG naredbe

Naredbama MIN i MAX je moguće pronaći najmanju i najveću vrijednost pohranjenu u nekoj varijabli. Slika 60. prikazuje primjer u kojem su pronađene minimalne i maksimalne temperature taljenja pojedinih legura.

Query

```
SELECT ?legura (MIN(?mp) AS ?minMP) (MAX(?mp) AS ?maxMP) WHERE
{ ?legura pro:hasMeltingPoint ?mp . }
```

Execute Save Load Clear

legura	minMP	maxMP
http://www.matsdb.eu/resource#TiAl5Sn2	1549.0	1649.0
http://www.matsdb.eu/resource#AlMg3	570.0	630.0
http://www.matsdb.eu/resource#AlSi12	574.0	582.0
http://www.matsdb.eu/resource#TiAl6V4	1650.0	1650.0
http://www.matsdb.eu/resource#CuAl10Fe	1040.0	1060.0

Slika 60. - SPARQL upit i rezultati MIN i MAX naredbi

3.1.7. Ostale SPARQL funkcije [8]

U SPARQL jeziku postoje funkcije za manipulaciju nad znakovima, brojevima, vremenom, datumom i jezikom, te za provjeru RDF podataka. Tablica 1., tablica 2., tablica 3. i tablica 4. prikazuju listu tih funkcija, i navode njihov kratak opis.

Tablica 1. - SPARQL numeričke funkcije

Numeričke	abs	Vraća apsolutnu vrijednost broja
	round	Zaokružuje decimalan broj na najbližu cjelobrojnu vrijednost
	ceil	Zaokružuje decimalan broj na višu cjelobrojnu vrijednost
	floor	Zaokružuje decimalan broj na nižu cjelobrojnu vrijednost
	rand	Vraća nasumičan decimalan broj

Tablica 2. - SPARQL znakovne funkcije

Znakovne	strlen	Vraća broj znakova u znakovnom nizu
	substr	Vraća dio znakovnog niza
	ucase	Vraća znakovni niz zapisan velikim slovima
	lcase	Vraća znakovni niz zapisan malim slovima
	strstarts	Provjerava počinje li znakovni niz drugim nizom
	strends	Provjerava završava li znakovni niz drugim nizom
	contains	Provjerava postoji li znakovni niz unutar drugog niza
	strbefore	Vraća dio znakovnog niza prije drugog niza
	strafter	Vraća dio znakovnog niza nakon drugog niza
	encode_for_uri	Vraća znakovni niz kompatibilan sa URI ograničenjima
	concat	Spaja dva znakovna niza u jedan
	langMatches	Provjerava jezik znakovnog niza
	regex	Uspoređuje znakovni niz sa zadanim uzorkom
	replace	Zamjenjuje znakovni niz unutar drugog niza

Tablica 3. - SPARQL datumske i vremenske funkcije

Datumske i vremenske	now	Vraća trenutno datum i vrijeme
	year	Vraća godinu iz zadane xsd:dateTime vrijednosti
	month	Vraća mjesec iz zadane xsd:dateTime vrijednosti
	day	Vraća dan iz zadane xsd:dateTime vrijednosti
	hours	Vraća sate iz zadane xsd:dateTime vrijednosti
	minutes	Vraća minute iz zadane xsd:dateTime vrijednosti
	seconds	Vraća sekunde iz zadane xsd:dateTime vrijednosti
	timezone	Vraća vremensku zonu kao xsd:dayTimeDuration vrijednost
	tz	Vraća vremensku zonu kao doslovnu vrijednost

Tablica 4. - SPARQL RDF funkcije

RDF	isIRI	Provjerava da li je vrijednost IRI
	isBlank	Provjerava da li je vrijednost prazan čvor
	isLiteral	Provjerava da li je vrijednost doslovna vrijednost
	isNumeric	Provjerava da li je vrijednost numerička vrijednost
	str	Vraća vrijednost kao znakovnu vrijednost
	lang	Vraća oznaku jezika doslovne vrijednosti
	datatype	Vraća tip doslovne vrijednosti
	iri	Stvara IRI vrijednost iz zadanog znakovnog niza
	bnode	Stvara prazan čvor različit od ostalih
	strdt	Dodjeljuje tip vrijednosti doslovnoj vrijednosti
	strlang	Dodjeljuje oznaku jezika doslovnoj vrijednosti

3.1.8. Filtriranje rezultata [6]

Naredba FILTER se u SPARQL jeziku koristi za odbacivanje rezultata koji ne zadovoljavaju uvjet postavljen unutar same naredbe. Ona se piše unutar grupe trojaca na koju utječe. Moguće je postaviti više FILTER naredbi unutar jedne grupe trojaca kao i postaviti više uvjeta unutar jedne FILTER naredbe.

Slika 61. prikazuje primjer u kojem je dan jednostavan upit koji pronalazi vlačne čvrstoće legura. Ovdje FILTER naredba ima ulogu zadržavanja samo onih rezultata koji imaju vrijednost pohranjenu u varijablu *?tensileStrength* veću od 700000000, dok ostale odbacuje.

Query

```
SELECT ?legura ?tensileStrength WHERE
{ ?legura pro:hasTensileStrength ?tensileStrength .
  FILTER (?tensileStrength > 700000000) }
```

Execute Save Load Clear

legura	tensileStrength
http://www.matsdb.eu/resource#TiAl5Sn2	8.2e+008
http://www.matsdb.eu/resource#TiAl6V4	1.15e+009
http://www.matsdb.eu/resource#TiAl5Sn2	1.1e+011

Slika 61. - SPARQL upit i rezultati FILTER naredbe

Unutar nje je moguće koristiti funkcije za manipulaciju nad znakovnim, brojčanim, vremenskim i datumskim vrijednostima, za manipulaciju nad jezikom teksta, te za provjeru RDF tipova podataka.

BOUND naredbu unutar FILTER naredbe je moguće koristiti za provjeru da li varijabla sadržava ijednu vrijednost. Tako je moguće dohvatiti sve titanove legure koje nemaju upisanu gustoću, odnosno nemaju upisanu doslovnu vrijednost vezanu relacijom *pro:hasDensity.* Slika 62. prikazuje takav upit. Prvo se traže sve titanove legure i njihove gustoće naredbom OPTIONAL kako bi bile vraćene i legure koje ne sadrže tu vrijednost. Uskličnik prije naredbe BOUND označava negaciju te naredbe, odnosno odbacuje rezultate koji imaju zapisanu vrijednost u varijabli *?density*.

Query

```
SELECT * WHERE
{ ?legura rdf:type res:TitaniumAlloy .
OPTIONAL {?legura pro:hasDensity ?density }
FILTER (!bound(?density)) }
```

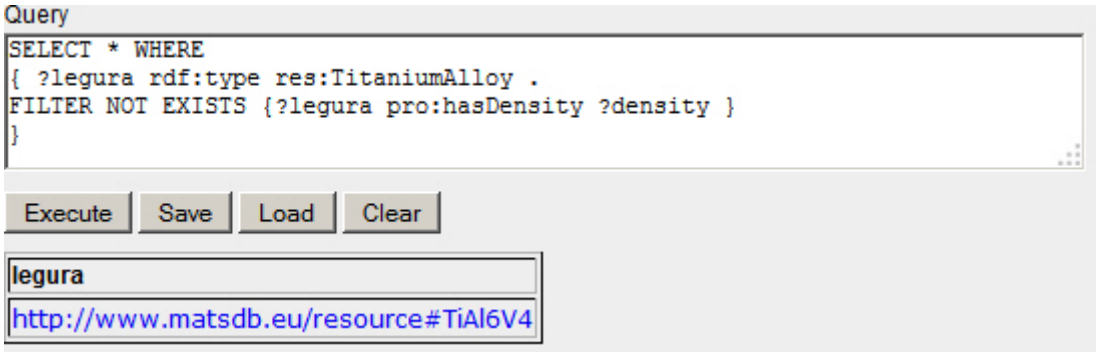
Execute Save Load Clear

legura	density
http://www.matsdb.eu/resource#TiAl6V4	

Slika 62. - SPARQL upit i rezultati BOUND naredbe

Dodatnim uvjetima FILTER naredbe EXISTS i NOT EXISTS moguće je zadržati samo RDF trojce u rezultatima koji se podudaraju, odnosno ne podudaraju sa trojcima unutar FILTER naredbe.

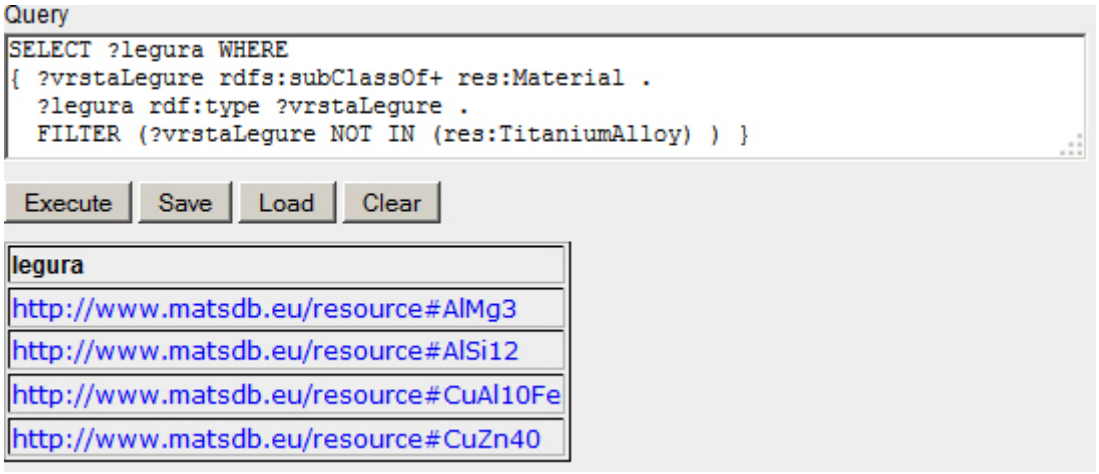
Slika 63. prikazuje prethodni primjer postavljen na drugi način u kojem će biti dohvaćeni samo URI-i titanovih legura koje ne sadrže doslovnu vrijednost vezanu preko relacije *pro:hasDensity*.



Slika 63. - SPARQL upit i rezultati FILTER NOT EXISTS naredbe

FILTER naredba koristiti i dodatne uvjete IN i NOT IN. Preko njih je moguće provjeriti da li je vrijednost u varijabli sadržana u nizu vrijednosti.

Neka je potrebno izvući sve materijale zapisane u bazi osim titanovih legura. Prvo se dohvate svi materijali, a nakon toga pomoću naredbe FILTER i NOT IN se ostave samo oni materijali koji na mjestu varijable *?vrstaLegure* ne sadrže resurs *res:TitaniumAlloy* kao što prikazuje. Slika 64. prikazuje taj upit.



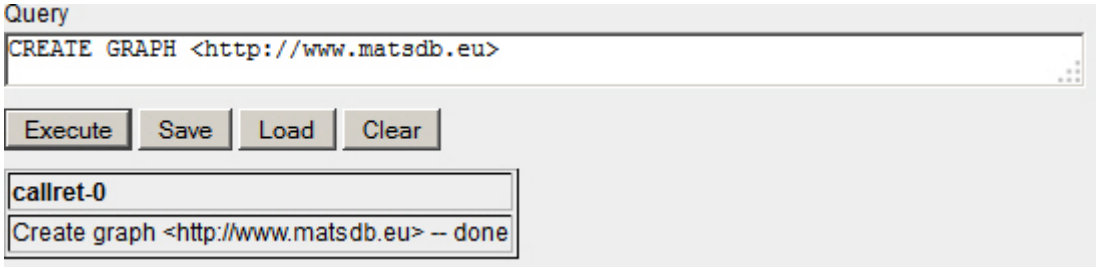
Slika 64. - SPARQL upit i rezultati FILTER NOT IN naredbe

3.1.9. Manipulacija podataka [6]

U verziji SPARQL 1.1 dodana je mogućnost umetanja, brisanja i ažuriranja RDF trojaca u bazi podataka direktno preko SPARQL upita.

Za umetanje novih RDF trojaca služe naredbe INSERT, INSERT DATA i LOAD, a za kreiranje RDF grafa naredba CREATE. [6]

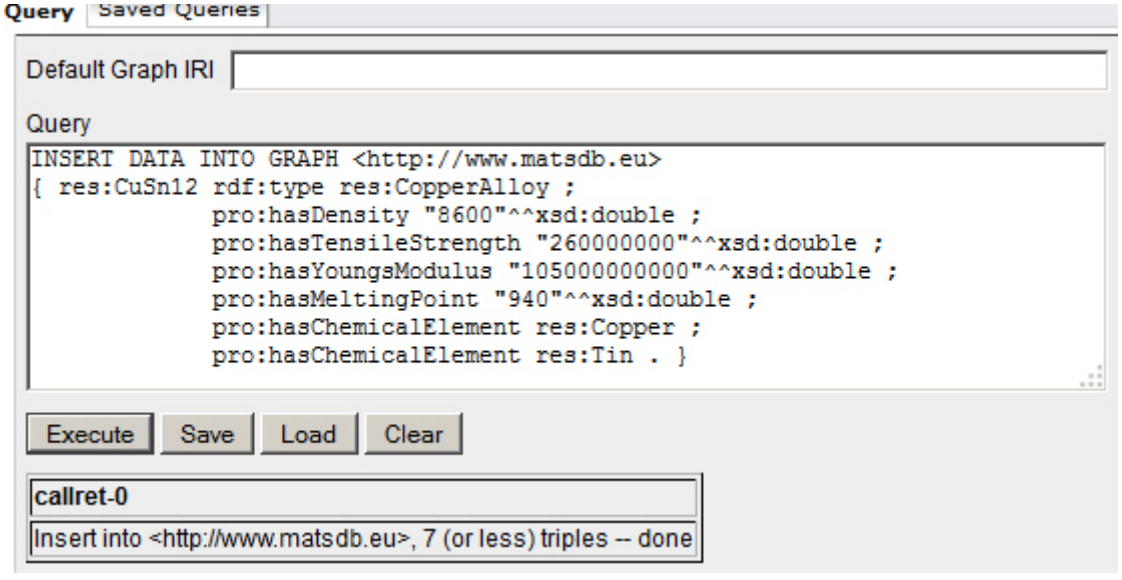
Naredbom CREATE GRAPH je moguće kreirati novi RDF graf. Slika 65. prikazuje kreiranje novog grafa.



Slika 65. - SPARQL upit i rezultati CREATE GRAPH naredbe

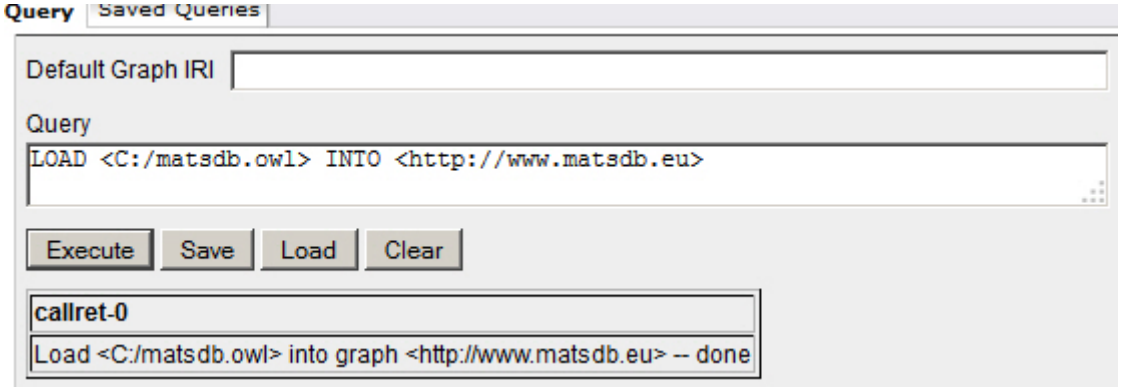
Naredba INSERT sadrži i WHERE klauzulu te je slična CONSTRUCT naredbi. Moguće je prvo pronaći RDF trojce u bazi podataka prema uzorku u WHERE klauzuli, a nakon toga stvoriti i ubaciti nove, odnosno kombinaciju elemenata novih i postojećih, navedenih u INSERT klauzuli.

Za razliku od nje, INSERT DATA nema WHERE klauzulu, već se u njenom dijelu upisuju RDF trojci koji se direktno ubacuju u bazu podataka, bez varijabli. Pomoću riječi INTO GRAPH se određuje graf u koji se RDF trojci ubacuju. Slika 67. prikazuje kreiranje novih RDF trojaca, odnosno nove bakrove legure sa odgovarajućim svojstvima vezanim preko već korištenih relacija.



Slika 66. - SPARQL upit i rezultati INSERT naredbe

Naredbom LOAD je preko SPARQL upita moguće ubaciti sve RDF trojce pohranjene u datoteci u bazu podataka. Slika 67. prikazuje ubacivanje datoteke *matsdb.owl* pohranjene na lokalnom računalu. Na isti način je moguće ubaciti RDF trojce iz datoteka koje se nalaze na drugim serverima u obliku datoteke.



Slika 67. - SPARQL upit i rezultati LOAD naredbe

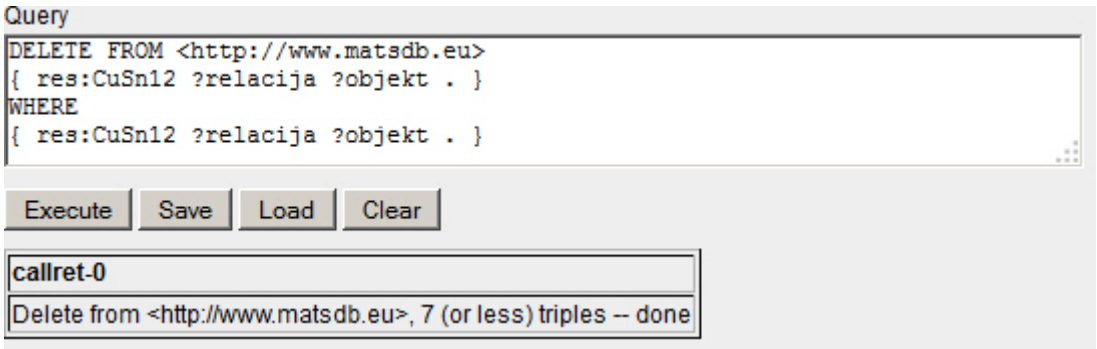
Za brisanje RDF trojaca i grafova iz baze podataka služe naredbe DELETE, DELETE DATA i CLEAR.

DELETE naredba kao i INSERT sadrži WHERE klauzulu u kojoj je moguće postaviti dodatne setove trojaca na temelju kojih će RDF trojci u bazi biti prvo pronađeni, a zatim obrisani.

DELETE DATA naredba nema WHERE klauzulu, već se unutar vitičastih zagrada upisuju točno oni RDF trojci koji će biti obrisani, bez varijabli.

Uz ove dvije naredbe pomoću riječi FROM se definira graf iz kojeg će RDF trojci biti obrisani.

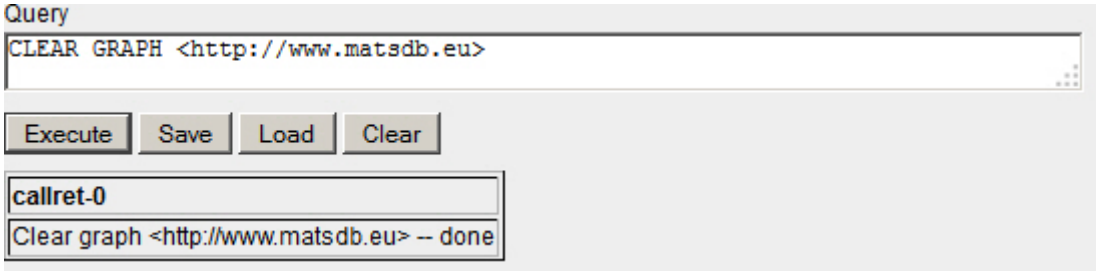
Slika 68. prikazuje primjer naredbe DELETE upita koji briše prethodno kreirane RDF trojce. Prvo pomoću WHERE klauzule pronalazi sve RDF trojce vezane sa bakrovom legurom *res:CuSn12*, a nakon toga ih sve briše.



Slika 68. - SPARQL upit i rezultati DELETE naredbe

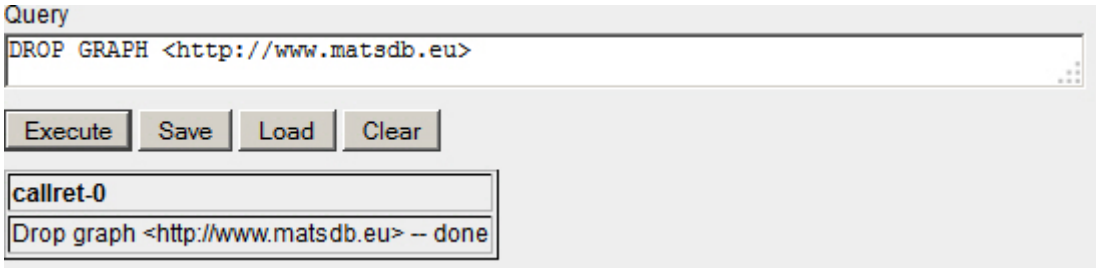
Naredbom CLEAR je moguće obrisati sve RDF trojce nekog grafa.

Slika 69. prikazuje brisanje svih RDF trojaca iz grafa *http://www.matsdb.eu*.



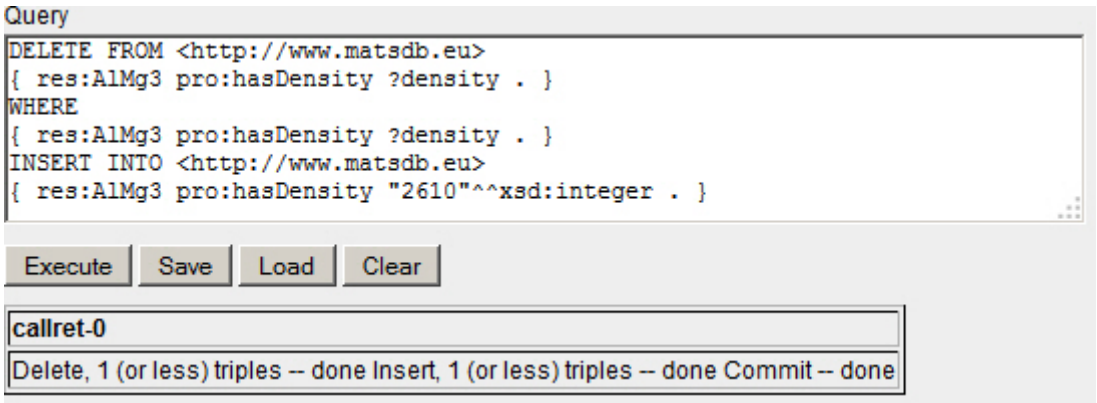
Slika 69. - SPARQL upit i rezultati CLEAR naredbe

Naredbom DROP je moguće obrisati sam RDF graf uključujući i RDF trojce u njemu. Slika 70. prikazuje taj upit.



Slika 70. - SPARQL upit i rezultati DROP naredbe

Za ažuriranje podataka u SPARQL jeziku ne postoji zasebna naredba koja je uvedena kao standard, već se odvija kombinacijom DELETE, INSERT i WHERE naredbi. Naredba WHERE je zadužena za pronalaženje RDF trojaca u bazi podataka koje treba izmijeniti. Naredbom DELETE se brišu postojeći RDF trojci iz baze podataka, ali ostaju pohranjeni u varijablama. Naredbom INSERT se odmah nakon toga kreiraju novi RDF trojci kombinacijom onih iz memorije, odnosno varijabli, i novih podataka te se spremaju u bazu podataka. Slika 71. prikazuje primjer izmjene vrijednosti gustoće aluminijeve legure *res:AlMg3*.



Slika 71. - SPARQL kombinacija naredbi za ažuriranje RDF trojaca

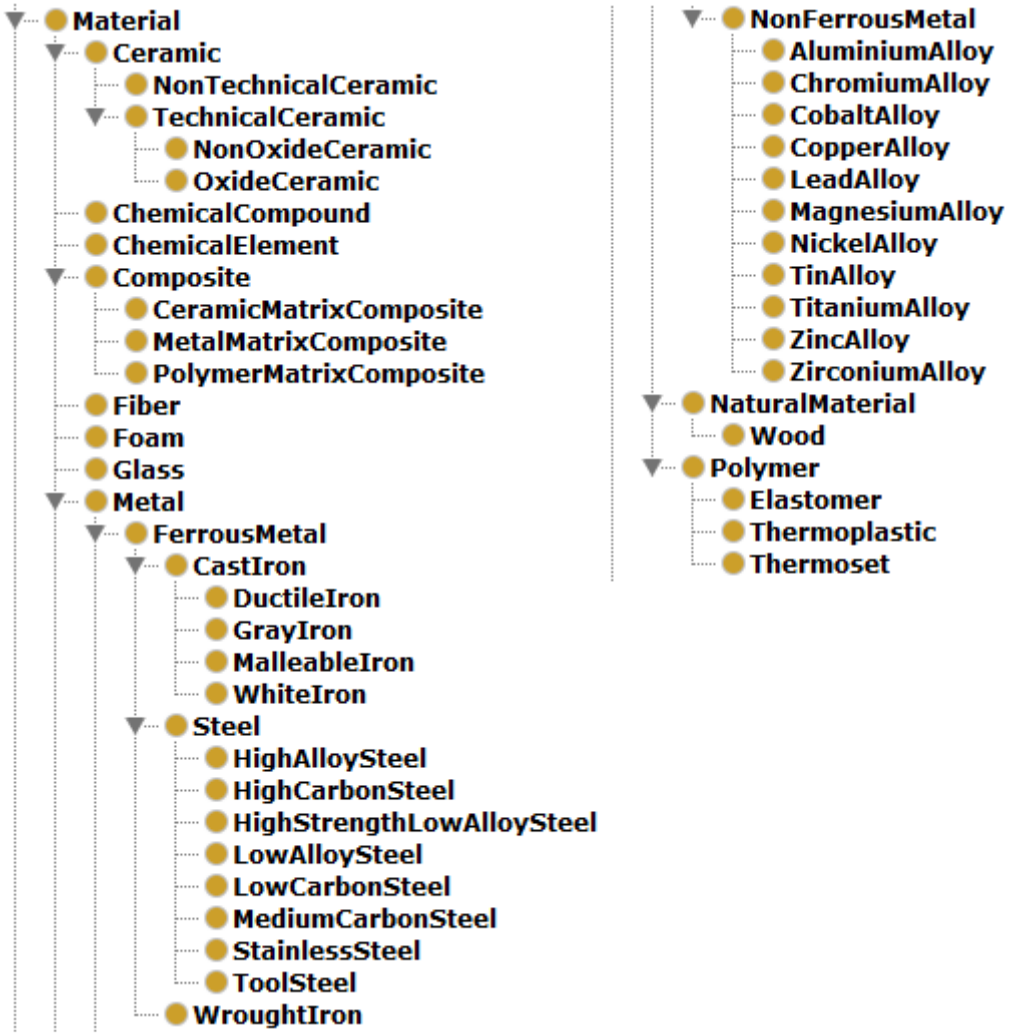
4. Materijali

4.1. Ontologijski model materijala i njihovih svojstava

Ontologijski model materijala i njihovih svojstava je izrađen u programskom paketu Protégé u dva dijela, odnosno dvije *.owl* datoteke zbog lakšeg nadopunjavanja i organizacije. Prvi dio sadrži vrste materijala i svojstva koja će te materijale opisivati. Može se reći da je prvi dio vokabular baze materijala, dok drugi dio sadrži individue. Vrste materijala su kreirane pomoću *owl:Class* resursa, dok su svojstva materijala kreirana pomoću *owl:DatatypeProperty* i *owl:ObjectProperty* relacija.

4.1.1. Ontologija klasa

Slika 72. prikazuje kreiranu strukturu klasa materijala.

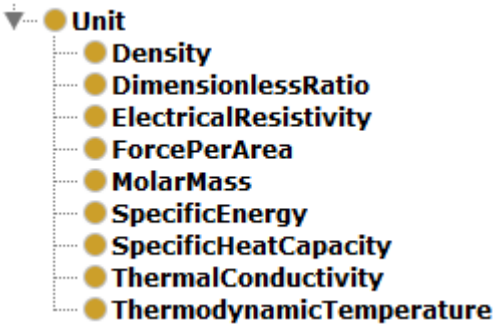


Slika 72. - Struktura klasa (vrste materijala)

Ispod osnovne klase *owl:Thing* OWL modela se nalazi klasa *res:Material* koja sadrži sve ostale podklase materijala, odnosno vrste materijala. Ovdje je napravljena osnovna podjela materijala na keramike, kompozite, vlakna, pjene, stakla, metale, prirodne materijale i polimere, koje su dalje podijeljene na svoje kategorije.

U budućnosti se podjela samih materijala može proširiti, te podijeliti prema nekim drugim kriterijima jer jedan materijal, odnosno resurs u ontologijskom modelu, može biti pripadnik više klasa.

Uz klase vrsta materijala, ovdje postoje i klase mjernih jedinica čija je vrhovna klasa *unit:Unit*. Od njih svaka sadrži varijante određene mjerne jedinice poput različitih standarda, prefiksa i sufiksa, zapisane kao individue. Slika 73. prikazuje neke od tih klasa.



Slika 73. - Struktura klasa (mjerne jedinice)

Kompletan Turtle zapis od klase *res:Material* do klase *res:AluminiumAlloy* je prikazan u sljedećim primjerima.

Turtle zapis osnovne klase *res:Material*:

```
res:Material    rdf:type        owl:Class ;
                rdfs:label      "materijal"@hr .
```

preko čega je definirano da je resurs *res:Material* klasa u ontologijskom modelu, a njegov naziv na hrvatskom jeziku je definiran preko predikata *rdfs:label*.

Na isti način je definirana podklasa *res:Metal*, *res:NonFerrousMetal*, te klasa *res:AluminiumAlloy* uz dodatak *rdfs:subClassOf*.

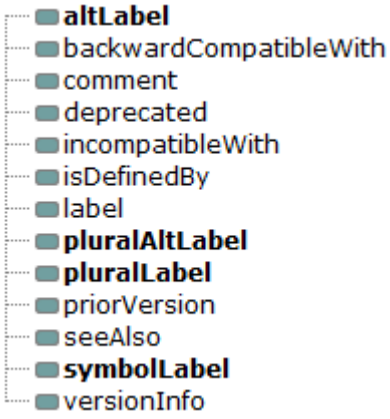
```
res:Metal           rdf:type            owl:Class ;
                    rdfs:label          "metal"@hr ;
                    rdfs:subClassOf     res:Material .

res:NonFerrousMetal rdf:type            owl:Class ;
                    rdfs:label          "obojeni metal"@hr ;
                    rdfs:subClassOf     res:Metal .

res:AluminiumAlloy  rdf:type            owl:Class ;
                    rdfs:label          "aluminijeva legura"@hr ;
                    rdfs:subClassOf     res:NonFerrousMetal .
```

4.1.2. Ontologija opisnih relacija

Kreirane su dodatne opisne relacije, odnosno *owl:AnnotationProperty* kako bi se povećao broj načina za pretraživanje i opisivanje materijala doslovnim vrijednostima. Slika 74. prikazuje unaprijed definirane opisne relacije i nove relacije kreirane za potrebe web aplikacije.



Slika 74. - Opisne relacije

Uz obavezno korištenu *rdfs:label* relaciju koja obično predstavlja pojam u jednini pisan na način razumljiv korisniku, dodane su još relacije *pro:altLabel*, *pro:pluralAltLabel*, *pro:pluralLabel*, *pro:symbolLabel*. Relacija *pro:altLabel* služi za dodatne alternativne nazive za pojedini resurs. Iako je to moguće napraviti sa standardnom *rdfs:label* relacijom, ovdje će ta relacija ostati rezervirana za prikaz, dok će se preko *pro:altLabel* tražiti alternativni nazivi. Relacije *pro:pluralLabel* i *pro:pluralAltLabel* služe za pohranjivanje naziva u množini, dok relacija *pro:symbolLabel* služi za povezivanje simbola koji predstavljaju mjerne jedinice ili simbola kemijskih elemenata.

Sljedeći Turtle zapisi prikazuju način na koji su definirane ove relacije:

```
pro:altLabel        rdf:type    owl:AnnotationProperty .
pro:pluralAltLabel  rdf:type    owl:AnnotationProperty .
pro:pluralLabel     rdf:type    owl:AnnotationProperty .
pro:symbolLabel     rdf:type    owl:AnnotationProperty .
```

Sada je moguće pojedinim vrstama materijala dodijeliti alternativne nazive, kao i nazive u množini na sljedeći način:

```
res:DuctileIron     rdf:type            owl:Class ;
                    rdfs:label          "žilavi lijev"@hr ;
                    pro:altLabel        "nodularni lijev"@hr ;
                    pro:pluralLabel     "žilavi ljevovi"@hr ;
                    pro:pluralAltLabel  "nodularni ljevovi"@hr ;
                    rdfs:subClassOf     res:CastIron .
```

4.1.3. Ontologija relacija

Svojstva pojedinih materijala su povezana relacijama tipa *owl:DatatypeProperty*, te *owl:ObjectProperty*, ovisno o tome da li je vrijednost novi resurs ili podatak. Slika 75. prikazuje relacije koje povezuju materijal sa doslovnim vrijednostima, a to su ujedno i osnovna svojstva materijala. Ovdje je također prikazana podjela svojstava prema vrsti, odnosno podjela na relacije i podrelacije.

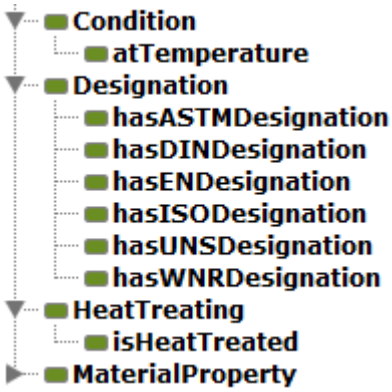
Slika 75. - Struktura relacija (svojstva materijala)

Vrhovna relacija je *pro:MaterialProperty* koja je podijeljena na grupe svojstava *pro:hasAtomicProperty*, *pro:hasChemicalProperty*, *pro:hasEcoProperty*, *pro:hasElectricalProperty*, *pro:hasMechanicalProperty*, *pro:hasOpticalProperty*, *pro:hasPhysicalProperty* i *pro:hasThermalProperty* od kojih svaka sadrži odgovarajuću vrstu svojstva materijala.

Sljedeći primjer prikazuje jedno svojstvo definirano kao *owl:DatatypeProperty*, sa nazivom „toplinska vodljivost“ definiranim preko relacije *rdfs:label* na hrvatskom jeziku, te kratkim opisom svojstva preko relacije *rdfs:comment* i dodijeljenim u grupu toplinskih svojstava, odnosno *pro:hasThermalProperty* preko relacije *rdfs:subPropertyOf*.

pro:hasThermalConductivity	rdf:type	owl:DatatypeProperty ;
	rdfs:label	"toplinska vodljivost"@hr ;
	rdfs:comment	"Određuje intenzivnost kojom se toplina provodi kroz materijal u stacionarnom stanju (temperaturni profil se ne mijenja tijekom vremena)."@hr ;
	rdfs:subPropertyOf	pro:hasThermalProperty .

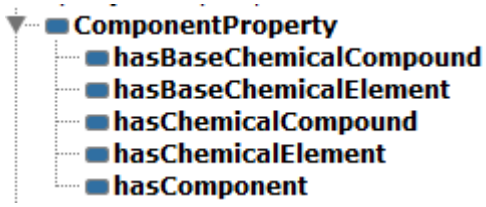
Osim samih svojstava materijala, ovaj tip relacija još sadrži relacije za oznake materijala prema raznim normama kao što su *pro:hasASTMDesignation*, *pro:hasDINDesignation*, *pro:hasENDesignation*, *pro:hasISODesignation*, *pro:hasUNSDesignation* i *pro:hasWNRDesignation*. Također sadrži relaciju za opis toplinske obrade kojim je materijal obrađen, te relacije kojim je moguće opisati pri kojim uvjetima je određeno svojstvo mjereno, poput *pro:atTemperature*. Slika 76. prikazuje te relacije uz osnovnu *pro:MaterialProperty*.



Slika 76. - Struktura relacija (oznake materijala, uvjeti)

4.1.4. Ontologija objektnih relacija

S obzirom da objektne relacije povezuju dvije individue, a u bazi će materijale i kemijske elemente predstavljati individue, potrebno je koristiti objektne relacije. Slika 77. prikazuje kreirane objektne relacije.



Slika 77. - Struktura objektnih relacija (sastav materijala)

Ove relacije su predviđene za povezivanje pojedinih materijala sa njegovim sastavom, odnosno komponentama bile one kemijski elementi ili kemijski spojevi. Relacija *pro:hasBaseChemicalElement* je predviđena za osnovni, najzastupljeniji element koji čini pojedini materijal, a *pro:hasChemicalElement* za ostale koji su zastupljeni u manjoj

mjeri. Isto tako vrijedi i za kemijske spojeve koji se koriste na primjer kod polimera, a mogu biti kreirani kao individue.

Slika 78. prikazuje relacije koje povezuju brojčanu vrijednost, mjernu jedinicu i svojstvo.

Slika 78. - Struktura objektnih relacija (mjerne jedinice)

4.1.5. Ontologija individua

Drugi dio je predviđen za materijale, kemijske spojeve, kemijske elemente i mjerne jedinice, odnosno individue tipa *owl:NamedIndividual*.

Turtle zapis jedne mjerne jedinice prikazuje sljedeći primjer:

```
unit:MegaPascal     rdf:type              unit:ForcePerArea ,
                                          owl:NamedIndividual ;
                    pro:symbolLabel       "MPa" ;
                    unit:hasBaseUnit      unit:Pascal .
```

Resurs *unit:MegaPascal* je svrstan u klasu *unit:ForcePerArea* te je definiran kao *owl:NamedIndividual* što predstavlja individuu. Definirana mu je oznaka *"MPa"*, te kao osnovna jedinica mu je pridružen *unit:Pascal* preko relacije *unit:hasBaseUnit* što sa dodatnim vezama omogućuje pretvorbu mjernih jedinica.

Sljedeće je dan Turtle zapis jednog kemijskog elementa dodijeljen u odgovarajuću grupu, sa opisom i svojstvima.

```
res:Iron            rdf:type                res:ChemicalElement ,
                                            owl:NamedIndividual ;
                    rdfs:label              "željezo"@hr ;
                    pro:symbolLabel         "Fe" ;
                    pro:hasAtomicNumber     [ unit:hasValue "26"^^xsd:double ] ;
                    pro:hasAtomicWeight
[ unit:hasValue "55.845"^^xsd:double ; unit:hasUnit unit:KilogramPerKiloMole ] ;
                    pro:hasDensity
[ unit:hasValue "7870"^^xsd:double ; unit:hasUnit unit:KilogramPerCubicMeter ] ;
                    pro:hasModulusOfElasticity
[ unit:hasValue "211"^^xsd:double ; unit:hasUnit unit:GigaPascal ] ;
                    pro:hasShearModulus
[ unit:hasValue "83.1"^^xsd:double ; unit:hasUnit unit:GigaPascal ] ;
                    pro:hasBulkModulus
[ unit:hasValue "168"^^xsd:double ; unit:hasUnit unit:GigaPascal ] ;
                    pro:hasPoissonsRatio
[ unit:hasValue "0.28"^^xsd:double ] ;
                    pro:hasMeltingPoint
[ unit:hasValue "1810"^^xsd:double ; unit:hasUnit unit:Kelvin ] ;
```

```
                pro:hasBoilingPoint
        [ unit:hasValue "3020"^^xsd:double ; unit:hasUnit unit:Kelvin ] ;
                pro:hasSpecificHeatCapacity
        [ unit:hasValue "440"^^xsd:double ; unit:hasUnit unit:JoulePerKilogramKelvin ] ;
                pro:hasThermalConductivity
        [ unit:hasValue "80.2"^^xsd:double ; unit:hasUnit unit:WattPerMeterKelvin ] ;
                pro:hasElectricalResistivity
        [ unit:hasValue "9.71"^^xsd:double ; unit:hasUnit unit:MicroOhmCentimeter ] .
```

Iz prikazanog je lako uočiti da je kemijski element *res:Iron* dodijeljen u klasu *res:ChemicalElement*, te da je on individua što je prikazano preko *owl:NamedIndividual*. Nadalje, njegov naziv na hrvatskom je preko relacije *rdfs:label* postavljen kao željezo, a njegov simbol *"Fe"* je dodijeljen preko relacije *pro:symbolLabel*.

Nadalje slijede svojstva željeza. Uglate zagrade u Turtle sintaksi označavaju prazne čvorove preko kojih je grupa koju čine vrijednost i mjerna jedinica povezana sa svojstvom materijala. Prazni čvorovi su korišteni kako bi kasnije svako svojstvo pojedinog materijala moglo sadržavati više vrijednosti bez međusobnog ispreplitanja. Više vrijednosti se koristi kako bi se prikazao interval vrijednosti pojedinog svojstva, odnosno uspoređivale minimalne i maksimalne vrijednosti sa traženim vrijednostima.

Tako na primjer relacija *pro:hasMeltingPoint* predstavlja talište željeza, te je preko praznog čvora i relacije *unit:hasValue* dodana brojčana vrijednost *1810*, a preko relacije *unit:hasUnit* mjerna jedinica *unit:Kelvin*.

Za kraj je prikazan kompletan Turtle zapis materijala AW-1050A, kojemu je u ovom slučaju dan URI *res:m31247134*:

```
res:m31247134       rdf:type        res:AluminiumAlloy ,
                                    owl:NamedIndividual ;
                    rdfs:label      "AW-1050A" ;
                    pro:hasUNSDesignation       "A91050" ;
                    pro:hasENDesignation        "AW-1050A" ;
                    pro:hasDINDesignation       "AL99.5" ;
                    pro:hasWNRDesignation       3.0255 ;
                    pro:hasDensity
[ unit:hasValue "2680"^^xsd:double ; unit:hasUnit unit:KilogramPerCubicMeter ] ,
[ unit:hasValue "2740"^^xsd:double ; unit:hasUnit unit:KilogramPerCubicMeter ] ;
                pro:hasBaseChemicalElement
[ pro:hasComponent res:Aluminium ; unit:hasValue "99.5"^^xsd:double ; unit:hasUnit unit:Percent ] ;
                pro:hasChemicalElement
[ pro:hasComponent res:Copper ; unit:hasValue "0"^^xsd:double ; unit:hasUnit unit:Percent ] ,
[ pro:hasComponent res:Copper ; unit:hasValue "0.05"^^xsd:double ; unit:hasUnit unit:Percent ] ;
                pro:hasChemicalElement
[ pro:hasComponent res:Iron ; unit:hasValue "0"^^xsd:double ; unit:hasUnit unit:Percent ] ,
[ pro:hasComponent res:Iron ; unit:hasValue "0.4"^^xsd:double ; unit:hasUnit unit:Percent ] ;
                pro:hasChemicalElement
[ pro:hasComponent res:Manganese ; unit:hasValue "0"^^xsd:double ; unit:hasUnit unit:Percent ] ,
[ pro:hasComponent res:Manganese ; unit:hasValue "0.05"^^xsd:double ; unit:hasUnit unit:Percent ] ;
```

```
                pro:hasChemicalElement
[ pro:hasComponent res:Silicon ; unit:hasValue "0"^^xsd:double ; unit:hasUnit unit:Percent ] ,
[ pro:hasComponent res:Silicon ; unit:hasValue "0.25"^^xsd:double ; unit:hasUnit unit:Percent ] ;
                pro:hasChemicalElement
[ pro:hasComponent res:Titanium ; unit:hasValue "0"^^xsd:double ; unit:hasUnit unit:Percent ] ,
[ pro:hasComponent res:Titanium ; unit:hasValue "0.03"^^xsd:double ; unit:hasUnit unit:Percent ] ;
                pro:hasChemicalElement
[ pro:hasComponent res:Vanadium ; unit:hasValue "0"^^xsd:double ; unit:hasUnit unit:Percent ] ,
[ pro:hasComponent res:Vanadium ; unit:hasValue "0.05"^^xsd:double ; unit:hasUnit unit:Percent ] ,
                pro:hasChemicalElement
[ pro:hasComponent res:Zinc ; unit:hasValue "0"^^xsd:double ; unit:hasUnit unit:Percent ] ,
[ pro:hasComponent res:Zinc ; unit:hasValue "0.05"^^xsd:double ; unit:hasUnit unit:Percent ] ;

                pro:hasModulusOfElasticity
[ unit:hasValue "69"^^xsd:double ; unit:hasUnit unit:GigaPascal ] ,
[ unit:hasValue "72"^^xsd:double ; unit:hasUnit unit:GigaPascal ] ;
                pro:hasFlexuralModulus
[ unit:hasValue "69"^^xsd:double ; unit:hasUnit unit:GigaPascal ] ,
[ unit:hasValue "72"^^xsd:double ; unit:hasUnit unit:GigaPascal ] ;
                pro:hasShearModulus
[ unit:hasValue "25"^^xsd:double ; unit:hasUnit unit:GigaPascal ] ,
[ unit:hasValue "27"^^xsd:double ; unit:hasUnit unit:GigaPascal ] ;
                pro:hasBulkModulus
[ unit:hasValue "64"^^xsd:double ; unit:hasUnit unit:GigaPascal ] ,
[ unit:hasValue "71"^^xsd:double ; unit:hasUnit unit:GigaPascal ] ;
                pro:hasPoissonsRatio
[ unit:hasValue "0.325"^^xsd:double ] ,
[ unit:hasValue "0.335"^^xsd:double ] ;
                pro:hasYieldStrength
[ unit:hasValue "33"^^xsd:double ; unit:hasUnit unit:MegaPascal ] ,
[ unit:hasValue "37"^^xsd:double ; unit:hasUnit unit:MegaPascal ] ;
                pro:hasUltimateTensileStrength
[ unit:hasValue "76"^^xsd:double ; unit:hasUnit unit:MegaPascal ] ,
[ unit:hasValue "84"^^xsd:double ; unit:hasUnit unit:MegaPascal ] ;
                pro:hasCompressionStrength
[ unit:hasValue "33"^^xsd:double ; unit:hasUnit unit:MegaPascal ] ,
[ unit:hasValue "37"^^xsd:double ; unit:hasUnit unit:MegaPascal ] ;
                pro:hasFlexuralStrength
[ unit:hasValue "33"^^xsd:double ; unit:hasUnit unit:MegaPascal ] ,
[ unit:hasValue "37"^^xsd:double ; unit:hasUnit unit:MegaPascal ] ;
                pro:hasElongationAtBreak
[ unit:hasValue "35.2"^^xsd:double ; unit:hasUnit unit:Percent ] ,
[ unit:hasValue "40.9"^^xsd:double ; unit:hasUnit unit:Percent ] ;
                pro:hasMeltingPoint
[ unit:hasValue "918"^^xsd:double ; unit:hasUnit unit:Kelvin ] ,
[ unit:hasValue "931"^^xsd:double ; unit:hasUnit unit:Kelvin ] ;
                pro:hasMaximumServiceTemperature
[ unit:hasValue "403"^^xsd:double ; unit:hasUnit unit:Kelvin ] ,
[ unit:hasValue "473"^^xsd:double ; unit:hasUnit unit:Kelvin ] ;
                pro:hasThermalConductivity
[ unit:hasValue "224"^^xsd:double ; unit:hasUnit unit:WattPerMeterKelvin ] ,
[ unit:hasValue "234"^^xsd:double ; unit:hasUnit unit:WattPerMeterKelvin ] ;
```

```
                pro:hasSpecificHeatCapacity
[ unit:hasValue "893"^^xsd:double ; unit:hasUnit unit:JoulePerKilogramKelvin ] ,
[ unit:hasValue "903"^^xsd:double ; unit:hasUnit unit:JoulePerKilogramKelvin ] ;
                pro:hasElectricalResistivity
[ unit:hasValue "2.8"^^xsd:double ; unit:hasUnit unit:MicroOhmCentimeter ] ,
[ unit:hasValue "3"^^xsd:double ; unit:hasUnit unit:MicroOhmCentimeter ] .
```

Svojstva materijala su u većini slučajeva opisana sa intervalom vrijednosti tako da ovdje svako svojstvo sadrži po dvije vrijednosti, iako je moguće i više.

5. Web aplikacija za pretragu materijala

Sučelje za pretraživanje baze materijala izrađeno je kao web aplikacija na bazi PHP jezika, a kao sama baza podataka sa SPARQL procesorom korišten je Open Link Virtuoso. Poveznica između njih je ODBC konekcija koja dolazi sa Virtuoso mrežnim poslužiteljem. Iako komercijalna verzija Virtuosa ima mogućnost raditi kao PHP server, te predstavljati kompletno rješenje, ovdje je kao besplatna zamjena korišten WAMP Server.

U nastavku će biti objašnjeni samo glavni SPARQL upiti koji se odvijaju u pozadini sučelja pri pretraživanju baze podataka.

5.1. SPARQL upiti za pretraživanje baze podataka

5.1.1. Osnovni upiti

Kao prvo pretraživanje, u sučelju postoji mogućnost pretrage svojstava, grupa i individualnih materijala prema njihovom nazivu ili oznakama. Za svaki od njih postoji opcija za uključivanje i isključivanje iz pretrage, kako bi se što lakše i brže došlo do željenih rezultata.

Kroz sve tri mogućnosti, u SPARQL upitu se pojavljuje PHP varijabla *$searchInput* u kojoj je spremljen tekst prema kojim se pretražuje. Tako na primjer želimo li naći svojstvo *vlačna čvrstoća*, dovoljno je početi upisivati naziv svojstva i sa svakom promjenom teksta se ta vrijednost sprema u PHP varijablu *$searchInput* kako bi se prenijela u SPARQL upit.

SPARQL upit koji pretražuje postojeća svojstva materijala u bazi:

```
SELECT DISTINCT ?u ?l ?al (GROUP_CONCAT(?pl; SEPARATOR = " > ") AS ?b) ?t WHERE {
VALUES ?s { "$searchInput" }
?p rdfs:subPropertyOf* pro:MaterialProperty .
?u rdfs:subPropertyOf{1,} ?p .
?p pro:pluralLabel ?pl .
?u rdfs:label ?l .
OPTIONAL { SELECT ?u (GROUP_CONCAT(?dl; SEPARATOR = ", ") AS ?al)  WHERE {
            ?u pro:altLabel ?dl . } }
FILTER (CONTAINS(lcase(str(?l)), lcase(?s)) || CONTAINS(lcase(str(?al)), lcase(?s)))
BIND("1" AS ?t) .
} GROUP BY ?u ?l ?al ?t
```

Kod ovog upita u prvom redu se vrijednost iz PHP varijable *$searchInput* sprema u SPARQL varijablu *?s*.

Nakon toga se pronalaze sva svojstva materijala u bazi koja su spremljena kao podrelacije glavne relacije *pro:MaterialProperty*. Ona se spremaju u varijablu *?p*. Tada se u varijablu *?u* spremaju sve relacije koje su za jedno ili više koljena udaljene od relacije *?p*. Ovdje se koriste dvije varijable kako bi se omogućilo dohvaćanje putanje svake relacije, odnosno njihove vrhovne relacije, a time odredila vrsta pojedinog svojstva.

U sljedećim koracima se dohvaćaju doslovni nazivi svakog svojstva preko relacija *pro:pluralLabel* i *rdfs:label*. U varijabli *?p* su spremljene vrste svojstava tako da se tu dohvaćaju nazivi u množini poput: mehanička svojstva, optička svojstva, toplinska svojstva... U varijabli *?u* su spremljena pojedina konkretna svojstva poput: gustoća, vlačna čvrstoća, temperatura taljenja...

S obzirom da neka svojstva mogu ali i ne moraju imati alternativne nazive, taj korak je potrebno staviti pod naredbu OPTIONAL kako bi se rezultat uzeo u obzir, iako vrijednost ne postoji. Sva alternativna imena se spremaju u varijablu *?al* odvojena zarezom.

Pomoću naredbe FILTER i naredbe CONTAINS filtriramo sve rezultate koji u sebi ne sadrže vrijednosti koje se traže, odnosno niz znakova za kojima se vrši pretraga. U isto vrijeme se zadan niz znakova i znakovi u bazi pretvaraju u mala slova kako bi se izbjeglo nepodudaranje zbog razlike u velikim i malim slovima.

Kod svojstava materijala, pretražuju se nazivi i alternativni nazivi spremljeni u varijable *?l* i *?al*.

Na kraju upita u varijablu *?t* je spremljen „1“ koji određuje vrstu pronađenog rezultata, a u ovom slučaju predstavlja svojstvo materijala. On kasnije ima ulogu samo u PHP kodu.

Ovaj upit vrača sadržaj varijabli: *?u, ?l, ?al, ?b* i *?t*. One redom sadrže URI, naziv i alternativne nazive svojstva materijala, vrstu svojstva, te vrstu rezultata.

Drugi upit kojim se pretražuju grupe materijala izgleda ovako:

```
SELECT DISTINCT ?u ?l ?al (GROUP_CONCAT(?cl; SEPARATOR = " >") AS ?b) ?t WHERE {
VALUES ?s { "$searchInput" }
?c rdfs:subClassOf* res:Material .
?u rdfs:subClassOf{1,} ?c .
?c pro:pluralLabel ?cl .
?u pro:pluralLabel ?l .
OPTIONAL { SELECT ?u (GROUP_CONCAT(?dl; SEPARATOR = ", ") AS ?al)  WHERE {
            ?u pro:pluralAltlabel|pro:altLabel ?dl . } }
FILTER (CONTAINS(lcase(str(?l)), lcase(?s)) || CONTAINS(lcase(str(?al)), lcase(?s)))
BIND ("3" AS ?t) .
} GROUP BY ?u ?l ?al ?t
```

Ovaj upit je vrlo sličan upitu za pretragu svojstava materijala, a razlika je u tome što umjesto relacije *rdfs:subPropertyOf* i *pro:MaterialProperty,* preko relacije *rdfs:subClassOf* i klase *res:Material* pretražuju se klase, odnosno vrste materijala.

Vrijednosti koje ovaj upit vraća su URI, naziv i alternativni naziv klase materijala (u ovom slučaju u množini), grupu materijala i vrstu rezultata. Ovdje vrijednost „3“ spremljena u varijablu *?t* predstavlja pretragu za grupom materijala i koristi se samo u PHP kodu.

SPARQL upit za pretragu samih materijala izgleda ovako:

```
SELECT DISTINCT ?u ?l ?al (GROUP_CONCAT(?cl; SEPARATOR = " > ") AS ?b) ?t WHERE {
VALUES ?s { "$searchInput" }
?c rdfs:subClassOf* res:Material .
?u rdf:type/rdfs:subClassOf* ?c .
?c pro:pluralLabel ?cl .
?u rdfs:label ?l .
OPTIONAL { SELECT ?u (GROUP_CONCAT(CONCAT( ?ol, " ", ?dl); SEPARATOR = ", ") AS ?al) WHERE
{
?o rdfs:subPropertyOf pro:Designation .
?u ?o ?dl .
?o rdfs:label ?ol } GROUP BY ?u  }
FILTER (CONTAINS(lcase(str(?l)), lcase(?s)) || CONTAINS(lcase(str(?al)), lcase(?s)))
BIND ("5" AS ?t) .
} GROUP BY ?u  ?l ?al ?t
```

Kod ovog upita se prvo pronalaze sve podklase koje bi mogle sadržavati individue materijala, a to su sve podklase glavne klase *res:Material*, te se one spremaju u varijablu *?c*. Nakon toga se u varijablu *?u* spremaju svi materijali koji se nalaze u tim klasama.

U sljedećem koraku se dohvaćaju nazivi grupa materijala preko relacije *pro:pluralLabel*, kao i nazivi samih materijala vezanih relacijom *rdfs:label*. Ti nazivi se spremaju u varijable *?cl* i *?l*.

S obzirom da su alternativna imena materijala ustvari oznake po različitim standardima, ovdje se preko OPTIONAL naredbe pretražuju u prvom redu sve vrste oznaka koje postoje u bazi, te spremaju u varijablu *?o*, a zatim se preko tih istih relacija dohvaćaju doslovne vrijednosti koje se spremaju u varijablu *?dl*, grupirane i odvojene zarezom.

Na kraju se kao i u prethodnim primjerima vrši filtriranje onih materijala koji u svom nazivu ili u ovom slučaju oznakama drugih standarda sadrže traženi niz znakova. Varijabli *?t* se dodaje „5“ kao oznaka da je pronađeni rezultat materijal.

Treba napomenuti da se na ovaj način pretražuju i sami kemijski elementi s obzirom da su oni podklasa klase *res:Material*.

5.1.2. Upiti za dohvaćanje vrijednosti

Nakon pretraživanja, kada su prikazani kompaktni rezultati pretrage, moguće je klikom na pojedini rezultat prikazati detaljne podatke o njemu, ovisno da li se radi o materijalu, grupi materijala ili svojstvu.

U svakoj vrsti tih rezultata prikazuje se dodatni opis vezan relacijom *rdfs:comment*, te njegov URI.

Kod svojstava materijala, cilj detaljnijeg prikaza je grafički prikazati vrijednosti tog svojstva kod pojedinih grupa materijala i usporediti ih sa ukupnim intervalom vrijednosti koji postoji u bazi podataka.

Želi li se tako na primjer pronaći gustoća, tada u detaljnom prikazu gustoće cilj je dohvatiti minimalnu i maksimalnu vrijednost gustoće svih materijala koji postoje u bazi, a zatim i pojedinih grupa materijala kako bi se prikazalo kolike su gustoće pojedinih grupa u odnosu na sve materijale koji postoje u bazi.

Isto tako vrijedi i za pretraživanje grupa materijala, samo u tom slučaju, u detaljnom prikazu se prikazuju sva svojstva te grupe materijala i uspoređuju se sa minimalnim i maksimalnim vrijednostima svih svojstava materijala koje postoje u bazi.

Na sličan način se prikazuju i detaljne informacije kod pretrage pojedinog materijala. U tom slučaju se također prikazuju vrijednosti svojstava tog materijala i uspoređuju sa ostalim materijalima u bazi podataka. Uz to se dohvaća i sastav materijala, odnosno zastupljenost pojedinih kemijskih elemenata i spojeva u njemu, te se prema tim podacima kreira dijagram.

SPARQL upit za prikaz pojedinog svojstva materijala se sastoji od dva dijela. U prvom dijelu se dohvaća minimalna i maksimalna vrijednost tog svojstva koja postoji u bazi i, naziv tog svojstva, te komentar ako postoji. Taj upit izgleda ovako:

```
SELECT DISTINCT ?p ?co ?ul (MIN(?t) AS ?minT) (MAX(?t) AS ?maxT) WHERE {
VALUES ?p { <$uri> } .
_:a ?p [ unit:hasValue ?t ; unit:hasUnit ?u ] .
OPTIONAL { ?u pro:symbolLabel ?ul . }
OPTIONAL { ?p rdfs:comment ?co . }
} GROUP BY ?p ?co ?ul
```

Sada je u PHP varijabli *$uri* spremljen URI svojstva za koje je potrebno pronaći dodatne informacije, te se on dodaje SPARQL varijabli *?p*. nakon toga se traže sve vrijednosti koje su povezane sa tim svojstvom. Unutar naredbi OPTIONAL se pronalaze nazivi mjerne jedinice i komentar, tj. dodatni opis svojstva materijala. Minimalna i maksimalna vrijednost se spremaju u varijable *?minT* i *?maxT*.

U drugom koraku se pretražuju svi materijali uz njihove odgovarajuće grupe, te se za svaku pojedinu grupu pronalaze minimalna i maksimalna vrijednost odabranog svojstva.

```
SELECT ?c ?l ?al (MIN(?v) AS ?minV) (MAX(?v) AS ?maxV) WHERE {
VALUES ?p { <$uri> }
?c rdfs:subClassOf* res:Material .
?m rdf:type ?c .
?c pro:pluralLabel ?l .
OPTIONAL { SELECT ?c (GROUP_CONCAT(?dl; SEPARATOR = ", ") AS ?al)  WHERE {
 ?c pro:pluralAltlabel ?dl . } }
?m ?p [ unit:hasValue ?v ] .
} GROUP BY ?c ?l ?al
ORDER BY ?l
```

Kao i u prvom dijelu, URI pojedinog svojstva se sprema u varijablu *?p*, te se grupe materijala spremaju u varijablu *?c*, a sami materijali u varijablu *?m*. Za razliku od prvog dijela gdje je na mjestu subjekta stajala privremena varijabla *_:a* jer nije bilo bitno koji materijal posjeduje koju vrijednost, već je bilo potrebno samo izvući najmanju i najveću, ovdje je potrebno odrediti koji materijal posjeduje koju vrijednost, kako bi ih se moglo grupirati, tj. odrediti minimalnu i maksimalnu vrijednost pojedine grupe. Te vrijednosti se uz odgovarajuću grupu spremljenu u varijabli *?c*, spremaju u varijable *?minV* i *?maxV*. U varijablama *?l* i *?al* su spremljeni naziv, odnosno alternativni nazivi pojedine grupe.

SPARQL upit koji služi za detaljni prikaz grupe materijala se također sastoji od dva upita u kojem prvi dohvaća samo dodatni opis grupe i vrlo je sličan prethodnome, dok drugi upit izgleda ovako:

```
SELECT ?sp ?sl ?p ?l ?al ?minV ?maxV ?minT ?maxT ?ul WHERE {
{ SELECT ?p (MIN(?v) AS ?minV) (MAX(?v) AS ?maxV) ?u WHERE {
VALUES ?c { <$uri> }
?m rdf:type/rdfs:subClassOf* ?c .
?m ?p [ unit:hasValue ?v ; unit:hasUnit ?u ] .
} GROUP BY ?p ?u }
{ SELECT ?p (MIN(?t) AS ?minT) (MAX(?t) AS ?maxT) WHERE {
_:m ?p [ unit:hasValue ?t ] .
} GROUP BY ?p }
?p rdfs:subPropertyOf ?sp .
?sp rdfs:subPropertyOf pro:MaterialProperty .
OPTIONAL { ?u pro:symbolLabel ?ul . }
OPTIONAL { ?p rdfs:label ?l . }
OPTIONAL { ?sp pro:pluralLabel ?sl . }
OPTIONAL { SELECT ?p (GROUP_CONCAT(?dl; SEPARATOR = ", ") AS ?al) WHERE {
?p pro:altLabel ?dl . } }
} ORDER BY ?sp
```

Vidljivo je da se ovaj upit sastoji od nekoliko dodatnih podupita. U prvom podupitu se varijabli *?c* dodaje URI klase, odnosno grupe materijala za koju je potrebno prikazati detaljne informacije, a on se nalazi u PHP varijabli *$uri*. Nakon toga pronalaze se svi materijali i koji su sadržani u toj klasi ili njezinim podklasama, te njihova svojstva. Tada se pronalaze minimalna i maksimalna vrijednost pojedinog svojstva u pojedinoj grupi materijala, te se spremaju u varijable *?minV* i *maxV*. U drugom podupitu se pronalaze ukupne minimalne i maksimalne vrijednosti pojedinih svojstva koje se nalaze u bazi i

one se spremaju u varijable *?minT* i *?maxT*. Ostale varijable *?sp, ?sl, ?p, ?l, ?al* i *?ul* redom sadrže vrstu svojstva, naziv vrste svojstva, svojstvo materijala, naziv i alternativni naziv svojstva, te mjernu jedinicu.

SPARQL upit za prikaz detaljnih podataka o pojedinom materijalu se sastoji od tri dijela. Prvi dio kao i dosad služi za dohvaćanje dodatnog opisa (ako postoji) pojedinog materijala. Drugi dio je zadužen za dohvaćanje sastava materijala a izgleda ovako:

```
SELECT ?l (MIN(?v) AS ?minV) (MAX(?v) AS ?maxV) ?ul WHERE {
VALUES ?m { <$uri> } .
?p rdfs:subPropertyOf* pro:ComponentProperty .
?m ?p [ pro:hasComponent ?c ; unit:hasValue ?v ; unit:hasUnit ?u ] .
?u pro:symbolLabel ?ul .
?c rdfs:label ?l .
} GROUP BY ?l ?ul ORDER BY DESC (?maxV)
```

U SPARQL varijablu *?m* se sprema vrijednost pohranjena u PHP varijabli *$uri*, odnosno URI samog materijala. Nakon toga se u varijabli *?p* spremaju sve relacije koje su podrelacije *pro:ComponentProperty*. Na taj način su u varijabli *?p* spremljene relacije *pro:hasBaseChemicalElement* i *pro:hasChemicalElement*. U sljedećem koraku se preko praznog čvora za odgovarajući materijal dohvaćaju sve komponente tog materijala preko relacije *pro:hasComponent* koje se spremaju u varijablu *?c*, iznosi koji se spremaju u varijablu *?v*, te mjerna jedinica koja se sprema u varijablu *?u*. Pretražuje se simbol mjerne jedinice koji se sprema u varijablu *?ul*, te naziv same komponente koji se sprema u varijablu *?l*. U ovom slučaju će naziv biti ime kemijskog elementa, a na mjestu mjerne jedinice će stajati *„%“*.

Treći dio je sličan upit kao i upit za detaljni prikaz svojstava materijala pojedine grupe materijala, samo u ovom slučaju se ne uspoređuju svojstva svih materijala u pojedinoj grupi, već samo jednog materijala.

Nakon što se u web aplikaciji dodaju svi filtri za pretraživanje kao što su ograničenja vrijednosti svojstva materijala, vrste materijala koje se žele pretraživati i sami materijali, tada dolazi upit koji objedinjuje sve uvjete i izgleda ovako:

```
SELECT DISTINCT * WHERE {
{ SELECT ?m WHERE {
 { SELECT ?m WHERE {
  VALUES ?class { $class }
  ?m rdf:type/rdfs:subClassOf* ?class .}
} UNION {    SELECT ?m WHERE {
VALUES ?m { $material } } }
} }
VALUES ( ?p ?l ?h ) { ( <$property> $low $high )}
{ SELECT DISTINCT ?m ?ml ?p ?pl (MIN(?v) AS ?minV) (MAX(?v) AS ?maxV) ?ul WHERE {
   ?m ?p [ unit:hasValue ?v ; unit:hasUnit ?u ] .
            OPTIONAL { ?u pro:symbolLabel ?ul . }
            OPTIONAL { ?p rdfs:label ?pl . }
            OPTIONAL { ?m rdfs:label ?ml . }
  } GROUP BY ?m ?ml ?p ?pl ?ul  } }
```

Ovaj upit se sastoji od više podupita. U prvom podupitu traže se svi materijali koji spadaju u željene vrste materijala. U varijablu *?c* su iz PHP varijable *$class* spremljene sve željene vrste materijala. Ako je kao kriterij za pretragu dodana vrsta materijala ili više njih, tada će ovaj podupit pronaći sve materijale koji spadaju u te grupe, odnosno individue koje su članovi te klase i njenih podklasa, te ih spremiti u varijablu *?m*.

Isto tako će se u varijablu ?m spremiti i svi individualno dodani materijali koji se žele usporediti sa ostalima, a spremljeni su u PHP varijabli *?material*.

Nakon toga u varijablu *?p* se spremaju sva svojstva materijala zadana kao filtar u web aplikaciji, te zadane minimalne i maksimalne tražene vrijednosti, koje će se kasnije u PHP kodu uspoređivati sa dobivenim vrijednostima iz baze.

Drugi podupit je zadužen za pronalazak kombinacije materijala i svojstava materijala koja su zadana u filtrima, te njihove minimalne i maksimalne vrijednosti, nazive i mjerne jedinice.

Za pronalaženje materijala čije vrijednosti odgovaraju traženim vrijednostima zadužen je PHP kod. On uzima dvije zadane (tražene) vrijednosti pojedinog svojstva, minimalnu i maksimalnu, te uspoređuje sa minimalnom i maksimalnom vrijednosti pojedinog materijala. Ako je minimalna tražena vrijednost manja od maksimalne vrijednosti materijala i ako je maksimalna tražena vrijednost veća od minimalne vrijednosti materijala, tada se zadani (traženi) interval vrijednosti i interval vrijednosti određenog svojstva pojedinog materijala preklapaju. U tom slučaju određeno svojstvo materijala zadovoljava uvjete. Ako sva svojstva na ovaj način zadovoljavaju uvjete, tada se materijal uzima kao zadovoljavajući rezultat pretrage.

5.2. Prikaz i korištenje web aplikacije za pretragu materijala

Prije svega treba napomenuti da trenutno u bazi postoji samo 5 materijala, od toga su 3 aluminijeve legure, 2 titanove legure i nehrđajući čelik. Na prethodno prikazan način moguće je dodati materijale, svojstva materijala, grupe materijala bez ikakvih ograničenja.

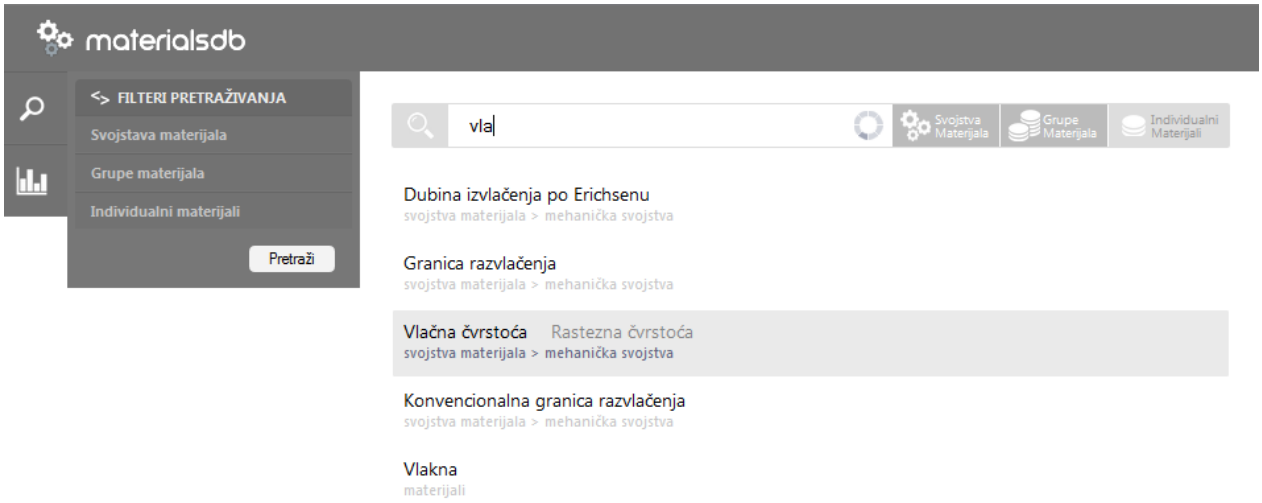
5.2.1. Pretraga

Slika 79. prikazuje početnu stranicu, ujedno i stranicu za osnovnu pretragu materijala, vrsta materijala i njihovih svojstava.

Slika 79. - Početna stranica web aplikacije

Forma za pretraživanje sadrži tri gumba, *Svojstva Materijala*, *Grupe Materijala* i *Individualni Materijali*. Pomoću njih je moguće odrediti što se pretražuje kako bi se kod velikog broja rezultata neki filtrirali i na taj način smanjio njihov broj, te postigla veća preglednost i brže pronalaženje. Ovdje su *Svojstva Materijala* i *Grupe Materijala* uključene, dok je pretraga za *Individualnim Materijalima* isključena.

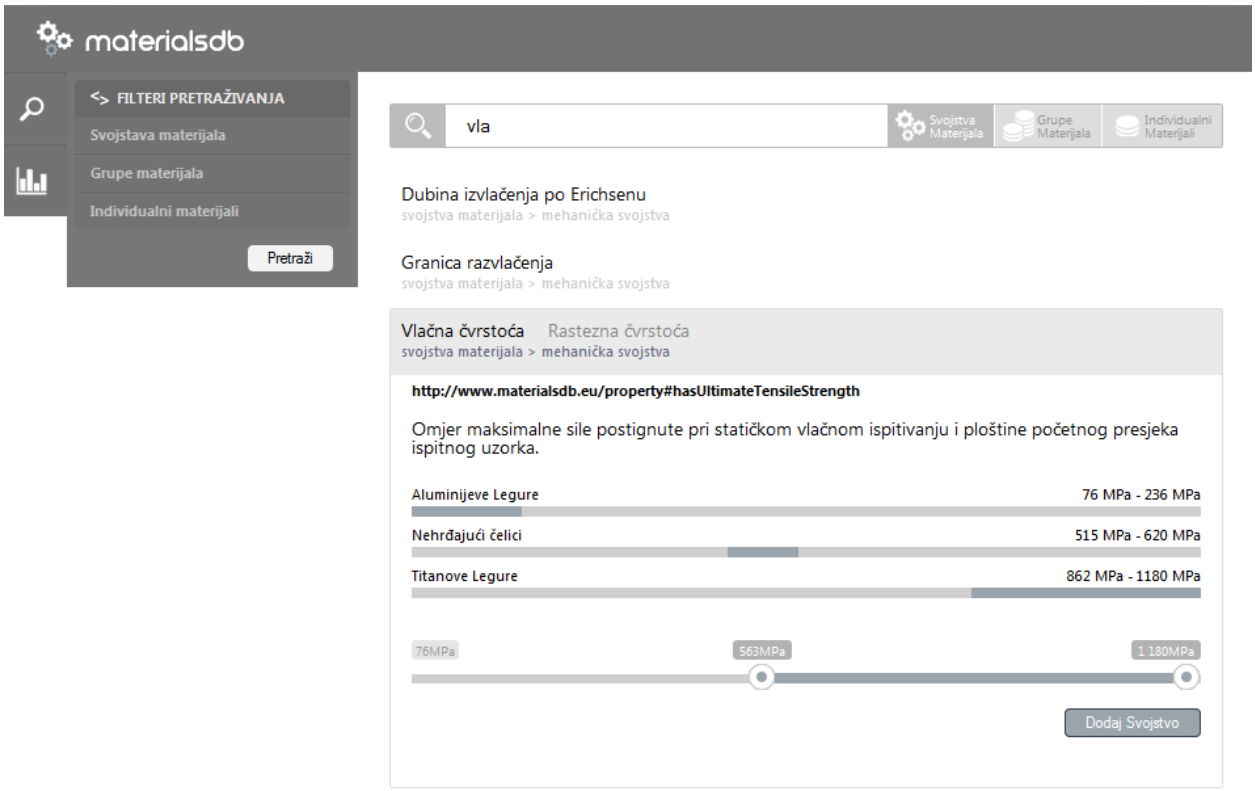
Kod upisa se automatski nude rezultati koji u sebi sadrže zadane znakove, tako na primjer ako je potrebno pronaći svojstvo materijala *vlačna čvrstoća*, kod upisa znakova *vla* će se već ponuditi rezultati koji u sebi sadrže taj niz znakova, a u ovom slučaju rezultati koji su ujedno svojstva materijala ili grupe materijala. Slika 80. prikazuje osnovno pretraživanje.



Slika 80. - Osnovno pretraživanje

Vidljivo je da su u rezultatima prikazana 4 svojstva materijala i jedna grupa materijala, njihova imena, alternativna imena i njihove vrste i podvrste. Tražena *vlačna čvrstoća* spada u svojstva materijala i to u mehanička svojstva, kao i ostala pronađena svojstva. *Vlakna* spadaju u grupu *materijali*.

Klikom na pojedini rezultat otvaraju se njegove detaljne informacije. Slika prikazuje prozor sa dodatnim informacijama.



Slika 81. - Prikaz dodatnih informacija o svojstvu materijala

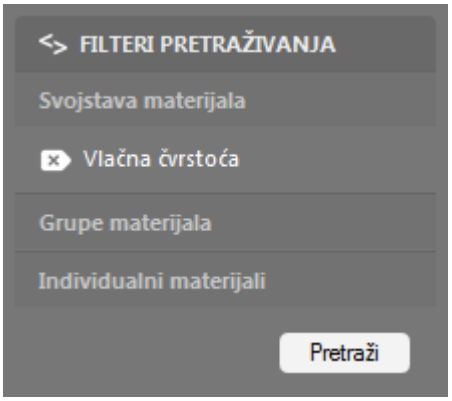
Prije svega treba napomenuti da u ovim rezultatima, minimalna i maksimalna vrijednost pojedinog svojstva se ispisuje samo za postojeće materijale u bazi, a ne kao općeniti podatak.

Odmah ispod naslova prikazuje se URI vlačne čvrstoće, a nakon njega se prikazuje dodatni opis.

Nadalje se automatski generira grafički prikaz odnosa vlačne čvrstoće pojedine grupe materijala naspram ukupnog iznosa vlačne čvrstoće, a isto tako i odnose vlačne čvrstoće između samih grupa materijala. Tako na primjer aluminijeve legure, koje postoje u bazi, imaju najniže vrijednosti koje se kreću između 76 MPa i 236 MPa, dok se vrijednosti titanovih legura kreću od 862 MPa pa do maksimalnih 1180 MPa.

Na dnu se nalazi pomična traka pomoću koje je moguće odrediti traženi interval vrijednosti vlačne čvrstoće, odnosno minimalnu i maksimalnu vrijednost koja će se pretraživati u bazi materijala. Odabranu vrijednost je moguće dodati u filtre pretraživanja klikom na gumb *Dodaj Svojstvo*. Taj se kriterij pretraživanja, odnosno svojstvo

materijala ograničeno vrijednostima, prikazuje na popisu filtera pretraživanja. Slika 82. prikazuje prozor sa filtrima pretraživanja.

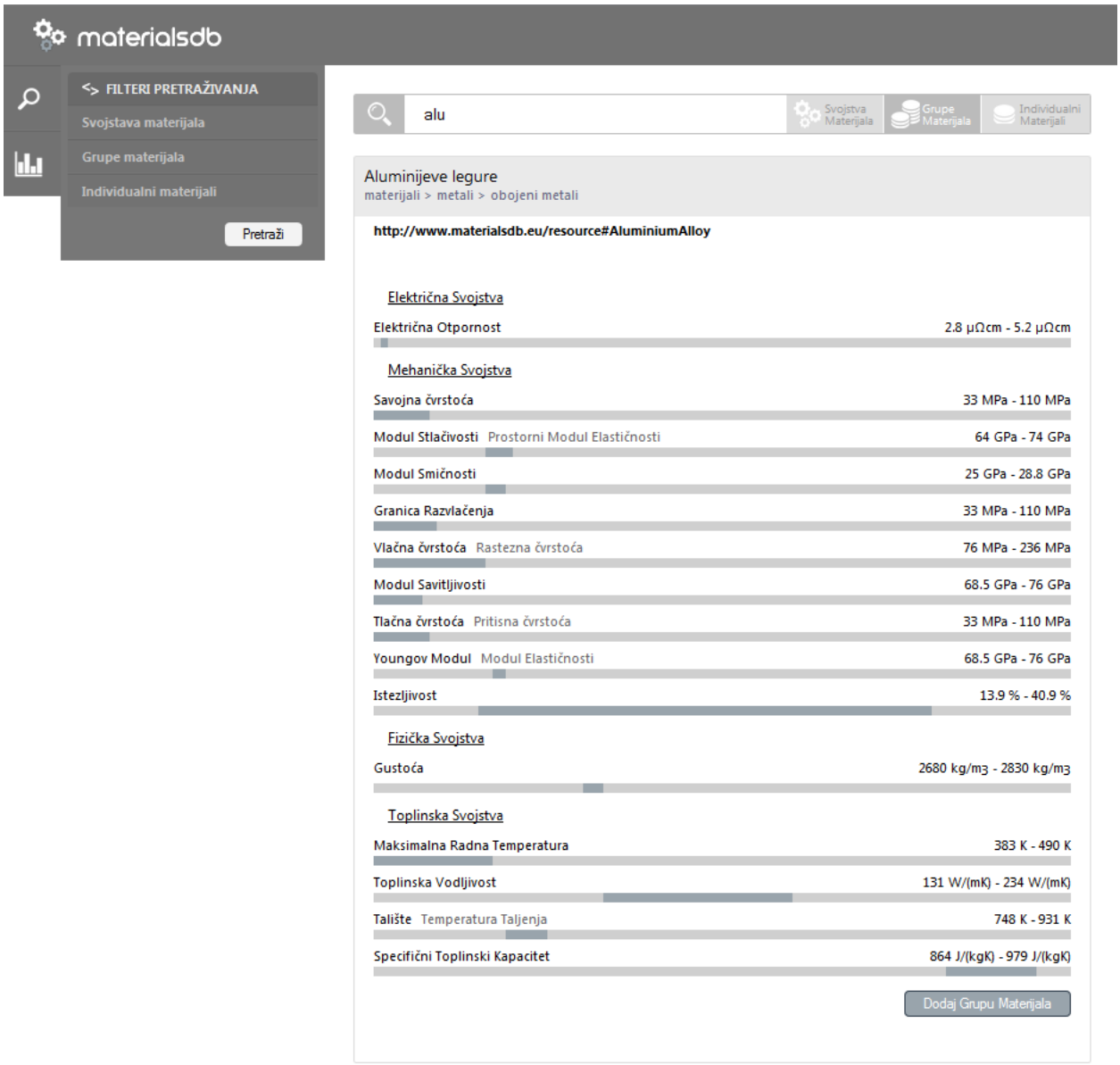


Slika 82. - Filteri pretraživanja

Dodane filtere je moguće ukloniti klikom na gumb sa oznakom x pored imena.

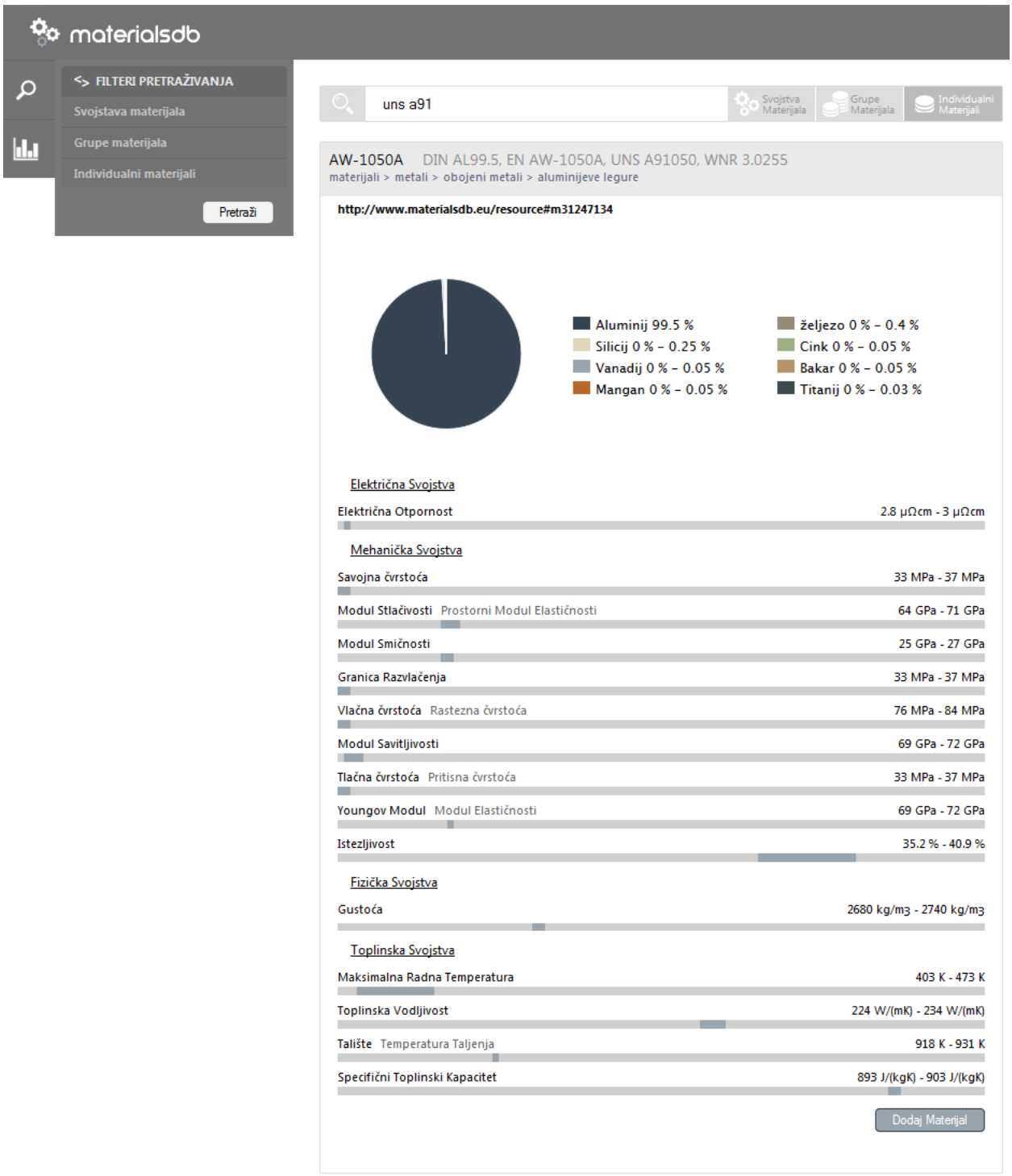
Pretraga grupa materijala se odvija na isti način, ali u ovom slučaju će pri prikazu dodatnih podataka biti prikazana sva svojstva koja odabrana grupa materijala posjeduje, njihove odnose te odnos pojedinog svojstva grupe naspram svih ostalih materijala u bazi. Kod dodavanja grupa materijala u filtre za pretraživanje, pretraga se ograničava samo na te grupe materijala.

Slika 83. prikazuje podatke o svim aluminijevim legurama upisanim u bazu.



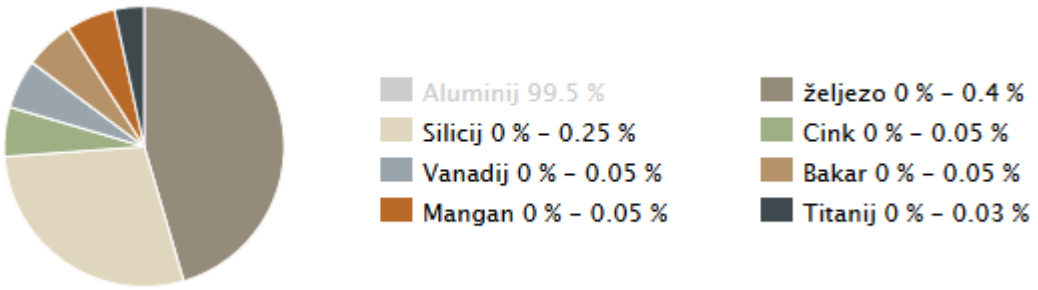
Slika 83. - Prikaz dodatnih informacija o klasi materijala

Slika 84. prikazuje pretraživanje individualnih materijala uključujući dodatne informacije o materijalu koji je u ovom slučaju aluminijeva legura AW-1050A. Prikaz i usporedba su jednaki kao i za grupe materijala, ali ovdje je dodan dijagram koji prikazuje komponente materijala i u kojoj su mjeri zastupljeni.



Slika 84. - Prikaz dodatnih informacija o materijalu

U ovome dijagramu, zastupljenost aluminija je prevelika da bi se ostali elementi mogli prikazati, ali je moguće kliknuti na željeni element te sakriti njegov udio kako bi ostale komponente došle do izražaja. Na taj način je moguće prikazati odnose komponenti sa vrlo malim udjelom. Slika 85. pokazuje dijagram sa skrivenim osnovnim elementom, odnosno aluminijem.

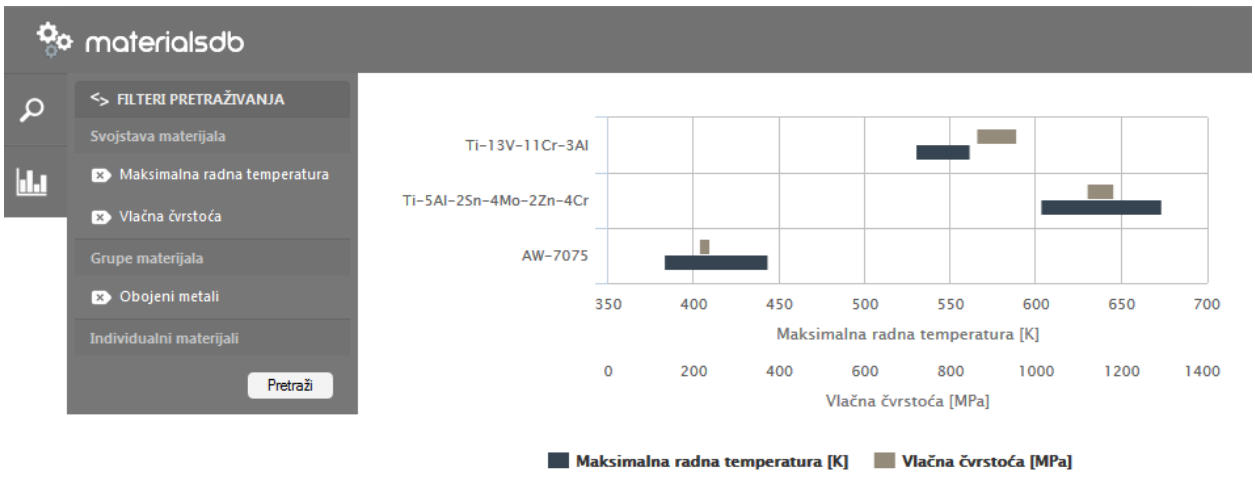


Slika 85. - Dijagram sa skrivenim osnovnim kemijskim elementom

5.2.2. Prikaz rezultata i vizualizacija

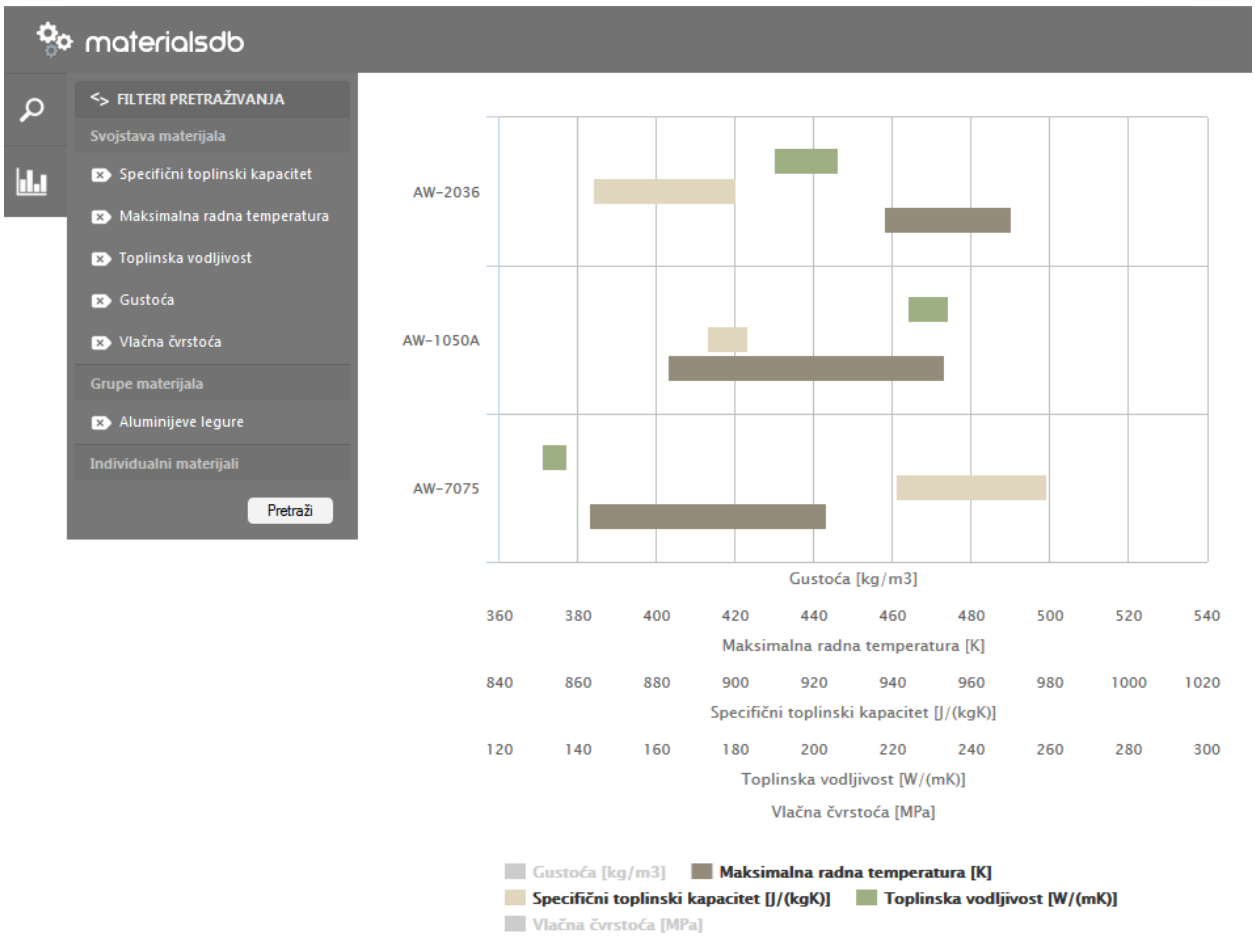
Kombinirajući ove postupke, moguće je dodati razna svojstva materijala, pretraživati grupe materijala (ili sve materijale ako se grupa ne specificira), te individualno dodavati materijale za koje se žele prikazati vrijednosti.

Neka je za primjer potrebno pronaći legure koje mogu izdržati radnu temperaturu najmanje 400K, te imaju vlačnu čvrstoću veću od 130 MPa. Slika 86. prikazuje ta svojstva dodana na listu filtera, uz grupu koja ograničava pretragu na obojene i lake metale, odnosno legure, te dijagram sa rezultatima.



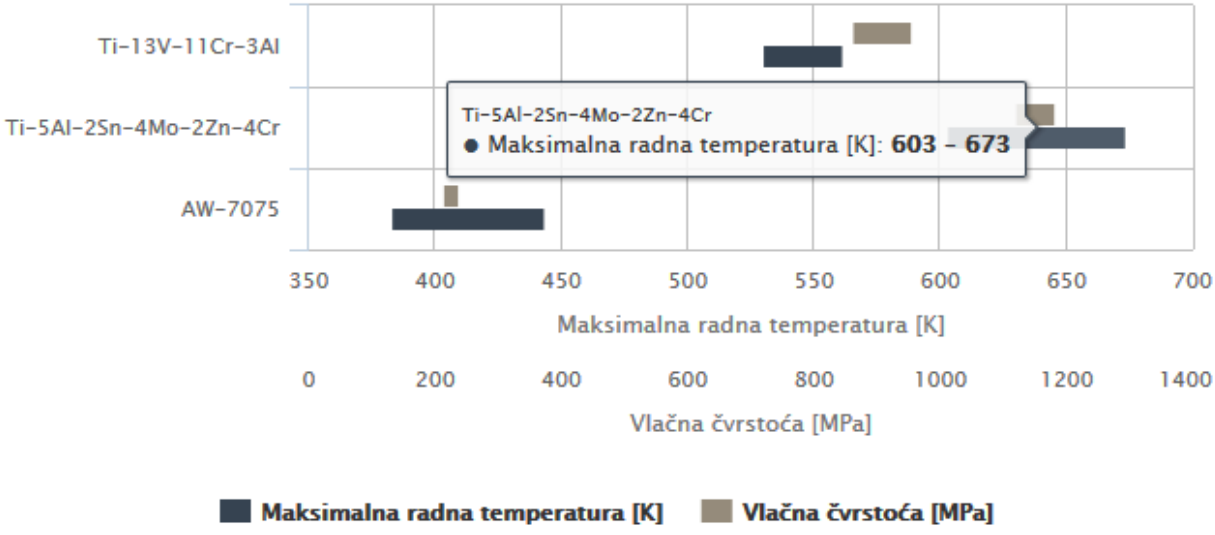
Slika 86. - Vizualizacija rezultata pretrage

Na ovome dijagramu, na y-osi se nalazi popis materijala koji zadovoljavaju zadane kriterije. Što se tiče x-osi, njih postoji onoliko koliko je zadano svojstava materijala sa odgovarajućim intervalom. Svako svojstvo se može sakriti u prikazu klikom na njegovo ime kako bi se dobila bolja preglednost. Tako je moguće filtrirati materijale po više svojstava, a samo odabrane prikazati u dijagramu. Slika 87. prikazuje rezultate sa skrivenim svojstvima.



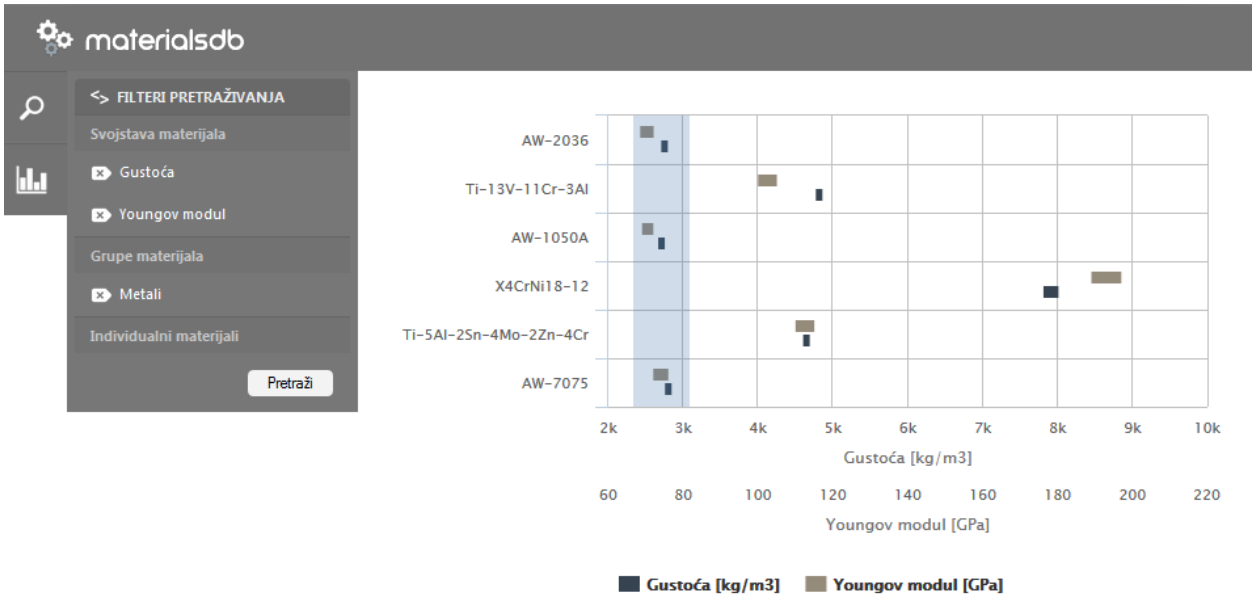
Slika 87. - Sakrivena svojstva materijala u vizualizaciji rezultata

Prelazeći mišem preko pojedinog intervala, prikazuju se točne vrijednosti pojedinog svojstva za određeni materijal. Slika 88. prikazuje prozor sa točnim vrijednostima.



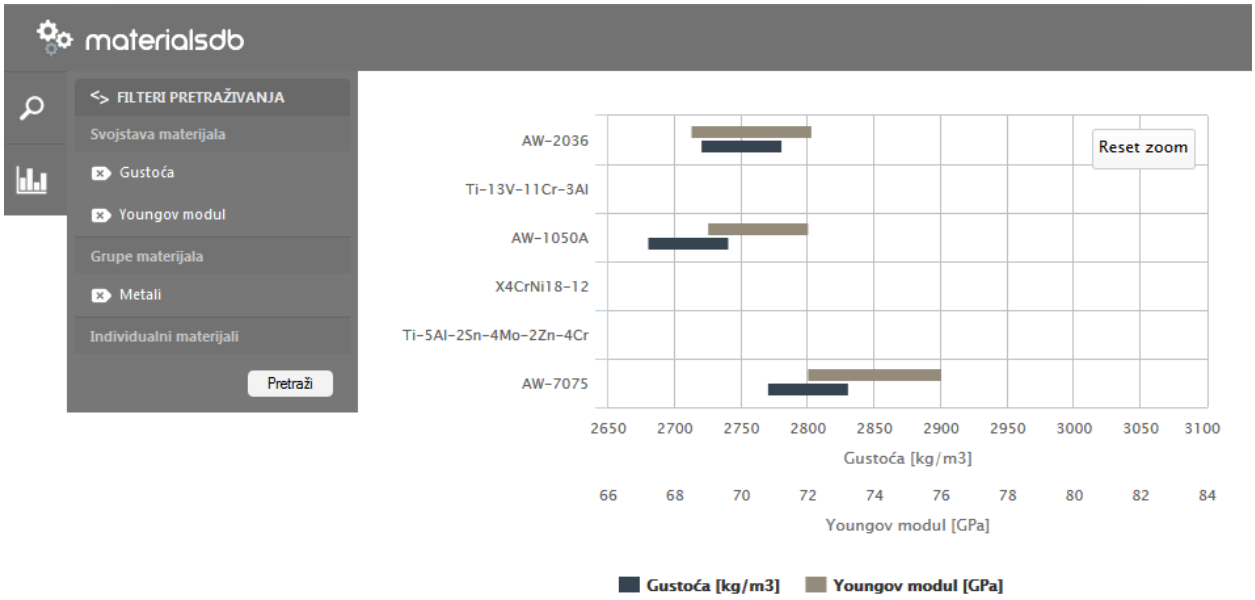
Slika 88. - Točne vrijednosti pojedinog svojstva u vizualizaciji rezultata

Kod vrijednosti koje su premalene u odnosu na ukupni interval nekog svojstva, poput električne otpornosti kod vodiča i izolatora, postoji mogućnost povećanja dijagrama na način da se mišem povuče preko dijela dijagrama koji se želi povećati. Slika 89. prikazuje označen dio za uvećanje.



Slika 89. - Označavanje dijela za uvećanje

Slika 90. prikazuje uvećani prikaz odabranog dijela. Valja napomenuti da se tada i intervali vrijednosti na x-osi prilagođavaju pa je moguće da odnos između dva svojstva nije grafički jednak na oba prikaza.



Slika 90. - Uvećani prikaz rezultata

Pretraživanje materijala se odvija na način da materijal mora zadovoljavati sva svojstva postavljena kao kriteriji u filtrima pretraživanja. U slučaju da materijal nema upisano određeno svojstvo u bazi, on će se u tom slučaju i dalje smatrati kao rezultat. Iz rezultata će se izuzeti jedino ako posjeduje svojstvo, ali se njene vrijednosti ne poklapaju sa traženima. Na taj način se dohvaćaju materijali koji nisu potpuno opisani u bazi, što ne znači da ne zadovoljavaju tražene vrijednosti. U tom slučaju je potrebno provjeriti svojstva pojedinog materijala iz drugih izvora i po mogućnosti nadopuniti bazu.

6. Zaključak

U ovome radu prikazan je temeljni model semantičkog weba, te njegova implementacija pri izradi web aplikacije za izbor inženjerskih materijala. Upravo semantička struktura baze materijala i njihovih svojstava omogućuje daljnje proširenje i povezivanje podataka na globalnoj razini. Uvođenjem semantičkih veza između podataka, moguće je ujedno razviti i bazu znanja bez dodatnih algoritama, koristeći samo procesorsko odlučivanje.

Iako se u ovome radu materijali povezuju samo sa svojim svojstvima, predviđena je mogućnost proširenja baze sa raznim informacijama dobivenim sa različitih izvora. Tako je na primjer moguće da proizvođači materijala povežu materijale iz svoje ponude, njihove odgovarajuće karakteristike i cijenu sa svojstvima kreiranim u ovome radu, te na taj način bude generiran globalni katalog materijala. Isto tako je moguće da proizvođači opišu svoje gotove proizvode i poluproizvode sa materijalima u bazi, tj. od čega su izrađeni. Na taj način ova baza i aplikacija, uz određenu nadogradnju, mogu postati aplikacija za odabir proizvoda i poluproizvoda prema kriterijima koje materijali od kojih su izrađeni ispunjavaju.

Uvođenjem semantičkih veza između podataka, računalni sustavi postaju sustavi koji sadrže bazu znanja uz same podatke. U tom slučaju je još uvijek potreban čovjek za konačni odabir materijala između ponuđenih rješenja. Sljedeći korak u razvoju ove tehnologije bi bio kombinacija umjetne inteligencije sa semantičkim bazama podataka, što bi na kraju rezultiralo sa potpunom automatizacijom odabira inženjerskih materijala.

7. Literatura

[1] Grigoris Antoniou, Frank van Harmelen: A Semantic Web Primer, 2nd edition, 2008.

[2] Dean Allemang, James Hendler: Semantic Web for the Working Ontologist, 2nd edition, 2011.

[3] Tomislav Filetin, Franjo Kovačiček, Janez Indof: Svojstva i primjena materijala, Zagreb, 2006.

[4] J.R. Dixon, C. Poli: Engineering Design and Design for Manufacturing, Field Stone Publishers, 1995.

[5] George E. Dieter: ASM Handbook, Materials Selection and Design Volume 20, 1997.

[6] Bob DuCharme: Learning SPARQL, 2nd edition, 2013.

[7] http://www.w3.org/TR/sparql11-property-paths/#simple_paths, 2014.

[8] http://www.w3.org/TR/sparql11-query/#expressions, 2014.

[9] Tomislav Filetin, Mladen Franz, Đurđica Španiček, Vinko Ivušić: Svojstva i karakteristike materijala, Zagreb, 2012.

[10] Toby Segaran, Colin Evans, Jamie Taylor: Programming the Semantic Web, 2009.

8. Prilozi

I. CD-R sa web aplikacijom i ontologijom materijala