Modeliranje podataka u sustavu za upravljanje bazama podataka AllegroGraph

Košćak, Luka

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:289788

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Luka Košćak

MODELIRANJE PODATAKA U SUSTAVU ZA UPRAVLJANJE BAZAMA PODATAKA ALLEGROGRAPH

DIPLOMSKI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Luka Košćak

Matični broj: 43971/15-R

Studij: Baze podataka i baze znanja

MODELIRANJE PODATAKA U SUSTAVU ZA UPRAVLJANJE BAZAMA PODATAKA ALLEGROGRAPH

DIPLOMSKI RAD

Mentorica:

Prof. dr. sc. Lovrenčić Sandra

Varaždin, rujan 2020.

Luka Košćak

Izjava o izvornosti

Izjavljujem da je moj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema rada koji slijedi prikaz je posebitosti pohrane i rada s podacima u AllegroGraph sustavu za upravljanje bazama podataka koji objedinjuje čak tri različita modela podataka. Fokus rada bit će na analizi arhitekture sustava kao takvog, grafovskog, dokument i RDF modela podataka te korisnosti i praktičnoj primjenjivosti tehnologija semantičkog weba podržanih od strane sustava. Teorija koja će biti dana kroz rad u mnogim slučajevima biti će dodatno objašnjena i prikazana kroz primjere dok će sveukupne mogućnosti AllegroGraph sustava biti praktično primijenjene razvojem RDF baze podataka i manipulacijom podacima u istoj.

Ključne riječi: AllegroGraph, RDF, graf, sustav za upravljanje bazama podataka, zaključivanje, ontologija

Sadržaj

1. Uvod

Vratimo se dvadesetak godina u prošlost i stavimo u položaj osobe koja za potrebe pohrane i upravljanja podacima, u sklopu određenog projekta ili aplikacije, mora odabrati tehnologiju koja će joj navedeno omogućiti, odnosno, mora odabrati bazu podataka. Vrlo je vjerojatno, ako ne i neminovno, da će se odluka bazirati na željenim performansama, skalabilnosti i sigurnosti baze dok će se sama priroda podataka na sve načine pokušati prilagoditi, na tržištu dominantnom, relacijskom modelu podataka.

Zasićenost tržišta baza podataka te sama heterogenost načina na koje se manipulira podacima danas, u isto vrijeme pred nas stavlja nove mogućnosti ali i probleme odabira ispravnog pristupa rukovanja podacima. Naime, ne samo da se baze podataka na trenutnome tržištu razlikuju u modelima podataka oko kojih se grade i koji su prilagođeni korištenju u određenim domenama, već se mnoge od njih karakteriziraju kao takozvani multi-model sustavi što korisnicima uglavnom otežava odabir, jer čak i tehnologije s kojima su relativno dobro upoznati uključivanjem sve većeg broja funkcionalnosti i različitih pristupa postaju kompliciranе i nejasne. Razlog sve češćeg objedinjavanja različitih modela podataka unutar jednog sustava vrlo je očit i proizlazi iz pojave novih domena te želje za ostvarenjem čim veće tržišne prevlasti.

Sustav čije će se mogućnosti rukovanja podacima opisati u ovome radu upravo je jedna takva tehnologija, prvotno proizašla iz područja semantičkog weba i bazičnog RDF (engl. *Resource Description Framework*) modela podataka, da bi danas naglasak stavljala na mogućnostima pohrane podataka u obliku grafa te dokumenata. Pregledom i analizom čimbenika koji su uvjetovali razvoj AllegroGraph sustava u ovome smjeru probat će se ustanoviti da li i u kojoj mjeri multi-model sustavi donose raznolikost u pristupu pohrani podataka ili se tu radi samo o nadovezivanju modela uslijed njihove sličnosti zbog prije navedenog razloga. Kako je AllegroGraph ujedno sustav za upravljanje bazama podataka koji je potekao, a i trenutno se bazira na mnogim standardiziranim tehnologijama danas još, u temeljnoj ideji potpuno nezaživljenog semantičkog weba, u radu će se također moći procijeniti trenutna korisnost te steći praktičan uvid u korištenje istih.

No, prije samog fokusa na AllegroGraph sustav te praktičnog dijela rada u kojem će se izraditi baza podataka u istome, potrebno je teoretski analizirati i kroz razne primjere prikazati karakteristike sustava kao redom grafovske, dokument te naposljetku RDF baze podataka.

2. Grafovske baze podataka

Grafovske baze podataka pristup su pohrani podataka koje karakterizira nestrukturiranost te visoki stupanj povezanosti koji se nastoji efikasnije iskoristiti izostavljanjem skupih spajanja podataka korištenjem indeksa. Iako su godinama bile područje istraživanja te tehnologija koja se nametala kao logičan način rješavanja problema rukovanja podacima u mnogim domenama, tek su nedavno zaživjele u sklopu takozvanog NoSQL (SQL - engl. *Structured Query Language*) pokreta.

2.1. NoSQL pokret

Dugi niz godina, od prve objave rada *A relational model of data for large shared data banks* britanskog znanstvenika i začetnika teorije relacijskih baza podataka Edgara F. Codda 1970. godine [1], sustavi za upravljanje relacijskim bazama podataka, nadalje SURBP, bili su dominantna tehnologija za pohranu strukturiranih podataka ponajprije u poslovnim, a kroz vrijeme i web aplikacijama. Temeljene na relacijskoj algebri i modelu podataka visoke razine apstrakcije u odnosu na prijašnje modele podataka hijerarhijskih i mrežnih baza podataka, relacijske baze podataka praktično nisu imale alternative, a svaki specifičan pokušaj drugačijeg pristupa pohrani podataka u ranim godinama weba, početak 90-tih godina prošlog stoljeća, poput XML (engl. *EXtensible Markup Language*) i objektnih baza podataka ili se nije uspio proširiti tržištem ili je u najčešćem slučaju bio upijen, to jest ukomponiran u SURBP kao jedna od njegovih funkcionalnosti.

Kako to najčešće biva, potrebe današnjice nisu nužno potrebe sutrašnjice. Težnja da SURBP postanu "jedna veličina odgovara svima" (engl. *one size fits all*) rješenja dovela je zapravo do toga da se isti ne ističu ni u čemu [2, str. 8] u kontekstu činjenice da se ne mogu natjecati sa specijaliziranim sustavima za pohranu podataka, dizajniranim za rad u specifičnoj domeni. Ti specijalizirani sustavi za pohranu podataka počeli su se isticati krajem 20. stoljeća kada je i po prvi puta korišten naziv koji je, iako nesretno definiran tako da više govori o tome što ti sustavi odnosno baze podataka nisu nego što jesu [3], obuhvatio i dao naziv pokretu razvoja baza podataka koje se odmiču od tradicionalne upotrebe relacijskog modela podataka, NoSQL.

"Ne SQL-u" (engl. *No to SQL*) bilo je prvo općeprihvaćeno tumačenje naziva pokreta kao puta prema potpunoj zamjeni relacijskih baza novim rješenjima i nemogućnosti kohibitacije s njima u budućnosti. No, uskoro se prvotno oduševljenje smirilo, došlo je do obrata u definiciji

pokreta i realizacije da relacijske baze podataka ne idu nikuda, ali u isto vrijeme nisu u stanju samostalno riješiti sve probleme svijeta [2, str. 7]. Novonastali tok misli dao je NoSQL pokretu novo značenje, “Ne samo SQL“ (engl. *Not only SQL*).

Takva definicija zagovara udaljavanje od i neprihvaćanje pretpostavke da se podaci moraju pohraniti u neki oblik relacijske baze podataka. Kraj pretpostavkama da su SQL i ACID (engl. *atomicity, consistency, isolation, durability*) svojstva transakcija jedini i nezamjenjivi alati za rješavanje problema u domenama za koje su prvotno namijenjeni kao i kraj miješanja relacijskog modela i aplikacijskog koda, tipično kroz razne ORM (engl. *Object-relational mapping*) alate [2, str. 17]. Navedeno je vrlo dobro sročeno u članku programera Davida Keysa koji NoSQL pokret opisuje kao post-relacijski i u kojem govori kako „baš kao što je bit post-modernizma u ponovnom razmatranju prošlosti u umjetnosti i arhitekturi, post-relacijski pokret šansa za programere da preispitaju vlastite pristupe. Kao što post-modernizam nije poništio cjelokupnu povijest umjetnosti, NoSQL neće i ne mora poništiti korisnost relacijskog pristupa pohrani podataka“ [4].

2.1.1. Vrste NoSQL baza podataka

Postoje mnogi razlozi koji su doveli do eksperimentiranja s podacima i traženja novih rješenja za njihovu pohranu, što je pak rezultiralo rješenjima objedinjenima pod nazivom NoSQL. Većinom ti razlozi proizlaze, kao što je već napomenuto, iz pokušaja prilagodbe SURBP novim zahtjevima tržišta, što je u mnogim slučajevima moguće, ali isto tako vrlo nelogično. Relacijske baze podataka dobre su za pohranu relacijskih podataka, ali zašto bi se koristile za pohranu ne-relacijskih podataka te za potrebe za koje prvobitno nisu osmišljene, ukoliko postoje bolja rješenja [2, str. 21]. Razlozi su svakojaki i ne postoji jasno definiran problem kojeg se NoSQL pokretom želi riješiti. Neki autori u fokus razvoja novih rješenja u pohrani podataka stavljaju probleme skalabilnosti i performansi, drugi pak naglašavaju količinu podataka dok treći kao najveću prepreku u efikasnom rukovanju podacima vide njihovu nestrukturiranost i sklonost čestim promjenama. Upravo zbog te raznolikosti u pogledima na najbitnije karakteristike efikasnog i učinkovitog sustava za pohranu podataka te specifičnosti domena u kojima se rukuje podacima, iz NoSQL pokreta potekle su, a potječu i danas, mnoge tehnologije koje dijele velik broj ideja, ali se isto tako međusobno razlikuju kao što se bilo koja od danih tehnologija razlikuje od relacijskih baza podataka. Upravo je to i sama bit NoSQL pokreta, specifičan pristup ispunjenju potreba korisnika.

U Tablici 1 dana je jedna od mnogih klasifikacija dominantnih i prihvaćenih NoSQL tehnologija odnosno baza podataka uz dodanu usporedbu s relacijskim bazama podataka u osnovnim karakteristikama traženim kod današnjih tehnologija pohrane podataka.

Tablica 1: Klasifikacija i usporedba NoSQL baza podataka s relacijskim bazama podataka

	Performanse	Skalabilnost	Fleksibilnost	Kompleksnost	Funkcionalnost
Ključ-vrijednost baze podataka (engl. *Key-Value stores*)	visoke	visoka	visoka	vrlo niska	-
Stupčane baze podataka (engl. Column stores)	visoke	visoka	umjerena	niska	-
Dokument baze podataka (engl. Document stores)	visoke	varijabilna	visoka	niska	-
Grafovske baze podataka (engl. Graph databases)	varijabilne	varijabilna	visoka	visoka	teorija grafova
Relacijske baze podataka	varijabilne	varijabilne	niska	umjerena	relacijska algebra

(Prema: [2, str. 26])

Iz dane klasifikacije i usporedbe može se primijetiti kako se NoSQL tehnologije pohrane podataka razlikuju već u osnovnim karakteristikama što upućuje na njihovu specijaliziranost za određene domene okarakterizirane različitim potrebama. Izdvajanje karakteristika koje su zajedničke svim NoSQL tehnologijama zbog toga je vrlo teško, no sve dane karakteristike proizlaze iz dizajna NoSQL tehnologija pohrane podataka koji je pod utjecajem određenih, NoSQL tehnologijama zajedničkih, koncepata poput CAP (engl. *Consistency, Availability and Partition Tolerance*) teorema i BASE (engl. *Basically available, Soft state, Eventual consistency*) svojstava transakcija.

2.1.2. NoSQL koncepti dizajna

CAP teorem akronim je za tri osnovne karakteristike distribuiranih baza podataka kojima se teži NoSQL pokretom: konzistentnost, dostupnost i toleriranje particioniranja koje je nemoguće postići istovremeno [2, str. 30].

- Konzistentnost je karakteristika koja definira je li i kako baza podataka ostaje u konzistentnom stanju nakon završetka određene operacije nad njenim podacima [2, str. 30]. Za distribuiranu bazu podataka tipično kažemo da je u konzistentnom stanju ako nakon operacije ažuriranja njenih podataka svi korisnici nadalje čitaju takve ažurirane podatke.
- Dostupnost se odnosi na dizajn i implementaciju baze podataka koja je u mogućnosti nastaviti s radom, dozvoljava čitanje iz i pisanje u bazu podataka, u trenutcima u kojima su neki njeni dijelovi u fazi nadogradnje ili nedostupni uslijed nepredvidljivih grešaka.
- Toleriranje particioniranja definira se kao sposobnost baze podataka da nastavlja s radom kad se u njen sustav dodaju novi čvorovi [2, str. 30]. Čvorovi u distribuiranoj bazi podataka podrazumijevaju jedinice koji sadrže particiju ili repliku baze podataka.

Prema ideji teorema, sustav odnosno baza podataka, može istovremeno postići samo dvije od navedenih karakteristika. Izbor ovisi o specifičnostima domene u kojoj se baza podataka koristi, a utječe na odabir ACID ili BASE modela transakcija.

Ukoliko baza podataka mora biti konstantno u konzistentnom stanju te mora dozvoljavati dinamičko dodavanje novih čvorova, najčešće kao odgovor na potrebe skaliranja, ACID svojstva transakcija su neophodna, a ako pak baza podataka ispred konzistentnosti stavlja dostupnost, tada se BASE svojstva transakcija nameću kao bolje rješenje [2, str. 30].

BASE svojstva transakcija ključ su NoSQL rješenja koja u korist skaliranja odbacuju strogu konzistentnost podataka te u fokus stavljaju stalnu dostupnost i dinamička horizontalna proširenja sustava. Akronim definira dizajn sustava, baze podataka, koji radi u osnovi cijelo vrijeme, ne treba biti u konzistentnome stanju cijelo vrijeme ali će to stanje postići eventualno [2, str. 31]. Činjenica da eventualna konzistentnost podrazumijeva stanja baze podataka u kojima svi korisnici ne vide iste podatke na prvi pogled se ne čini zadovoljavajuća, no ta karakteristika omogućuje aplikacijama poput društvenih mreža vrlo lako i efikasno skaliranje. Promjena korisničkih podataka na profilu Facebooka ne treba biti propagirana svim korisnicima Facebooka instantno već eventualno. Imajući na umu da u kontekstu baza podataka

eventualno označava vremenske intervale kraće od sekunde, ovakav pristup dizajnu baze podataka je i više nego prihvaćen na tržištu.

2.2. Osnovno o grafovskim bazama podataka

Iako su grafovske baze podataka kao efikasan način pohrane i manipulacije podacima zaživjele tek nedavno, sama ideja rada s podacima povezanim u strukturu grafa nije nova. Kroz godine koje su uslijedile nakon razvoja prvih baza podataka, koje su implementirale hijerarhijske i mrežne modele podataka, ideja korištenja grane matematike pod nazivom teorija grafova, kao temelja razvoja grafovskih baza podataka, neprestano je rezultirala određenim pomacima, no nikada upotrebljivim rješenjima. Razlog tome može se pronaći u razvoju tehnologija za pohranu podataka poput XML i prostornih baza podataka koje su na vrlo jednostavan način rješavale probleme u domenama prvotno namijenjenim grafovskim bazama podataka [5, str. 1].

Pojavom weba, koji povlači velike količine nestrukturiranih podataka, te posebice neprestanim povećanjem kompleksnosti i dinamičnosti veza između podataka, pojavile su se nove domene i zahtjevi za manipulacijom podataka. Obrada podataka pohranjenih u strukturu grafa postala je važan dio velikog broja područja računalnih znanosti poput strojnog učenja, analize društvenih mreža i sličnog [6, str. 1].

2.2.1. Model grafa

Kao što je već napomenuto, temelj grafovskih baza podataka predstavlja grana matematike utemeljena 1736. godine od strane švicarskog matematičara Leonharda Eulera, koji je rješavanjem problema *Sedam mostova Konigsberga* došao do definicije grafa kao kolekcije čvorova i veza između njih.

Graf je formalno, par (V, E) gdje V predstavlja skup čvorova (engl. *vertices*), a $E \subseteq V \, x \, V$ skup rubova (engl. *edges*) odnosno veza [6, str. 4].

Već ovako jednostavno definiran model grafa, iako rijetko kada korišten kao model podataka implementiran od strane grafovskih baza podataka, ima određene prednosti u pohrani i rukovanju podacima. Kao prvo, omogućuje prirodnije modeliranje podataka na visokoj razini apstrakcije. Podatke o entitetu iz domene drži u pojedinačnome čvoru, a veze između entiteta reprezentiraju veze između čvorova. Objekti grafa (čvorovi, veze ali čak i

putanje te susjedstva) mogu se enkapsulirati i direktno referencirati u upitima [5, str. 5]. Naposljetku, tu je teorija grafova čiji se koncepti i algoritmi prirodno primjenjuju na model grafa.

Cilj svakog modela na kojem se temelji baza podataka, bio on relacijski, objektno-orijentiran ili graf, pružiti je apstrakciju odnosno simplifikaciju domene i podataka koji se u njoj pojavljuju [7, str. 25]. Ono što izdvaja graf kao model podataka zasigurno je izjednačavanje značenja podataka i veza između njih ali i mala razlika između logičkog i fizičkog modela, što omogućuje da na model grafa i upite koje postavljamo nad istim gledamo kao na dvije strane istog novčića .

Ne postoji standardni model grafa kojeg grafovske baze podataka implementiraju i nad kojim se razvijaju algoritmi za provedbu tipičnih operacija poput prolaska grafom, traženja uzoraka, pod-grafova i analize susjedstva. Svaki model grafa, pa tako i svaka grafovska baza podataka, optimizirana je za izvršavanje specifičnih zadataka i upita. Usprkos tome, svi modeli grafa dijele strukturalnu, operativnu i integritetnu komponentu. Navedene komponente opisuju svaki model podataka [8].

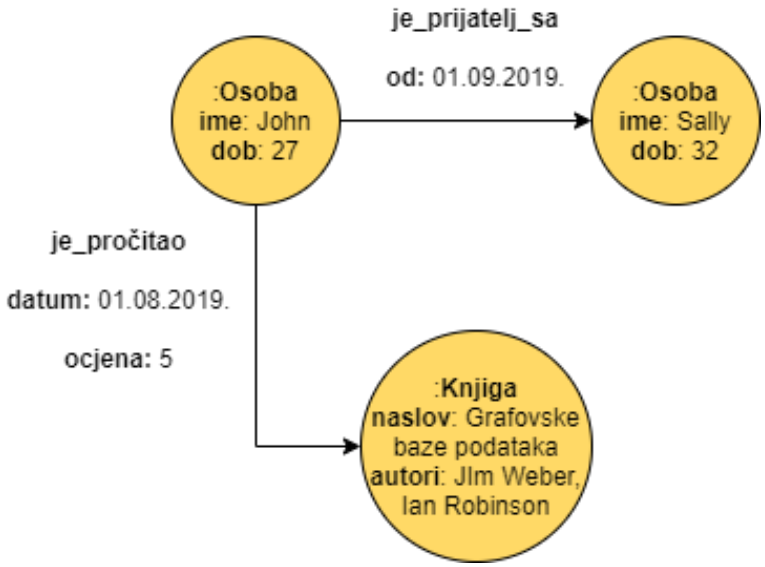
- Strukturalna komponenta opisuje način na koji su reprezentirani podaci. U modelu grafa podaci su predočeni kao čvorovi i veze između njih.
- Operativnu komponentu čine operatori koji potječu iz teorije grafova, a omogućuju transformacije grafa i provođenje operacija nad strukturom grafa.
- Integritetna komponenta odnosi se na ograničenja nad shemom podataka ili samim podacima. Uključuje koncepte poput uvođenja oznaka za čvorove i veze s jedinstvenim nazivima, određivanje domena te raspona vrijednosti podataka i slično.

Jednostavni model grafa nije dovoljno bogat, nema dovoljno koncepata, za adekvatno modeliranje današnjih domena. Dodavanjem koncepta usmjerenosti grafa kojim se ograničava kretanje među čvorovima, oznaka čvorova i veza kojima se diferenciraju njihovi tipovi, atributa kojima se definiraju dodatni podaci o grafu, dolazi se do jednog od dva temeljna modela grafa kojeg implementira većina grafovskih baza podataka:

- označeni graf sa svojstvima (engl. *labeled property graph*), dok je drugi dominantan model grafa
- RDF (engl. *Resource Description Framework*) model grafa uglavnom implementiran od strane RDF baza podataka

2.2.1.1. Označeni graf sa svojstvima

Označeni graf sa svojstvima, kraće samo graf sa svojstvima, formalno se definira kao uređena devetorka $(V, E, L, l_V, l_E, K, W, p_V, p_E)$ u kojoj V predstavlja čvorove, a E veze između njih. L je set oznaka. Funkcije $l_V : V \to P(L)$ i $l_E : E \to P(L)$ pridružuju oznake čvorovima i vezama. Svakom čvoru kao i vezi mogu biti pridružena svojstva i to najčešće kao parovi ključ-vrijednost $p = (ključ, vrijednost)$, gdje $ključ \in K$ i $vrijednost \in W$. K i W setovi su svih mogućih ključeva i vrijednosti. Naposljetku, $p_V(u)$ označuje set svojstava određenog čvora u, a $p_E(e)$ set svojstava veze e [6, str. 5].



Slika 1: Primjer označenog grafa sa svojstvima (Prema: https://s3.amazonaws.com/dev.assets.neo4j.com/wp-content/uploads/modeling_johnsally_properties.jpg)

Slika 1 predstavlja jednostavan primjer grafa sa svojstvima. Model se sastoji od tri čvora koji nose oznake *:Osoba* i *:Knjiga* te dvije veze između čvorova, sa oznakama *je_prijatelj_sa* i *je_pročitao.* Dane oznake definiraju ulogu čvora te značenje veze u domeni koju modeliramo. Čvorovima i vezama pridružen je određen broj svojstava koji ih pobliže opisuju, tako primjerice čvor s oznakom *:Knjiga* sadrži svojstvo *naslov* sa vrijednošću *Grafovske baze podataka.* Mogućnost dodavanja svojstava vezama posebice je korisno kao izvor meta-podataka za algoritme koji ih koriste.

Čak i ovako formalno definiran model grafa često je implementiran s određenim varijacijama. Primjerice, Neo4j SUGBP (Sustav za upravljanje grafovskim bazama podataka) podržava pridruživanje većeg broja oznaka čvorovima, dok ArangoDB SUGBP podržava

maksimalno jednu oznaku po čvoru [6, str. 5]. Neki sustavi također dozvoljavaju proizvoljan broj veza između dva čvora.

2.2.2. Definicija grafovske baze podataka

Sustav za upravljanje grafovskim bazama podataka (SUGBP), sinonimno grafovska baza podataka, SUBP je sa CRUD (engl. *Create, Read, Update, Delete*) metodama implementiranim za rad s modelom grafa [7, str. 5].

Ovako formulirana definicija određuje vrlo malo ograničenja u kategoriziranju sustava kao grafovske baze podataka. Naime, svaki sustav koji korisnicima omogućuje provedbu operacija nad modelom grafa može se smatrati grafovskom bazom podataka neovisno o specifičnim implementacijskim odlukama kao što je pristup pohrani podataka te korištenju određenog graf modela podataka. Samim time grafovske baze podataka mogu se podijeliti u dvije distinktne skupine, nativne i nenativne.

2.2.2.1. Nativne grafovske baze podataka

Nativne grafovske baze podataka u svim svojim dijelovima razvijene su u svrhu efikasnog rada s modelom grafa, a temeljne karakteristike dane su svojevrsnim konsenzusom o pristupu pohrani grafa te implementaciji procesora upita (engl. *query engine*).

Pohrana grafa, odnosno podataka povezanih u strukturu grafa, odnosi se na perzistentnu pohranu istog na disku te njegovu logičku reprezentaciju kod dohvaćanja. Pohrana grafa kod nativnih grafovskih baza podataka podrazumijeva implementacije određenih struktura za reprezentaciju grafa na način koji je optimiziran za provođenje operacija nad istim [7, str. 5]. Strukture podataka, koje su istodobno koriste za reprezentaciju grafa, a koje s određenim ustupcima nativne grafovske baze podataka koriste za pohranu grafa su:

- *Matrica susjedstva* (engl. *Adjacency matrix*) reprezentira graf kao dvodimenzionalno polje bool (engl. *boolean*) vrijednosti koje poprimaju vrijednosti (0 i 1) u ovisnosti o tome jesu li dva čvora međusobno povezana. Vrlo je efikasna struktura podataka za provođenje upita o povezanosti dvaju čvorova te dodavanje novih veza u graf, no mana se nazire u pretraživanju susjedstva određenog čvora te dodavanju novih čvorova [9, str. 9].
- *Matrica incidencije* (engl. *Incidence Matrix*) također je dana u obliku dvodimenzionalnog polja. Razlika u odnosu na matricu susjedstva je u tome što

stupci matrice incidencije predstavljaju veza između čvorova, a ne same čvorove. Povezanost čvorova na taj način može se iščitati iz stupaca [9, str. 9]. Matrica incidencija nerijetko zahtijeva više memorijskog prostora zbog usmjerenosti grafa na veze.

- *Lista susjedstva* (engl. *Adjacency list*) struktura je podataka sastavljena od kolekcija listia gdje je svaka lista povezana s određenim čvorom i sadrži sve njegove susjedne čvorove. Efikasna je struktura podataka za provedbu tipičnih upita nad strukturom grafa koji uključuju prolazak grafom dok joj performanse opadaju kod provedbe upita u kojima algoritam mora odrediti povezanost čvorova danom vezom [9, str. 9].
- *Lista veza* (engl. *Edge list*) gotovo je jednaka listi susjedstva u generalnom dizajnu. Razlika je u tome što su umjesto čvorova, veze eksplicitno spremljene, a pridruženi su im ishodišni i odredišni čvor [6, str. 7].

Procesor upita jedan je od najvažnijih dijelova svakog SUBP-a i odgovoran je za izvršavanje upita te dohvat ili ažuriranje podataka [10]. U suštini predstavlja dio SUBP-a koji izlaže model grafa kroz CRUD operacije. Kao što je već navedeno, grafovske baze podataka stavljaju naglasak na veze između podataka te se većina upita temelji na prolasku grafom. Nativne grafovske baze podataka u svrhu ubrzavanja takvih upita spremaju veze eksplicitno u čvorovima grafa kao pokazivače na susjedne čvorove [10]. Na taj način baza podataka ne treba izvoditi veze koristeći globalne indekse, što je karakteristika nenativnih grafovskih baza podataka. Navedeni pristup implementaciji procesora upita kod nativnih grafovskih baza podataka naziva se susjedstvo bez indeksiranja (engl. *indeks-free adjacency*) [7, str. 5].

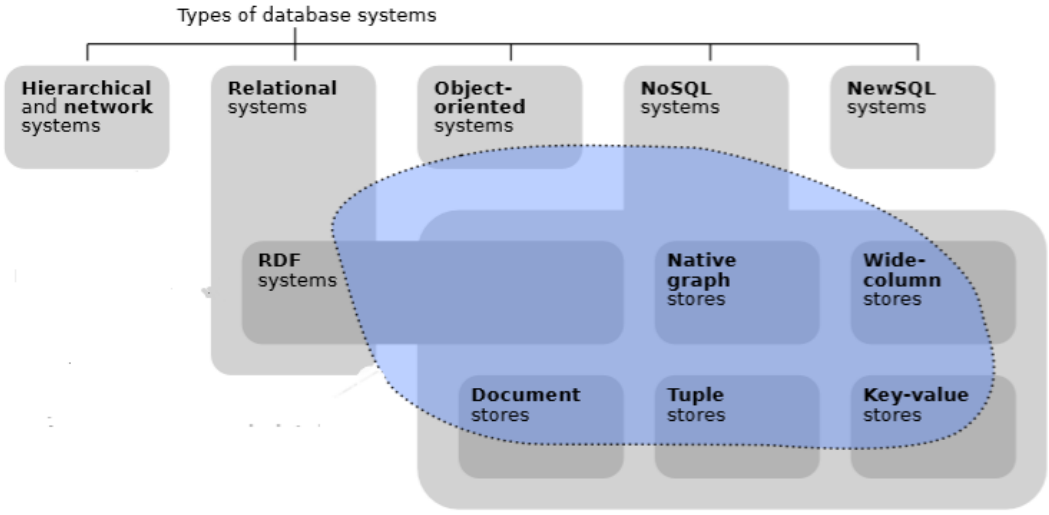
Važno je naglasiti kako nativna pohrana i procesiranje grafa predstavljaju implementacijske karakteristike koje izdvajaju nativne grafovske baze podataka od nenativnih. Dolaze s određenim prednostima poput poboljšanih performansi kod provođenja operacija koje uključuju prolazak grafom, no ne predstavljaju pravilo u razvoju grafovskih baza podataka [7, str. 6].

2.2.2.2. Nenativne grafovske baze podataka

Za razliku od nativnih, nenativne grafovske baze podataka nisu prvotno zamišljene i razvijene za spremanju grafova i manipulaciju istima. Prilagodbom pristupa pohrani podataka, svojevrsnom serijalizacijom podataka, te dodavanjem modula koji omogućuju provedbu operacija nad strukturom grafa primjenom koncepata teorije grafova mogu se okarakterizirati kao grafovske baze podataka [6, str. 9]. U ovu skupinu ubrajamo RDF baze podataka, o kojim će biti riječi u nastavku rada, relacijske baze podataka, objektne te različite vrste NoSQL baza podataka poput dokument i ključ-vrijednost baza podatka.

Zajednička karakteristika im je to što koriste model podataka koji nije zamišljen ali je prilagođen za pohranu grafova. Tako primjerice relacijske baze podataka modeliraju graf kao kolekciju tablica odnosno relacija na način da implementiraju čvorove i veze između njih kao redove u dvije distinktne relacije. Svaki čvor identificiran je jedinstvenim primarnim ključem, dok je veza između dva čvora postignuta referenciranjem primarnih ključeva ishodišnog i odredišnog čvora kao vanjskih ključeva u relaciji veza. Dodatna svojstva modeliraju se uvođenjem stupaca u relacije [6, str. 9].

Na Slici 2 prikazana je klasifikacija specifičnih tehnologija pohrane podataka, kao i njihova upotreba kao grafovskih baza podataka. Vidljivo je kako se nativne grafovske baze podataka smatraju tehnologijom proizašlom iz NoSQL pokreta, dok se nenativne grafovske baze podataka prilagođavaju modelu grafa i nisu prvotno zamišljene za njegovu pohranu. AllegroGraph je jedan od primjera RDF nenativne grafovske baze podataka.



Slika 2: Ilustracija tipova baza podataka uzetih u obzir (Izvor: [6, str. 3])

2.3. Prednosti grafovskih baza podataka

Pretpostavka da se gotovo sve može prikazati pomoću grafova i da grafovske baze podataka nude djelotvoran način modeliranja podataka nije ni približno suficijalno opravdanje zamjene dobro proučenih i usvojenih tehnika rukovanja podacima kao primjerice relacijskih baza podataka. Potrebna je značajna praktična korist. Kod grafovskih baza podataka, bilo nativnih ili nenativnih, sljedeće karakteristike prema [7] nadjačavaju druge tehnologije baza podataka s obzirom na današnje poslovne zahtjeve okarakterizirane stalnim promjenama te uskim poslovnim rokovima:.

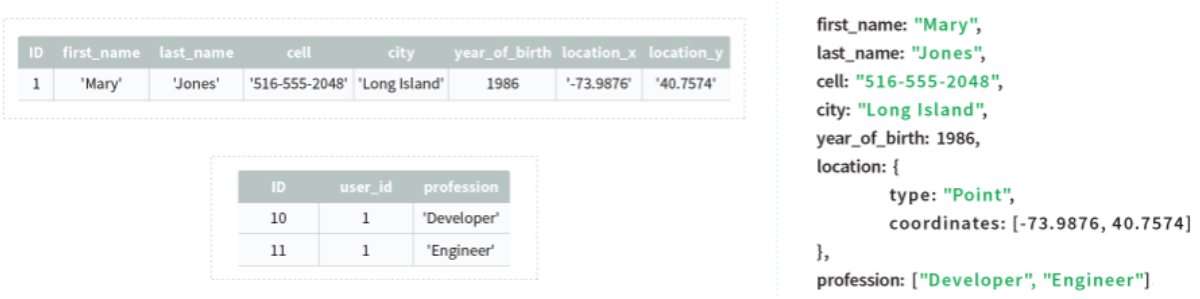
- *Performanse* koje uslijed povećanja količine podataka uglavnom ostaju konstantne zbog mogućnosti lokalizacije upita na dio grafa, što pak rezultira vremenom izvršavanja upita koje ne ovisi o veličini cijelog grafa [7, str. 8];
- *Fleksibilnost* koja proizlazi iz grafa kao modela podataka i omogućuje mijenjanje sheme grafovske baze podataka u tandemu s razumijevanjem domene u kojoj se koristi;
- *Agilnost* kao preduvjet korištenja agilnih metoda razvoja programskih rješenja vođenih testiranjem i iteracijama u kojima se shema baze podataka, kao i samo rješenje, konstantno mijenja i nadograđuje [7, str. 9].

3. Dokument baze podataka

Dokument baze podataka jedna su od glavnih kategorija tehnologija za spremanje podataka proizašlih iz NoSQL pokreta, kod kojih temeljnu jedinicu pohrane, kao i sam model podataka, čini dokument.

Definicija dokumenta varira od sustava do sustava, no generalno dokument enkapsulira određene podatke ili informacije u nekom od standardnih formata za enkodiranje podataka poput JSON-a, XML-a, YAML-a i sličnih. Dokumenti su vrlo bliski redovima relacija u relacijskim bazama podataka samo što su atributi te njihove vrijednosti u dokumentima dani kao setovi parova ključ/vrijednost [11]. Velika prednost dokumenata kao jedinica pohrane podataka, kao i temelj njihove popularnosti, pretežito kod programera, je njihova ekvivalentnost s objektima [11]. Naime, kako se objekti, čak i oni proizašli iz iste klase, razlikuju u svojoj internoj reprezentaciji, tako ni jedan dokument u dokument bazi podataka ne treba čvrsto pratiti unaprijed definiranu standardnu shemu odnosno, sastojati se od istih atributa čak i ako se radi o istim klasama entiteta. Sasvim je realno i moguće opisati recimo entitet knjige dvama dokumentima u istoj bazi podataka koji ne dijele ni jedan zajednički atribut.

Iako sustavi za upravljanje dokument bazama podataka (SUDBP) mogu koristiti razne formate za enkodiranje podataka kako bi iste efikasno spremali kao setove parova ključ/vrijednost, JSON format je vodeći i najzastupljeniji format zbog svoje jednostavnosti i općeprihvaćenosti. Često se nadopunjuje određenim funkcionalnostima, poput posebnih tipova podataka, što je primjerice ostvareno u MongoDB sustavu koji dokumente pohranjuje u BSON (engl. *Binary JSON*) formatu, kako bi se podaci što efikasnije spremali u i dohvaćali iz sustava [11]. Na Slici 3 prikazana je razlika između pohrane podataka u relacijsku i dokument, MongoDB, bazu podataka.



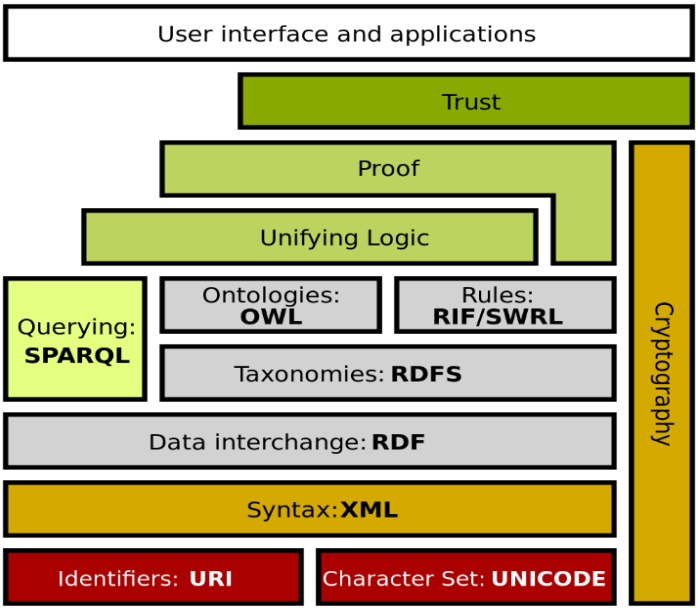
Slika 3. Prikaz razlike u pohrani podataka između relacijske i dokument baze podataka (Izvor: https://webassets.mongodb.com/_com_assets/cms/Relational_vs_DocumentDB-imgngssl17.png)

4. RDF baze podataka

Semantički repozitorij, semantička graf baza podataka, ontološki server, spremnik trojki, samo su neki od sinonima za tehnologiju pohrane podataka proizašlu iz takozvanog semantičkog weba, RDF baze podataka. Karakteristike RDF baza podataka temelje se na određenim standardima te će se zbog toga u nastavku, prije fokusa na same RDF baze, dati osvrt na te standarde te njihovo polazište, odnosno semantički web.

4.1. Semantički web

Semantički web, prema originalnoj viziji Tima Berners-Lee, proširenje je standardnog weba koje omogućuje računalima da razumiju i odgovaraju na kompleksne ljudske zahtjeve na temelju njihova značenja, koristeći izvore informacija koji su semantički strukturirani [12]. Prema danoj definiciji, kako bi računala mogla zadovoljavajuće odgovarati na ljudske zahtjeve, u podatke dostupne na webu potrebno je ugraditi semantičku komponentu odnosno pridružiti im značenje. Semantika pridružena podacima, podržana standardima semantičkog weba danim od strane WWW konzorcija (W3C), omogućuje integriranje i korištenje podataka iz različitih izvora te razumijevanje istih od strane računala. Standardi semantičkog weba promoviraju standardne formate za publikaciju i dijeljenje podataka na webu. Na Slici 4 dan je slojeviti prikaz standardnih tehnologija koje omogućuju postojanje semantičkog weba. Važno je za napomenuti kako se tehnologije međusobno nadopunjuju, a niži slojevi temelj su tehnologijama na višim slojevima.



Slika 4: Prikaz arhitekture semantičkog weba (Izvor: https://upload.wikimedia.org/wikipedia/commons/thumb/f/f7/Semantic_web_stack.svg/800px-Semantic_web_stack.svg)

Iako su standardi uglavnom jedna od stvari koje pridonose popularizaciji određene tehnologije, kao što to slučaj kod razvoja originalnog weba, mnogi autori u području semantičkog weba slažu se kako je velika pogreška i razlog zbog kojeg semantički web nije zaživio još ni danas upravo u naporima u standardiziranju semantičkog weba u njegovim začecima bez da se u obzir uzela stvarna upotrebljivost takvih standardiziranih tehnologija.

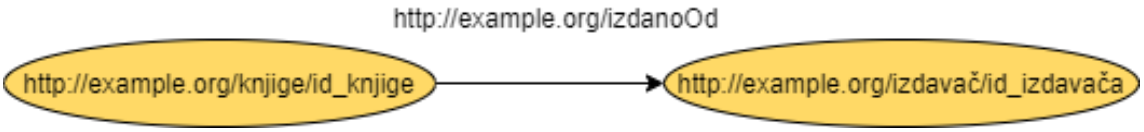
Verzija semantičkog weba, inicijalno osmišljena i predložena od Tima Berners-Leea, koja se nekad činila neizbježnim stanjem weba ne nazire se u bliskoj budućnosti. Usprkos tome, veliki broj tehnologija i ideja koje su inicijalno namijenjene ostvarenju semantičkog weba prenamijenjene su i žive u raznim drugim aplikacijama i proizvodima [13]. Tako se primjerice Google oslanja na semantičke tehnologije, primarno JSON-LD, za generiranje konceptualnih sažetaka rezultata pretraživanja dok facebookov OpenGraph protokol definira shemu, koristeći RDFa (engl. *RDF in attributes*), koju developeri mogu koristiti kako bi sami odlučili o prikazu svoje web stranice kod dijeljenja iste na društvenim mrežama. Zbog navedenog, RDF baze podataka nisu vezane isključivo uz semantički web već se kao tehnologija za spremanje podataka koriste u raznim situacijama gdje se žele iskoristiti njihove prednosti temeljene na RDF modelu podataka.

4.2. RDF model podataka

RDF (engl. *Resource Description Format*) standardni je format, a ujedno i model podataka, za opisivanje, publikaciju i razmjenu informacija na semantičkome webu. RDF je, kao što je već napomenuto, grafovski model podataka kojem su osnovni ciljevi rukovanje nestrukturiranim podacima, što povlači vrlo fleksibilnu shemu podataka i činjenje tih podataka razumljivim kako ljudima tako i računalima [14, str. 12].

Temeljni koncept semantičkog weba predstavlja URI (engl. *Universal Source Identifier*), globalno jedinstveni identifikator resursa na webu neovisno o tome radi li se o fizičkome objektu, konceptu, vezi između njih ili pak atributu koji je resursu pridružen. URI omogućuje povezivanje resursa iz različitih izvora te definiranje osnovne jedinice koja čini RDF model podataka, RDF trojke.

RDF trojka (s, p, o) interpretira se kao izjava gdje je objekt o u vezi p sa subjektom s. U kontekstu izjave, rečenice, s je subjekt, p je predikat, a o objekt. Formalno RDF trojka predstavlja izraz $(U \cup B)\ x\ U\ x\ (U \cup B \cup L\ )$ kojim se osim strukture opisuje i mogući tipovi vrijednosti koje dijelovi trojke mogu poprimiti. Tako subjekt trojke može biti URI ili prazan čvor (engl. *blank node*), predikat može biti isključivo URI, a objekt uz URI i prazan čvor može biti i RDF literal [14, str. 13]. Primjer jednostavne RDF trojke kod koje su svi dijelovi, subjekt, predikat i objekt, predstavljeni jedinstvenim identifikatorima (URI) dan je na Slici 5.



Slika 5: Primjer RDF trojke (Izvor: autorova izrada)

Transformacija bilo koje vrste podataka u RDF trojku laka je i koncizna operacija koja ne rezultira gubljenjem informacija. Zbog svoje jednostavnosti RDF trojke dozvoljavaju povezivanje, publikaciju i dijeljenje strukturiranih, polu-strukturiranih te nestrukturiranih podataka [15, str. 4]. Povezivanjem RDF trojki, kao temeljnih jedinica znanja, dolazimo do koncepta RDF grafa.

RDF graf set je trojki koje povezane čine označeni usmjereni graf gdje čvorovi predstavljaju subjekte i objekte, a označene veze predikate koji povezuju subjekte s objektima. Karakteristike koje ponajviše izdvajaju RDF kao model podataka prema [14, str. 12] su:

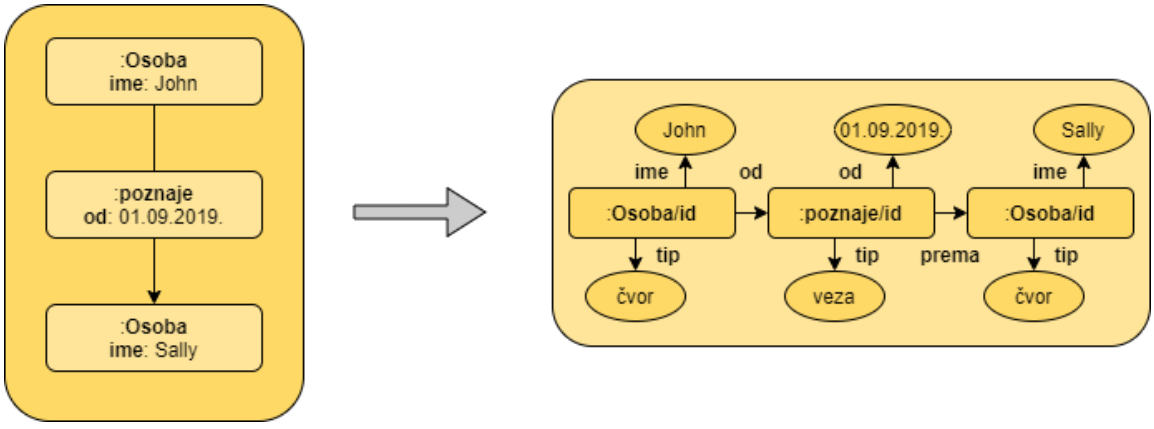
- fleksibilnost u prikazu strukture podataka bez definiranja sheme. Omogućuje povezivanje podataka u strukturu grafa što je vrlo učinkovito kada struktura podataka nije poznata unaprijed. Graf je kao model podataka prirodno aditivan.
- povezivanje različitih izvora podataka dodavanjem RDF trojki koje opisuju veze između tih izvora podataka. Navedeno je puno teže postići u tradicionalnim relacijskim bazama podataka ponajviše zbog nužnosti prilagođavanja sheme podataka.

Kao jedan od dva dominantna modela podataka implementiranih od strane grafovskih baza podataka, drugi je graf sa svojstvima, vrlo važno je istaknuti razlike i sličnosti između njih.

4.2.1. Usporedba RDF-a i grafa sa svojstvima

Kao što je već rečeno, RDF je standard WWW konzorcija za objavu i razmjenu podataka na webu. Kao model podataka izlaže grafovsku strukturu što mu je glavna poveznica s grafom sa svojstvima, a samim time i grafovskim bazama podataka. Graf sa svojstvima kao model podataka nije razvijen imajući na umu standardiziranje objave i prikaza podataka već u svrhu efikasne pohrane i rada s povezanim podacima.

Što se razlika u samoj strukturi tiče, RDF grafu nedostaje karakteristika svojstava pridruženih čvorovima i vezama, odnosno interna struktura [16]. Reprezentacijom podataka kao RDF trojki povezanih u graf podaci se dekomponiraju do najsitnijih dijelova [16], što znači da se sama svojstva entiteta prikazuju kao čvorovi u grafu. Navedeno je najbolje vidljivo kroz jednostavni primjer. Slika 6 prikazuje graf sa svojstvima te RDF graf koji opisuju vezu, poznavanje, između dvoje ljudi.



Slika 6: Razlika između grafa sa svojstvima i RDF grafa (Prema: [6, str.6])

Vidljivo je kako graf sa svojstvima iziskuje manji broj čvorova od trenutka kada se neki od entiteta opisuje svojstvima. Primjerice, ako imamo čvorove s pet svojstava i dvije veze prema drugim čvorovima, za svaki čvor u grafu sa svojstvima imamo 7 RDF trojki. Osim strukturalnih razlika postoje mnogobrojne razlike koje se pojavljuju u modeliranju određenih domena i moraju se imati na umu kod odabira grafovske baze podataka:

- RDF jedinstveno ne identificira instance veza istog tipa [16] što u principu znači da u RDF grafu ne može postojati više veza istog tipa između istog para čvorova. Vezama istog tipa između dva ista čvora ne generiraju se dodatne informacije to jest radi se o istoj RDF trojci [16]. Navedeno je karakteristika poželjna u RDF, no neadekvatna u grafovskim bazama podataka.
- Prethodna razlika povlači nemogućnost pridruživanja atributa vezama u RDF modelu podataka [16]. U svrhu nadilaženja problema koji se pojavljuju u sklopu nemogućnosti kvantificiranja veza između čvorova u RDF grafu, koristi se takozvani koncept reifikacije. Reifikacija predstavlja postupak davanja izjava o izjavama, odnosno tretiranje RDF trojki kao čvorova.
- Ukoliko određena svojstva čvora u grafu poprimaju više vrijednosti, u RDF graf uvode se nove RDF trojke s objektima koji odgovaraju tim vrijednostima. Graf sa svojstvima koristi liste vrijednosti kao vrijednost danog svojstva.

Razlike između dominantnih modela grafa opet nas vraćaju na ideje koje se nalaze u pozadini njihova razvoja. RDF model podataka nije razvijen imajući na umu da će se koristiti kao model implementiran od strane grafovskih baza podataka i zbog toga se nedostatci, od kojih su neki dani kroz navedene razlike, vrlo često rješavaju nadopunjavanjem RDF modela karakteristikama grafa sa svojstvima, o čemu će više riječi biti u nastavku rada. Transformiranje RDF grafa u graf sa svojstvima rijetko se provodi zbog kompleksnosti operacije. Najveći problem predstavljaju predikati RDF trojki, koji su predstavljeni URI-jima i koji bi se normalno transformirali u veze, no to bi dovelo do moguće pojave nepovezanog grafa. Zbog toga se veze, predikati, mapiraju u čvorove te onda povezuju s drugim čvorovima [6, str. 6]. Suprotna pretvorba, grafa sa svojstvima u RDF graf, vrlo je jednostavna i vrlo često se u usporedbama, kao što je ova, navodi kao prednost nativnih grafovskih baza podataka.

Kao što je vidljivo na slici arhitekture semantičkog weba, RDF je format i model podataka na kojem se temelje izražajnije tehnologije za opis podataka te modeliranje domena. Sintaksu RDF-a nadopunjavaju konstruktima koji omogućuju modeliranje formalnih ontologija kao sheme, odnosno strukture, RDF baza podataka.

4.3. Formalna ontologija

Formalna se ontologija može smatrati nekom vrstom minijaturnog jezika s vrlo limitiranim setom imenica i glagola kojima se na standardiziran način opisuje određena domena. Sintaksa tog jezika omogućuje reprezentaciju znanja i odnosa u domeni u obliku rečenica strukture S-P-O (Subjekt-Predikat-Objekt) koje se direktno preslikavaju u prirodni jezik [17]. Preslikavanje je namjerno te omogućuje da podaci predstavljeni tom strukturom budu čitljivi od strane ljudi, a isto tako zbog svoje formalnosti budu razumljivi računalima. Uz efikasno rješavanje problema formalizacije opisa domene, formalne ontologije rješenje su i drugog problema semantičkog weba, heterogenosti podataka to jest povezivanja podataka iz različitih izvora. Naime, formalnim se ontologijama nastoji razviti shema podataka koja se istovremeno može primijeniti na sve i ništa u određenoj domeni koju opisuje [17]. Dvije ključne stvari koje ontologija prema [17] mora ostvariti su:

1. Precizno i točno prikazati informacije od interesa u određenoj domeni te veze koje korisnici žele pratiti između entiteta koji se u toj domeni pojavljuju.
2. Pružiti koncept klasa te veza između klasa u svrhu kreiranja struktura koje dozvoljavaju reprezentaciju ne samo poznatih i preciznih, već i generalnih te manje poznatih informacija.

Primjena formalne ontologije kao sheme podataka, kojom se definira struktura RDF baze podataka, omogućena je RDF formatom podataka koji logičku strukturu formalnih ontologija u obliku izjava S-P-O implementira RDF trojkama. RDF format vrlo je efikasan u prikazu veza između resursa i njihovih svojstava kao što je navedeno u točki 1, no ne omogućuje modeliranje veza između samih subjekata i objekata, kao što ne omogućuje ni prikaz strukture bilo kojeg dijela RDF trojke, zahtjev točke 2 [12]. Kako bi RDF i dalje ostao jednostavan model podataka, njegova sintaksa nadopunjena je tehnologijama čiji se koncepti prirodno predstavljaju u RDF formatu, a u isto vrijeme ga nadopunjuju s ciljem modeliranja formalnih ontologija.

4.4. RDF Shema

RDFS ili RDF Shema (engl. *Resource Description Format Schema*) semantička je ekstenzija RDF-a koja pruža mehanizme za opis grupa povezanih resursa i veza između njih [18]. Vrlo često se definira kao rječnik koji se nastavlja na RDF format te uvodi sintaksu koja omogućuje definiranje taksonomija (hijerarhija) te ono najvažnije, omogućuje izricanje veza između samih subjekata kao i između objekata.

Ključne riječi definirane u tom rječniku dane su od strane WWW konzorcija u području imena (engl. *namespace*) formalno nazvanim *rdfs* [18], a neke od njih su: *rdfs:Class* kojom se definira klasa resursa i *rdfs:range* ključna riječ kojom se vrijednosti određenog svojstva definiraju kao instance jedne ili više klasa. U sljedećem primjeru prikazano je korištenje RDFS-a s fokusom na prikazu osnovnih prednosti koje RDFS ostvaruje u odnosu na RDF.

```
@prefix ex1: <http://example1.com/ontology#> .
@prefix ex2: <http://example2.com/ontology#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
ex1:imaKćer rdfs:subPropertyOf ex1:imaDijete .
ex1:imeKćeri rdfs:subPropertyOf ex2:imeDjeteta .
```

Nakon definiranja *ex1:, ex2:* i *rdfs:* područja imena dane su dvije RDF izjave sa RDFS ključnom riječju *subPropertyOf* kojom:

- definiramo ***ex1:imaKćer*** kao pod-svojstvo *ex1:imaDijete* što nam omogućuje da kao rezultate pretrage RDF baze podataka za RDF trojkama koje koriste svojstvo *ex1:imaDijete,* dobivamo i RDF trojke sa svojstvom ***ex1:imaKćer.***
- definiramo ***ex1:imeKćeri*** kao pod-svojstvo *ex2:imeDjeteta* čime negiramo potrebu da podaci iz različitih izvora koriste istu shemu.

Vrlo važna posljedica korištenja RDFS-a je i mogućnost korištenja RDFS alata za zaključivanje kako bi se, koristeći logičko zaključivanje, izvodile nove RDF trojke koje nisu eksplicitno pohranjene u spremniku podataka. Navedeno svojstvo zaključivanja jedna je od temeljnih karakteristika RDF baza podataka te će kao takva biti detaljnije opisana u sljedećom poglavlju u kojem će fokus biti na jeziku za modeliranje ontologija OWL-u.

4.5. OWL

Iako je RDFS dostatna tehnologije za opis manjih ontologija, njena vrlo važna karakteristika je to što ne sprječava unos izjava u bazu podataka. Naime, moguće je istovremeno tvrditi da je neki resurs klasa ali i instanca klase [19]. OWL (engl. *Web Ontology Language*) jezik rječnik je poput RDFS-a ali uvodi puno rigidniju strukturu za modeliranje ontologija iz čistog razloga što je osmišljen za korištenje od strane računala koja mogu efikasno raditi samo sa koncizno definiranom sintaksom te pravilima. Zaključivanje, odnosno izvođenje znanja koje nije eksplicitno navedeno u RDF trojkama osnovna je i najvažnija funkcionalnost ostvarena korištenjem OWL-a.

4.5.1. OWL zaključivanje

OWL alati za zaključivanje programi su koji na temelju pravila zaključivanja danih u OWL rječniku odnosno sintaksi i deskriptivnoj logici izvode logičke posljedice, RDF trojke i veze između njih, iz seta eksplicitno navedenih RDF trojki. Uglavnom, ontologije pa samim time i OWL alati za zaključivanje, optimizirani su za probleme zaključivanja nad taksonomijama, a vrlo važnu ulogu zaključivanje nad ontologijama ima i u otkrivanju njene moguće nekonzistentnosti [20]. Sljedeći primjer prikazuje tipičan proces zaključivanje nad činjenicama sadržanima u ontologiji.

Recimo da ontologija sadrži činjenice: *Pas je Sisavac* i *Lilly je Pas.* Alat za zaključivanje u bazu znanja ili kao odgovor na upit, na temelju pravila zaključivanja i pravila danih u OWL rječniku, može dodati ili vratiti kao rezultat činjenicu: *Lilly je Sisavac.*

Kako je uloga OWL-a dvojaka, kompromis između ekspresivnosti istog kao jezika za modeliranje ontologija i efikasnog zaključivanja koje iziskuje određene računalne resurse doveo je do definiranja nekoliko OWL profila koji pogoduju specifičnim scenarijima, a žrtvuju ekspresivnost za efikasno zaključivanje ili suprotno [21].

- OWL 2 / Full profil dopušta korištenje svih konstrukata, cijelog OWL rječnika, dostupnog u OWL 2 specifikaciji. Najekspresivniji je profil i ne preporuča se uz korištenje zaključivanja zbog vrlo kompleksnih pravila zaključivanja [21].
- OWL 2 / EL omogućuje efikasno zaključivanje nad taksonomijom ontologije [21]. Ekspresivniji je od profila koji slijede jer iziskuje definiranje složenih veza između klasa i svojstava.

- OWL 2 / QL žrtvuje nepotrebnu ekspresivnost u svrhu efikasnog pretraživanja i rada sa instancama klasa [21].
- OWL 2 / RL posebno je namijenjen za rad nad relacijskom bazom podataka, odnosno sa sustavom koji implementira poslovnu logiku [21]. Naglasak je zaključivanju, a ekspresivnost je svedena na minimum.

Prije OWL 2 specifikacije, OWL 1 specifikacija drugačije je definirale OWL profile u tri grupe : OWL / Lite, OWL / DL i OWL / Full. Navedeni profili, iako drugačije nazvani, i dalje se međusobno razlikuju u kompromisu između ekspresivnosti i efikasnosti u zaključivanju. OWL 2 profili često se smatraju i detaljizacijom OWL / DL profila.

Iako definirani kao standardi, OWL profili vrlo su često samo parcijalno implementirani u proizvode, dok neki proizvodi dolaze s vlastitim implementacijama alata za zaključivanje koji koriste samo određeni podskup OWL rječnika, kao što je to slučaj s AllegroGraph sustavom.

4.6. SHACL

Vrlo važna, moglo bi se reći i ključna, karakteristika opisanih jezika za modeliranje RDF-a, RDFS-a i OWL-a njihova je temeljenost na takozvanoj pretpostavci otvorenog svijeta (engl. OWA - *Open World Assumption*). Posljedica navedene pretpostavke dizajn je aplikacija kod kojih nepostojanje određene izjave ne znači da je ista lažna već suprotno, istinita je sve dok se ne dokaže suprotno [22]. Primjerice, ako OWL ontologija sadrži činjenicu kako je *rdfs:range* svojstva (predikata) *ex:imaOca* klasa *ex:Osoba* i aplikacija vidi samo trojku *ex:John ex:imaOca ex:Bob* i ni jednu drugu trojku, prema pretpostavci otvorenog svijeta aplikacija ne smije pretpostaviti kako *ex:Bob* nije instanca klase *ex:Osoba,* jer to nije eksplicitno izrečeno. Zapravo, aplikacija bi trebala moći automatski zaključiti odnosno izvesti trojku *ex:Bob rdf:type ex:Osoba.* Navedeni pristup dizajnu, iako dolazi s velikom prednošću mogućnosti povezivanja podataka iz različitih izvora, onemogućuje kvalitetnu validaciju podataka koji ulaze u RDF bazu podataka, a samim time njihovo korištenje u domenama koje karakteriziraju striktna poslovna pravila.

Do pojave SHACL-a (engl. *Shapes Constraints Language*) nije postojao standardizirani mehanizam za definiranje ograničenja nad podacima kojima bi se osigurala konzistentnost RDF podataka, odnosno kojim bi se sprovela validacija RDF grafa na temelju seta pravila. Većinom se pretpostavka otvorenog svijeta ignorirala, a alati kao primjerice Protégé, restrikcije poput *owl:maxCardinality 1* implementirali su doslovno kao restrikciju koja onemogućuje unošenje više od jedne vrijednosti, iako u OWL svijetu navedena restrikcija omogućuje da

svojstvo ima više vrijednosti samo što se te vrijednosti u tom slučaju smatraju istima [22]. SHACL zatvara svijet rječnikom pravila koja omogućuju validaciju RDF trojki što znači da se istinom smatra samo ono što je eksplicitno navedeno. Cijeli postupak validacije podataka korištenjem SHACL-a svodi se na usporedbu dvaju RDF grafova. Prvi graf, takozvani graf uvjeta (engl. *shapes graph*), sadrži pravila koja drugi graf koji sadrži podatke, graf podataka, mora zadovoljiti da bi se mogao opisati kao validan [23]. Kako SHACL u grafu uvjeta, izrađenog temeljem sintakse dane u SHACL rječniku, daje deskripciju podataka, a samim time i njihovu shemu, na njega se može gledati kao na jezik za modeliranje ontologija baš kao što je to i OWL. Pitanje koje se nameće je odabir između OWL-a i SHACL-a kao jezika za modeliranje sheme RDF baze podataka.

Tablica 2:Usporedba OWL-a i SHACL-a

OWL	**SHACL**
Bazira se na pretpostavci otvorenog svijeta	Bazira se na pretpostavci zatvorenog svijeta
Dizajniran je za zaključivanje nad podacima	Omogućuje zaključivanje nad podacima definiranjem velikog broja ad-hoc pravila i restrikcija
Validacija podataka djelomično postignuta kroz zaključivanje	Dizajniran za validaciju podataka
Služe za dokumentiranje RDF-a	

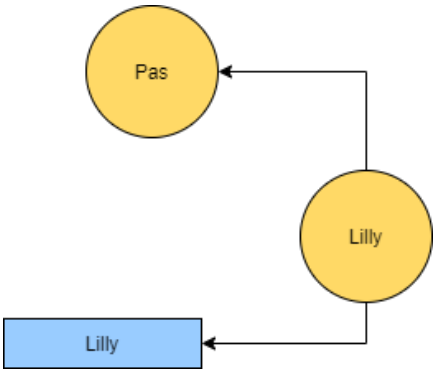
(Prema: https://stackoverflow.com/questions/44767223/owl-property-restrictions-vs-shacl)

Iz dane usporedbe u Tablici 2 vidljivo je kako su SHACL i OWL tehnologije i standardi nastali iz različitih potreba. Iako se SHACL na temelju karakteristika može koristiti kao jezik za modeliranje ontologija, to nije čest slučaj zbog većeg broja alata, primjera i iskoristivih OWL ontologija. Ono što je poželjno i zamišljeno standardizacijom SHACL-a, kombiniranje je njegovih individualnih snaga sa snagama OWL-a [22]. Implementacijom sustava u kojem RDF trojke izvedene OWL zaključivanjem nakon dodavanja u RDF graf prolaze validaciju korištenjem SHACL-a, primjer je česte i dobre prakse.

4.7. Serijalizacije RDF-a

RDF trojke povezane u strukturu grafa potrebno je na neki način reprezentirati te prenositi između informacijskih sustava koji ih koriste. Serijalizacija modela podataka omogućuje enkodiranje podataka u određeni format kako bi se isti kasnije u određenome trenutku i za određenu svrhu rekonstruirali u originalnu strukturu. Za razliku od mnogih drugih modela podataka, RDF nije vezan uz jedinstveni format serijalizacije, iako je to izvorno planirano kroz još jedan standard WWW konzorcija, XML. Odabir formata za serijalizaciju RDF-a ponajviše ovisi o preferencijama korisnika te o podržanosti formata od strane sustava koji se koristi, dok se najveće razlike između formata očituju u čitljivosti od strane korisnika i kompatibilnosti formata za specifične domene. U nastavku slijedi opis četiri najpopularnija formata za serijalizaciju RDF-a u svrhu davanja uvida u njihove sličnosti i razlike.

Uzmimo za primjer slučaj u kojem su u RDF bazi podataka spremljene dvije RDF trojke koje čine strukturu danu na Slici 7. Krugovi predstavljaju resurse definirane URI-jima, četverokuti predstavljaju resurse dane kao literale, dok su strelicama prikazani predikati. Za svaki opisani format prikazat će se primjer serijalizacije danog RDF grafa.



Slika 7: Jednostavan primjer RDF grafa (Izvor: autorova izrada)

RDF/XML najstariji je format za serijalizaciju RDF-a proizašao kao standard WWW konzorcija. U današnjim sustavima uglavnom je podržan kao zadani (engl. *default*) ulazni i izlazni format iz sustava [24].

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex1="http://example.com#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

  <ex1:Pas rdf:about="http://dbpedia.org/resource/Lilly">
    <rdfs:label xml:lang="en">Lilly</rdfs:label>
  </ex1:Pas>
</rdf:RDF>
```

N-Triples jednostavan je format u kojem svaka linija koja završava s točkom predstavlja RDF trojku. Vrlo je robustan format koji se efikasno komprimira. Kako je RDF graf u N-Triples formatu reprezentiran nizom linija koje predstavljaju RDF trojke i čiji poredak ne utječe na validno rekonstruiranje RDF-a, vrlo se često koristi u dizajniranju API-ja (engl. *Application Programming Interface*) koji vraćaju RDF podatke [24].

```
<http://dbpedia.org/resource/Lilly>

<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>

<http://example.com /Pas> .

<http://dbpedia.org/resource/Lilly>

<http://www.w3.org/2000/01/rdf-schema#label>
"Lilly"@en.
```

Turtle je format sličan RDF/XML uz prednost bolje čitljivosti. Za razliku od N-Triples formata, RDF trojke nisu dane u linijama već u blokovima gdje svaki blok predstavlja dio RDF grafa, odnosno veći broj RDF trojki [24].

```
@prefix dbr:     <http://dbpedia.org/resource/> .
@prefix dbo:     <http://dbpedia.org/ontology/> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#>
.
@prefix ex1:    <http://example.com/> .

dbr:Lilly
a ex1:Pas ;
rdfs:label "Lilly"@en .
```

JSON-LD format, kao i RDF/XML, pokušaj je prilagođavanja postojećeg formata, JSON-a, za pohranu RDF-a kako bi se iskoristile njegove karakteristike i postojeći alati za manipuliranje podacima.

```
[
  {
    "@id": "http://dbpedia.org/resource/Lilly",
    "@type": [
      "http://xmlns.com/Pas"
    ],
    "http://www.w3.org/2000/01/rdf-schema#label": [
      {
        "@value": "Lilly",
        "@language": "en"
      }
    ]
   }
]
```

4.8. Pretraživanje RDF grafa – SPARQL

Baš kao što SQL omogućuje rad s podacima pohranjenima u relacijske baze podataka, SPARQL, rekurzivni akronim za SPARQL protokol i RDF upitni jezik (engl. *SPARQL Protocol and RDF Query Language*), osigurava tu istu funkcionalnost RDF bazama podataka. SPARQL je, kao i SQL, standardizirani deklarativni upitni jezik. Sintaksno su ta dva jezika vrlo slična, no kako su temeljeni na potpuno različitim modelima podataka, u pozadini, na vrlo različite načine pristupaju manipulaciji podacima. SPARQL upitni jezik koristi se za rad s podacima pohranjenim u RDF modelu podataka koji je, kao što je već prije spomenuto, grafovski model podataka te se stoga upiti baziraju na pronalasku podudarajućih uzoraka grafa odnosno, sama sintaksa jezika prilagođena je pretraživanju grafa [25, str. 6].

SPARQL na podatke gleda u obliku usmjerenog, označenog grafa koji je interno predstavljen kao skup trojki. Posljedica toga je da se upiti sastoje od seta uzoraka trojki u kojima svaki element trojke, bio on subjekt, predikat ili objekt, može biti varijabla. Zamjena tih varijabli konstantama svrha je upita i vrši se traženjem RDF trojki u bazi podataka koje se podudaraju s uzorcima danima u upitu. Navedeno predstavlja najveću snagu SPARQL upitnog jezika, navigaciju vezama u RDF grafu kroz traženje uzoraka. U tom procesu jednostavni uzorci mogu se kombinirati u kompleksnije te se na taj način mogu istražiti i najsloženije veze između podataka [26] . U SPARQL-u postoje četiri vrste upita:

- *Ask* upit kao rezultat vraća odgovor na pitanje postoji li barem jedno podudaranje između uzorka danog u upitu te RDF podataka u spremniku to jest bazi podataka,
- *Select* upit vraća podatke proizašle iz podudaranja uzorka i podataka u tabularnoj formi,

- *Construct* upit, kao što mu samo ime govori, kao rezultat vraća konstruirani RDF graf dok,
- *Describe* upit vraća podudaranja proizašla iz konstruiranja relevantnog RDF grafa

U navedene vrste upite moguće je uvesti niz drugih operatora iz SPARQL sintakse te na taj način vrlo efikasno pretraživati RDF bazu podataka. U nastavku je dan primjer *Select* upita koji vraća maksimalno deset distinktnih RDF trojki, sortiranih prema vrijednosti predikata, krećući od pete pronađene trojke. Varijable u upitu započete su upitnikom (?) te će prema tome rezultat svakog uspješnog podudaranja uzorka, danog u *Where* bloku upita, biti RDF trojka.

```
PREFIX ex1: <http://example.com/ontology#>
SELECT ?subject ?a_predicate ?an_object
WHERE {
    ?subject ex1:nalaziSeU
    ?object FILTER regex(?object, "Varaždin") .
    ?subject ?a_predicate ?an_object . }
    DISTINCT
    ORDER BY ?a_predicate ?an_object
    LIMIT 10
    OFFSET 5
```

4.9. Definicija RDF baze podataka

Sumirano, prethodno opisane tehnologije i njihove međusobne poveznice svode se na sljedeće. RDF je model podataka za publiciranje i razmjenu podataka na webu. RDFS i OWL jezici su za modeliranje njegove sheme, dok je SPARQL upitni jezik kojim se manipulira RDF grafom te nad istim postavljaju upiti. RDF baze podataka nazivane i spremnicima trojki, tehnologija su za spremanje i manipulaciju činjenicama u RDF formatu. U idealnome slučaju RDF baza podatka omogućuje korištenje i drugih prethodno navedenih tehnologija koje se nadovezuju na RDF model podataka. To u dostupnim RDF bazama podataka najčešće nije slučaj već se svaka prilagođava određenoj domeni te u skladu s time implementira tehnologije potrebne za efikasan rad s podacima koji se u toj domeni pojavljuju. Primjer toga je i AllegroGraph baza podataka koja samo djelomično nudi podršku za OWL zaključivanje [27].

Iz navedenog razloga teško je obuhvatiti opće karakteristike RDF baza podataka koje proizlaze iz samog RDF-a kao modela podataka kojeg implementiraju te ostalih navedenih tehnologija. U nastavku su ipak dane neke od karakteristika RDF baza podataka koje ih ponajviše odvajaju kako od relacijskih tako i od NoSQL rješenja.

- Fleksibilnost sheme očigledna je prednost RDF baza podataka koja se temelji na RDF modelu podataka. Iako je modeliranje sheme moguće i preporučljivo u svrhu

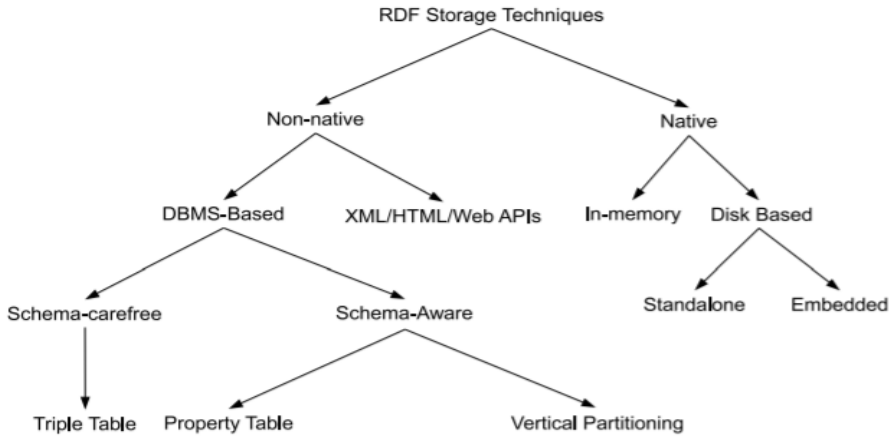
kasnijeg izvođenja novog znanja iz postojećih podataka, u mnogim slučajevima inkrementalno dodavanje RDF trojki u bazu i njihovo povezivanje u postojeći RDF graf ne zahtijeva postojanje detaljno definirane sheme.

- Standardi semantičkog weba pridonose jednostavnosti zamjene postojeće RDF baze podataka novom te migracije podataka između baza podataka [15, str. 5].
- Izvođenje novih činjenica na temelju postojećih naziva se zaključivanje te je jedna od najvažnijih karakteristika RDF baza podataka. Zaključivanje se temelji na jezicima za modeliranje ontologija, kao RDFS i OWL, ili pak na podskupovima njihove sintakse.

Manipulacija podacima pohranjenima u RDF modelu podataka podrazumijeva određene implementacijske izazove u vidu pohrane podataka, indeksiranja te procesiranja upita [14, str. 16].

4.9.1. Pohrana RDF grafa

Kao što su grafovske baze podataka podijeljene na nativne i nenativne u ovisnosti o njihovoj prilagođenosti radu s modelom grafa, tako se i RDF baze podataka dijele na nativne i nenativne ovisno o tome bazira li se njihova shema pohrane podataka na postojećim tehnologijama za pohranu podataka poput relacijskih baza podataka ili je ta shema razvijena isključivo za pohranu RDF modela podataka. Taksonomija pristupa pohrani RDF grafova dana je na Slici 8:



Slika 8. Klasifikacija pristupa pohrani RDF podataka (Izvor: [14, str. 17])

Prema [14, str. 20] i implementacijama, na tržištu najpopularnijih RDF baza podataka poput Sesame, Jena2 i 3Store, najbolji pristup pohrani RDF podataka je spremanje istih u relacijsku bazu podataka zbog dobro utemeljenih performansi iste. Postoje tri osnovne tehnike spremanja RDF trojki u relacije koje se razlikuju u tome stavlja li se fokus na performanse kod postavljanja upita ili ažuriranja baze podataka.

- Relacija trojki (engl. *Triple table*) izravan je pristup mapiranju RDF modela podataka u relacije. Svaka RDF trojka pohranjena je u jednu jedinstvenu relaciju čija se shema sastoji od tri stupca (stupac za subjekt, predikat i objekt). Postavljanje upita nad takvom tablicom, čak i s indeksiranim stupcima, može biti vrlo spora operacija zbog velikog broja spajanja nad tom tablicom (engl. *join*) kako bi se zadovoljio dani uzorak RDF grafa u upitu [14, str. 21].
- Relacija svojstava (engl. *Property table*) uvedena je s idejom ubrzavanja upita na način da se dozvoli dohvaćanje većeg broja trojki koje referenciraju isti subjekt, samim time uklanjajući potrebu za spajanjima nad stupcem koji predstavlja subjekt. Reprezentacija RDF-a je u ovome pristupu puno bliža tradicionalnim relacijskim shemama s većim brojem relacija, a temelji se na pronalasku klastera subjekata koji se često pojavljuju s istim predikatima. Veliki nedostatak ovog pristupa je što ograničava prirodnu fleksibilnost RDF sheme zahtijevajući uvođenje relacija za svaki novi predikat [14, str. 21].
- Vertikalno particioniranje (engl. *Vertical partitioning*) alternativa je relaciji svojstava u smislu lakše implementacije. Svaka relacija trojke vertikalno je particionirana na *n* dvostupčanih relacija gdje je *n* broj jedinstvenih predikata u skupu podataka [14, str. 22].

Osnovni argument u prilog korištenja nativnih pristupa pohrani RDF podataka činjenica je da se nedostatci relacijskih baza podataka, poput rigidne sheme, propagiraju na sam RDF model podataka.

4.9.2. Problem indeksiranja RDF baze podataka

Pristup podacima u bazi podataka pod utjecajem je određenog broja faktora. Najvažniji su zasigurno fizički medij na kojem su podaci pohranjeni te sama količina pohranjenih podataka. Jedan od pristupa kojim se ubrzava pristup podacima, odnosno čitanje iz baze podataka nauštrb pisanju u istu i zauzeću prostora, kreiranje je podatkovnih struktura, indeksa. U principu, indeksi omogućuju da se podaci u bazi podataka lociraju bez potrebe za pretragom svih zapisa pohranjenih u istoj.

U nenativnim RDF bazama podataka koje RDF graf ne mapiraju u druge vrste baza podataka, poput relacijskih, indeksiranje je najčešće postignuto postojećim tehnikama indeksiranja, dok je u nativnim i nekim nenativnim RDF bazama podataka indeksiranje prilagođeno samome RDF modelu podataka te se kao takvo temelji na ideji multi-indeksiranja [14, str. 15].

Multi-indeksiranje pristup je kod kojeg se RDF baza podataka indeksira setom od šest osnovnih indeksa koji pokrivaju sve ulazne sheme RDF upita. To su redom PSO, POS, SPO, SOP, OPS, i OSP indeks (gdje je S subjekt, P predikat ili svojstvo, a O objekt RDF trojke) [14, str. 23]. Zbog prirode RDF trojki pristup multi-indeksiranja ne rezultira kombinatoričkom eksplozijom što bi bio slučaj kod upotrebe ovog pristupa nad relacijama u relacijskoj bazi podataka. Prednost multi-indeksiranja RDF podataka ujedno je i problem indeksiranja RDF baza podataka. Mogućnost prilagodbe indeksa vlastitim RDF podacima istodobno korisnicima daje slobodu u indeksiranju ali, u slučaju velike količine podataka, negativno utječe na korištenje memorijskog prostora. Primjerice, ako je fokus upita na svojstvima koja povezuju subjekte trojki s objektima, bilo koji indeks koji trojke sortira prvo po predikatu, započinje sa P kao primjerice PSO ili POS, efikasno će ubrzati pretraživanje baze. Ako pak korisnik nema dominantnu formu upita i ne može iskoristiti prednosti multi-indeksiranja morat će trojke indeksirati većim brojem indeksa što zauzima memorijski prostor.

4.9.3. Usporedba relacijskih, grafovskih, dokument i RDF baza podataka

Prije prelaska na glavni dio ovog rada, opis AllegroGraph sustava te modeliranje podataka u istome, potrebno je sumirati osnovne karakteristike tehnologija za pohranu podataka koje sustav obuhvaća. Tako je u Tablici 3 taj sažetak dan kroz usporedbu relacijskih baza podataka, najzastupljenijeg pristupa pohrani podataka na tržištu, s grafovskim, RDF i dokument bazama podataka koje su tehnologije implementirane u AllegroGraph sustav. Karakteristike odabrane za usporedbu opće su karakteristike baza podataka te se kao takve mogu izdvojiti i prepoznati kod bilo koje njene varijacije. Isto tako, predstavljaju najefikasniji način prikaza različitosti između baza podataka.

Tablica 3: Usporedba relacijskih, grafovskih, RDF i dokument baza podataka

	Relacijske baze podataka	Grafovske baze podataka	RDF baze podataka	Dokument baze podataka
Model podataka	Relacijski	Graf. Tipično označeni graf sa svojstvima iako su moguće varijacije s obzirom na same karakteristike grafa kao što je usmjerenost	RDF. Tip graf modela podataka	Dokument
Pohrana podataka	Relacije	Podaci se kod nativnih grafovskih baza podataka, spremaju u strukture optimizirane za spremanje grafa. Nenativne graf baze podataka koriste strukture podataka drugih tehnologija za pohranu podataka, primjerice relacije	Kao i kod grafovskih baza podataka, nativne RDF baze podataka optimizirane su za spremanje RDF podataka, dok se nenativna rješenja oslanjanju na postojeće tehnologije pohrane podataka	Dokumenti

Fleksibilnost sheme	Shema je vrlo rigidna i potrebno ju je definirati prije samog početka unosa podataka u bazu. Kasnije promjene stavljaju bazu u nedostupno stanje	Vrlo fleksibilna shema podataka koju u principu nije potrebno definirati unaprijed te ju je moguće mijenjati i nadopunjavati kroz životni ciklus baze podataka.	Fleksibilna shema implementirana u podatke. Definiranje sheme omogućeno je kroz jezike za modeliranje ontologija	Vrlo fleksibilna shema podataka
Upitni jezik	SQL je standardni upitni jezik	Ne postoji standardni upitni jezik iako dobar dio grafovskih baza podataka podržava Gremlin upitni jezik	SPARQL je W3C standardni upitni jezik za rad s RDF modelom podataka. U upotrebi su i nestandardizirani jezici poput RQL-a i TRIPLE-a	Ne postoji standardni upitni jezik već svaki sustav korisnicima nudi vlastiti API za provođenje CRUD operacija nad dokumentima
Procesiranje upita	Indeksi	Nativne grafovske baze koriste tzv. Indeksiranje bez susjedstva čime eliminiraju potrebu za indeksima	Indeksi	Indeksi

5. Sustav za upravljanje bazama podataka AllegroGraph

AllegroGraph je multi-model sustav za upravljanje bazama podataka, kao i okvir za izradu kako semantičkih web aplikacija tako i poslovnih transakcijskih rješenja. Proizašao je iz razvoja tvrtke Franz Inc. krajem prošlog desetljeća na zahtjev Ministarstva Obrane Sjedinjenih Američkih Država (engl. *U.S. Department od Defence*.) u svrhu izgradnje grafa znanja kao temeljnog izvora podataka za donošenja krucijalnih odluka [28]. Danas se koristi u mnogim velikim projektima kao na primjer TwitLogic projektu s ciljem uvođenja semantičkog weba u Twitter podatke. Pisan je u posebnome dijalektu programskog jezika Lisp, Allegro CL (engl. *Allegro Common Lisp*), koji je također ponikao iz Franz Inc. tvrtke. Sama tvrtka utemeljena je 1984. godine i do danas je ostala jedna od vodećih kompanija koja promovira Lisp programski jezik te razvija programsku podršku u istome.

U današnje vrijeme multi-model baze podataka definitivno više nisu rijetkost i opće su prihvaćena praksa na tržištu. To dokazuje i trenutno stanje na ljestvici popularnosti SUBP [29] prema kojoj je čak dvanaest od dvadeset najbolje rangiranih sustava okarakterizirano kao multi-model rješenje. Prednosti multi-model baza podataka uglavnom se temelje na fleksibilnosti takvog sustava i mogućnosti korištenja u različitim domenama, a vrlo često su nuspojava malih razlika između samih modela podataka koje obuhvaćaju. Najnovija 7.1 verzija AllegroGraph baze podataka tako se u kontekstu multi-model sustava svrstava u nenativne grafovske baze podataka, jer implementira RDF model podataka koji je, kao što je već prethodno rečeno, vrsta graf modela podataka, te dokument baze podataka jer omogućuje spremanje JSON i JSON-LD dokumenata [30].

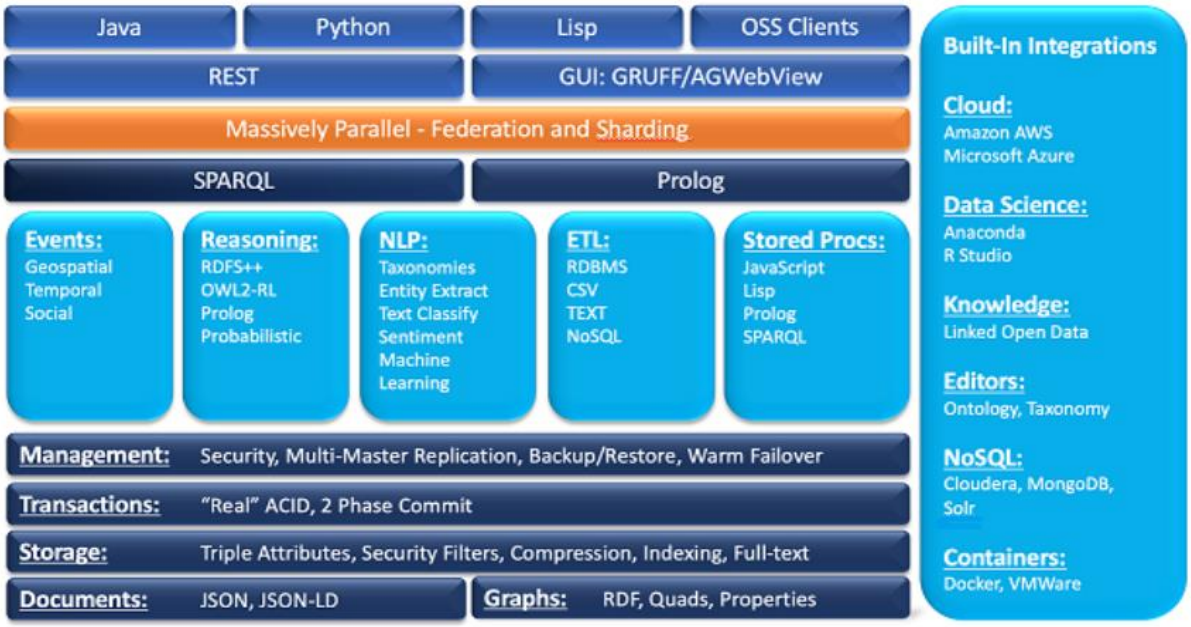
AllegroGraph rješenje je zatvorenog koda što je i logično s obzirom na broj funkcionalnosti i podršku koju nudi svojim korisnicima, a od kojih poseban naglasak, prema [30] stavlja na:

- Entitet – Događaj graf znanja (engl. *Entity – Event Knowledge Graph*) koji predstavlja novi model podataka kao spoj prethodno spomenutih RDF i dokument modela kojim se nastoji spojiti tipične transakcijske podatke nastale poslovanjem s izvorima znanja poput ontologija, taksonomija i sličnog u svrhu generiranja znanja.
- *FedShard* funkcionalnost kao patentirani pristup horizontalnom skaliranju baze podataka kojim se upiti transparentno provode na većem broju izvora podataka i

- tri nestandardne ekstenzije, neovisne o RDF-u i s njime povezanim tehnologijama koje omogućuju rad s geoprostornim i temporalnim podacima te analizu društvenih mreža.

5.1. Arhitektura AllegroGraph SUBP

Slika 9 prikazuje gradivne blokove koji čine AllegroGraph SUBP odnosno njegovu klijent – server arhitekturu.



Slika 9, Arhitektura AllegroGraph sustava (Izvor: https://franz.com/agraph/support/documentation/current/ag-arch.png)

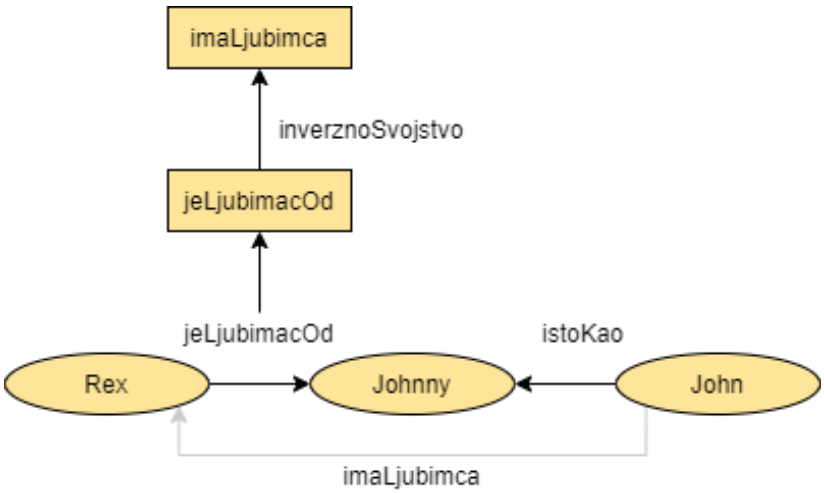
Pristup AllegroGraph bazi podataka, odnosno komunikacija s AllegroGraph serverom, temelji se na REST (engl. *Representational State Transfer*) arhitekturi odnosno HTTP (engl. *Hypertext Transfer Protocol*) protokolu [31]. Svaka od dostupnih klijentskih biblioteka, radilo se o Javi, C#-u ili Pythonu (samo neke od dostupnih biblioteka), u pozadini pristupa AllegroGraph bazi podataka koristeći standardne HTTP metode (GET, POST, PUT, PATCH i DELETE) [31]. Pristup bazi podataka omogućen je i kroz integrirano grafičko sučelje AGWebView, dok je vizualizacija samog grafa podataka dana kroz integrirani alat za vizualizaciju grafa, Gruff. Rad s navedenim alatima biti će prikazan u nastavku rada u praktičnom dijelu.

Postavljanje upita nad AllegroGraph bazom podataka omogućeno je kroz standardni RDF upitni jezik, SPARQL te Franz Inc. implementaciju Prologa. Kod korištenja biblioteka, SPARQL upiti su najčešće automatski generirani dok se kod direktne komunikacije sa serverom putem HTTP protokola oni prenose u obliku parametara u zahtjevu [31].

Na nižim slojevima arhitekture AllegroGrapha implementirane su tipične funkcionalnosti koje su temelj zrelim OLTP (engl. *Online Transaction Processing*) odnosno transakcijskim sustavima. Tako AllegroGraph korisnicima nudi cjeloviti ACID pristup upravljanju transakcijama, mogućnost repliciranja podataka i slično. Tehnologije nižih slojeva poput podrške zaključivanju također omogućuju da se sustav koristi u svrhe prediktivne analitike [30].

5.2. Pohrana podataka u AllegroGraph SUBP

Krenimo s prikazom podataka, i njima pripadne strukture, za čiju je pohranu AllegroGraph sustav dizajniran.



Slika 10. Prikaz podataka koji se spremaju u AllegroGraph bazu podataka (Izvor: autorova izrada)

Vidljivo je kako su podaci dani na Slici 10 povezani u strukturu grafa te da je na temelju istih moguće iščitati eksplicitne veze između entiteta poput *Rex je Johnnyjev ljubimac.* Isto tako, u danome grafu mogu se prepoznati i određene implicitne veze koje proizlaze iz podataka u grafu, kao primjerice činjenica *John ima Rexa za ljubimca.* Iako postoje mnogi načini

spremanja prikazanih podataka, AllegroGraph sustav, kao što je već više puta rečeno, bazira se na RDF modelu podataka koji te podatke razlaže na skup tvrdnji, odnosno trojke te omogućuje izvođenje novih informacija iz postojećih.

Ukoliko zanemarimo RDF na trenutak i sagledamo AllegroGraph kao SUBP izvan konteksta semantičkog weba i pripadnih tehnologija koje čine okosnicu sustava te se umjesto toga fokusiramo na samu fizičku pohranu podataka u AllegroGraph bazu, onda istu možemo okarakterizirati kao spremnik petorki. Naime, većina RDF baza podataka, ako ne i sve, podatke ne sprema kao tvrdnje u formatu trojki sastavljenih od subjekta, predikata i objekta već u formatima koji se sastoje od više od tri polja, a koji trojkama pridružuju dodatni kontekst i pojednostavljuju manipuliranje istima. Tvrdnje se stoga danas, kao osnovne jedinice pohrane u RDF bazu podataka, nazivaju trojkama samo zbog povezanosti s RDF modelom podataka čije se karakteristike, poput mogućnosti zaključivanja nad podacima, nastoje iskoristiti.

Tako je svaka tvrdnja u AllegroGraph bazi podataka zapravo petorka sastavljena od standardnih dijelova RDF trojke, kao i od dva dodatna polja [32]. Redom ta polja čine:

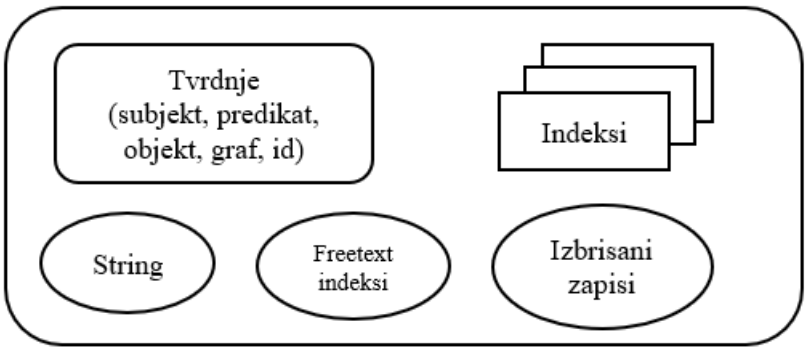
- subjekt (s)
- predikat (p)
- objekt (o)
- graf (g)
- identifikator trojke (i)

Identifikator trojke, kao što sam naziv sugerira, omogućuje jedinstveno identificiranje tvrdnje što olakšava referenciranje na istu. Navedeno omogućava jednostavno uvođenje takozvane reifikacije u AllegroGraph bazu podataka, što je još jedno od svojstava RDF-a. Reifikacija u principu omogućuje davanje izjava o izjavama to jest, omogućuje korištenje RDF trojke kao jednog od dijela neke druge RDF trojke. Recimo da u AllegroGraph bazu podataka želimo pohraniti sljedeću tvrdnju: *John vjeruje da je Zemlja ravna*, jedan od načina je upravo korištenjem reifikacije. Prvoj tvrdnji *Zemlja je ravna* biti će pridružen jedinstveni identifikator *i* koji će se nakon toga koristiti za definiranje početne tvrdnje u obliku *John vjeruje i.* RDF sintaksa definira ključnu riječ za korištenje svojstva reifikacije, *Statement.*

Kako se najveće prednosti RDF baza podataka postižu upravo agregiranjem znanja u jedan RDF graf, korištenje iste u granicama domene postaje nepraktično zbog same veličine baze. Iz tog razloga AllegroGraph tvrdnjama pridružuje oznaku grafa koja se odnosi na kontekst tvrdnje [32].

5.2.1. Logička struktura AllegroGraph baze podataka

Tvrdnje, odnosno petorke, podatkovne su jedinice koje se spremaju u bazu podataka, no predstavljaju samo jedan od elemenata, specifičnih AllegroGraph sustavu i prikazanih na Slici 11, koji se koristi za njihovo efikasno spremanje.



Slika 11. Logička struktura AllegroGraph baze podataka (Prema: [32])

- Sva prije definirana polja, izuzev identifikatora, nizovi su znakova (nadalje stringovi) varirajuće veličine. Kako bi bilo neefikasno spremati duplikate tih zapisa kod svakog unosa tvrdnje u bazu podataka, svakome se jedinstvenom stringu kod unosa pridjeljuje poseban broj UPI (engl. *Unique Part Identifier*). Rječnik stringova zadužen je za upravljanje UPI-jima te prevenciju duplikata [32].
- Kako bi se ubrzali upiti, AllegroGraph kreira indekse koji se uglavnom sastoje od tvrdnje, odnosno petorke, te dodatnih informacija. Također postoje i takozvani freetext indeksi koji omogućuju pretraživanje baze podataka prema stringovima pridruženima dijelovima tvrdnji [32].
- Naposljetku, logičku strukturu AllegroGraph spremnika trojki čini i modul koji prati brisanje tvrdnji u svrhu oporavka i reindeksiranja. Taj modul neophodan je za rješavanje česte pojave duplih tvrdnji u AllegroGraph bazi podataka koja može višestruko smanjiti efikasnost same baze. Problem je moguće riješiti i ugrađenim funkcijama unutar AllegroGraph sustava koje korisniku dopuštaju da definira kada su dvije tvrdnje iste [32] : kada imaju isti subjekt, objekt ili nešto treće.

Navedeni i ukratko opisani elementi koji čine logičku strukturu AllegroGraph baze podataka i nalaze se u pozadini svake pohrane podataka u istu, neovisni su o načinu na koji se baza podataka koristi. Koristila se ona kao RDF ili kao grafovska baza podataka, podaci će se uvijek pohranjivati u obliku petorki sastavljenih od prije navedenih polja. No, ovisno o načinu korištenja, AllegroGraph u kontekstu pohrane podataka dolazi s određenim posebitostima.

5.2.2. Pohrana podataka u AllegroGraph RDF bazu podataka

Kad korisnik koristi RDF bazu podataka, bilo na način da unosi podatke, postavlja upite nad njome ili nešto treće, svjestan je kako u pozadini zapravo upravlja RDF modelom podataka odnosno RDF grafom. Razlozi pohrane podataka upravo u formatu trojki sastavljenih od subjekta, predikata i objekta razni su, no u principu se nastoje iskoristiti mogućnosti koje proizlaze kako iz samog RDF modela podataka tako i iz tehnologija koje se nadovezuju na njega.

AllegroGraph kao RDF baza podataka naglasak stavlja na prva tri polja petorki (subjekt, predikat i objekt). Kako su navedena polja zapravo nizovi znakova varijabilne veličine, ne postoje ograničenja u unosu vrijednosti u iste, što doprinosi fleksibilnosti sheme, ali u isto vrijeme ograničava korištenje RDF baze podataka na jednostavan spremnik međusobno povezanih činjenica. Moguće je da je to u određenim slučajevima jedini zahtjev prema RDF bazi podataka, no puno je izglednije kako odabir RDF baze podataka kao tehnologije za pohranu podataka proizlazi iz zahtjeva domene, kao primjerice nužnost izvođenja informacija iz skupa podataka. Kako se zaključivanje nad RDF grafom provodi na temelju pravila danih u jezicima za modeliranje ontologija kao sheme podataka RDF baza, vrlo je važno biti svjestan na koji način, i da li u cijelosti, RDF baza podataka podržava konstrukte tih jezika kod provođenja zaključivanja.

5.2.2.1. Podrška zaključivanju

Definiranje sheme, ontologija, korištenjem RDFS i OWL sintakse u potpunosti je podržano od strane AllegroGraph sustava, no zaključivanje korištenjem RDFS i OWL predikata svedeno je na podskup njihove sveukupne sintakse. Sljedeći RDFS i OWL predikati mogu se koristiti u svrhu izvođenja RDF trojki iz postojećeg RDF grafa u AllegroGraph bazi podataka korištenjem jednog od dostupnih alata za zaključivanje, RDFS++-a:

- *rdfs:domain, rdfs:range, rdfs:subClassOf, rdfs:subPropertyOf, owl:inverseOf, owl:sameAs, owl:SymmetricProperty, owl:TransitiveProperty* [30],
- *owl:hasValue, owl:someValuesFrom i owl:allValuesFrom* korištenjem dodatnog modula koji se mora uključiti u RDFS++ alat za zaključivanje [30],
- dok je jedini predikat iz čistog RDF-a koji se koristi za zaključivanje *rdf:type* [30].

Navedena činjenica ima vrlo važan utjecaj na očekivane rezultate zaključivanja nad AllegroGraph bazom podataka. Primjerice, kako AllegroGraph alat za zaključivanje RDFS++

ne podržava OWL predikat *owl:minCardinality* on neće automatski klasificirati individue u ovisnosti o korištenju tog predikata.

RDFS++ alat za zaključivanje specifičan je pristup zaključivanju nad RDF podacima u AllegroGraph bazi podataka. Za razliku od najčešće korištenih OWL-DL alata za zaključivanje koji se temelje na deskriptivnoj logici i podržavaju predikate iz OWL 1 profila istog naziva, RDFS++ alat za zaključivanje nije kompletan, odnosno ne daje sve moguće odgovore na postavljeni upit kako bi ostvario karakteristiku predvidivog vremena izvršavanja [30]. RDFS++ alat za zaključivanje također nema eksplicitnu fazu materijalizacije. Materijalizacija je proces pohrane zaključenih odnosno izvedenih RDF trojki u bazu podataka kako bi se ubrzali budući upiti [30]. Centralni problem zbog kojeg AllegroGraph sustav zazire od materijalizacije izvedenih RDF trojki je održavanje baze podataka. Naime, promjene ontologije u RDF bazi podataka najčešće za posljedicu imaju promjenu izvedenih RDF trojki što pak zahtijeva ponovnu provedbu zaključivanja. U situacijama kada se RDF baza koristi kao transakcijski sustav i sadrži milijune trojki, navedeno jednostavno nije izvedivo.

No, kako se AllegroGraph koristi i u statičnijim okružjima u kojima su promjene strukture RDF grafa minimalne i rijetke, AllegroGraph s ciljem efikasnog zaključivanja u svrhu donošenja poslovnih odluka nudi i OWL2 RL alat za zaključivanje. OWL 2 RL alat za zaključivanje bazira se na OWL2 RL profilu koji je prilagođen korištenju od strane alata za zaključivanje na temelju pravila te uključuje fazu materijalizacije [30]. Navedeni alat kroz određeni broj setova pravila proširuje skup OWL predikata koji se uzimaju u obzir kod provedbe zaključivanja.

Iako nije specifično usmjeren na zaključivanje nad RDF podacima u AllegroGraph RDF bazi podataka, već kao jedan od načina postavljanja upita nad istom, AllegroGraphov RDF Prolog koristi se kao alat za zaključivanje u specifičnim domenama koje je teško modelirati sa RDF/RDFS-om ili OWL-om. Također se često koristi kao alat za zaključivanje na temelju pravila direktno nadovezan na RDFS++ [30].

Odabir alata za zaključivanje ovisi o domeni u kojoj se AllegroGraph RDF baza podataka koristi te specifičnim podacima koji se žele izvući iz baze. Kratak osvrt na mogućnosti zaključivanja u AllegroGraph sustavu dan je u Tablici 4.

Tablica 4. Načini provođenja zaključivanja nad AllegroGraph bazom podataka

Načini provođenja zaključivanja nad AllegroGraph bazom podataka	
RDFS++ alat za zaključivanje	Koristi podskup RDFS i OWL rječnika za izvođenje trojki iz baze podataka. Ne materijalizira izvedene trojke.

OWL2 RL alat za zaključivanje	Izvodi trojke iz baze podataka te ih materijalizira koristeći skupove pravila temeljenih na OWL rječniku.
Prolog	Koristi se za izvođenje trojki iz baze podataka u slučajevima kada se OWL i RDFS ne koriste kao shema baze ili ne omogućuju provedbu željenih operacija nad podacima.

5.2.2.2. Unos podataka

Podaci se u AllegroGraph bazu podataka mogu unositi programski ili u obliku datoteka u nekom od formata podržanih od strane AllegroGraph sustava. Programski unos podataka služi za inkrementalnu izgradnju RDF baze podataka, što se postiže SPARQL INSERT naredbom ili pak kroz HTTP POST metodu direktno ili u pozadini određene biblioteke programskog jezika [32]. Prema AllegroGraph dokumentaciji [33], RDF podaci generalno u AllegroGraph bazu podataka dolaze kao stringovi u nekom od formata za serijalizaciju RDF-a. Sljedeći formati za serijalizaciju RDF-a mogu se koristiti za unos RDF trojki ili pak cijelih RDF grafova u AllegroGraph bazu podataka:

- JSON-LD, N-Quads, N-Triples, Extended N-Quads, RDF/XML, TriG, TriX i Turtle [33]. Naravno, kako se AllegroGraph može koristiti i kao grafovska baza opće namjene, o čemu će biti više riječi u nastavku rada, podaci se u većim količinama mogu unositi u bazu i u nekim formatima osim RDF-a. Podaci u JSON, JSONLines i CSV formatu prije unosa u bazu podataka transformiraju se u RDF trojke prema pravilima interno definiranim u implementaciji sustava [33].

5.2.2.3. Očuvanje konzistentnosti baze podataka

Kako se AllegroGraph koristi kao OLTP sustav s potpunom podrškom ACID svojstava transakcija, validacija podataka koji se unose u bazu podataka kroz provjere tipa:

- koliko je trojki sa specificiranim subjektom i predikatom dozvoljeno pohraniti u bazu ili
- kojeg su tipa vrijednosti objekta povezanog s određenim predikatom i subjektom

neophodna je.

Jedan od prvotnih pristupa AllegroGrapha validaciji RDF-a bio je SPIN (engl. *SPARQL Inferencing Notation*). SPIN je omogućio da se pravila za validaciju definiraju kroz SPARQL

te, kao što je to slučaj kod SHACL-a, enkodiraju direktno u RDF graf kao trojke [34]. No, kako nikad nije postao jedan od standarda W3C skupa tehnologija semantičkog weba, zamijenjen je SHACL-om.

AllegroGraphov SHACL API svodi se na tri koraka prema [35].

1. određivanje skupa podataka, to jest RDF grafa, koji je potrebno validirati
2. izgradnja SHACL trojki koje predstavljaju željeno stanje podataka u RDF grafu
3. provođenje SHACL validacije

SHACL validacijom ne modificiraju se podaci nad kojima se ista provodi. Jednom kada se dobiju rezultati validacije, na korisniku je da na temelju utvrđenih nekonzistentnosti ažuriraju stanje podataka i ponovno provedu validaciju [35].

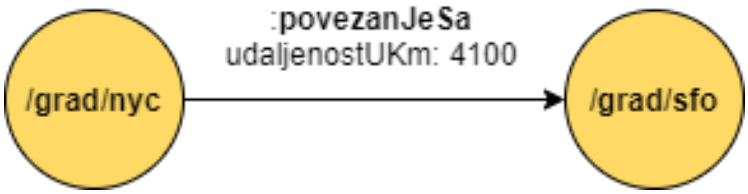
5.2.3. AllegroGraph sustav kao grafovska baza podataka

Kao što je već rečeno, svaka baza podataka koja se temelji na grafu kao modelu podataka može se nazvati grafovskom bazom podataka. Vodeći se tom tvrdnjom AllegroGraph je nenativna grafovska baza podataka, jer implementira RDF model podataka koji je jedan od tipova graf modela podataka. Iako je do sada o AllegroGraph bazi podataka govoreno kao o RDF bazi koja implementira razne tehnologije semantičkog weba koje se prirodno nadovezuju na RDF, apsolutno ništa ga striktno ne veže za RDF svijet. Ukoliko ne postoji potreba za korištenjem funkcionalnosti koje dolaze u paketu s RDF-om, poput recimo zaključivanja, AllegroGraph se može koristiti kao grafovska baza opće namjene. U tom slučaju u AllegroGraph bazu podataka možemo unositi izjave u kojima se subjekt i objekt RDF trojke tretiraju kao čvorovi koji su povezani bridom, predikatom, s dodatnim informacijama u polju grafa [32].

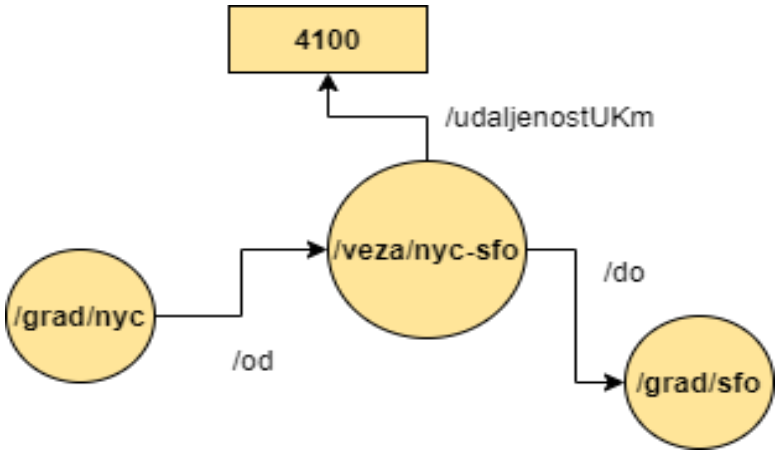
Naravno, unatoč vrlo širokoj definiciji grafovskih baza podataka, one se kao tehnologija za pohranu podataka rijetko odabiru samo kako bi se podaci koji prirodno tvore strukturu grafa na taj način spremili. Vrlo važno je da grafovska baza podataka između ostalog omogući korisnicima manipulaciju grafom kroz tipične operacije kao što su obilazak grafa, pretraživanje susjedstva, traženje klastera i slično. AllegroGraph se na više načina prilagođava navedenim zahtjevima te nastoji bazni RDF model podataka približiti grafu sa svojstvima s ciljem smanjenja granularnosti podataka koji se spremaju kao trojke.

5.2.3.1. Atributi trojki

Atributi trojki predstavljaju parove ključ/vrijednost koji se povezuju s trojkama u AllegroGraph bazi podataka. Moguće ih je pridružiti trojkama samo kod unosa istih u bazu podataka te ih je nemoguće kasnije mijenjati [36]. Iako se u AllegroGraph dokumentaciji [36] kao jedan od ciljanih slučajeva korištenja atributa trojki navodi kontrola pristupa trojkama, što je naravno vrlo važna karakteristika koja proizlazi iz korištenja AllegroGrapha kao transakcijske baze podataka, postoje indikacije [16] kako je pravi razlog pridruživanja atributa trojkama zapravo udaljavanje od RDF modela podataka prema grafu sa svojstvima koji puno efikasnije pohranjuje kako svojstva čvorova tako i svojstva bridova koji ih povezuju. Recimo da u graf bazu podataka želimo spremiti sljedeću informaciju: *Udaljenost između New Yorka i San Francisca je 4100 kilometara.* Ukoliko koristimo grafovsku bazu podataka koja se temelji na grafu sa svojstvima kao modelu podataka, *New York* i *San Francisco* predstavljat će čvorove koji će biti povezani bridom koji predstavlja konekciju između danih čvorova s pridruženim svojstvom udaljenosti, kao na Slici 12. Kako se predikatu RDF trojke ne može pridružiti svojstvo, problem se rješava uvođenjem dodatnog čvora između subjekta i objekta, kao i dodatnog čvora koji će pohraniti vrijednost svojstva udaljenosti. AllegroGraph će korištenjem atributa trojki u bazu spremiti samo jednu RDF trojku dok će se svojstvo udaljenosti spremiti kao atribut pridružen toj trojki, Slika 13. Naravno, uporaba atributa trojki nije nametnuta.



Slika 12: Primjer grafa sa svojstvima (Prema: https://dist.neo4j.com/wp-content/uploads/20170617200105/RDF-Modeling-Workaround.png)



Slika 13. Primjer RDF grafa (Prema: https://dist.neo4j.com/wp-content/uploads/20170617200105/RDF-Modeling-Workaround.png)

Kako bi se atributi trojkama pridruživali što efikasnije i jednostavnije, dizajniran je i novi format za serijalizaciju RDF-a baziran na N-Quad formatu pod nazivom Extended N-Quad format, kraće NQX [33]. Svaka linija u NQX datoteci, nakon standardne N-Quad linije, može biti nastavljena željenim brojem atributa u JSON formatu. Sljedeći primjer RDF trojke s pridruženim atributom u NQX formatu odnosi se na informaciju o udaljenosti između gradova o kojem je bilo riječi ranije u radu:

- *<grad:nyc><povezanJeSa><grad:sfo> {"udaljenostUKm": "4100"}.*

Vrlo važno je spomenuti kako se atributi pridruženi trojkama u NQX datoteci ili korišteni kod programskog unosa trojki u bazu prije korištenja moraju definirati u samome sustavu. Prednost tog, naizgled dodatnog koraka, omogućuje da se set atributa unaprijed definira i automatski dodijeli svakoj trojci neovisno u kojem se formatu unosi u bazu podataka [33].

5.2.3.2. JIG

Slika 14. JIG logo (Izvor: https://raw.githubusercontent.com/wiki/joshsh/jig/graphics/jig-logo-medium.png)

JIG (engl. *JavaScript-based interface for graph traversal*) programski je jezik po uzoru na Gremlin ili Ripple, specijalno razvijen za potrebe manipulacije grafom u AllegroGraph sustavu korištenjem JavaScript sintakse [36]. Potekao je iz razvoja algoritama za poseban modul AllegroGraph sustava, odnosno za potrebe analize društvenih mreža, no može ga se koristiti i izvan tog konteksta za provođenje raznih operacija nad grafom.

JIG je jedan od načina na koji se pristupa podacima u AllegroGraph bazi podataka, no nikako nije najkorišteniji način zbog njegove specijaliziranosti za slučajeve korištenja koji nadilaze tipične potrebe korisnika. Postavljanje upita nad AllegroGraph bazom podataka stoga je uglavnom ostvareno kroz standardizirani SPARQL upitni jezik.

5.2.4. AllegroGraph kao dokument baza podataka

JSON-LD do sada je bio spomenut i ukratko opisan kako jedan od podržanih formata za serijalizaciju RDF-a, no on je zapravo puno više od toga. Danas je jedna od najpopularnijih i najprihvaćenijih tehnologija proizašlih iz područja semantičkog weba. Uloga su mu svakojake, od formata za serijalizaciju do alata za pridavanje značenja podacima na webu odnosno web stranicama, no temeljna ideja za koju je prvotno osmišljen kreiranje je mreže podataka koji počivaju na standardima i razumljivi su računalima [37]. Zbog jednostavne sintakse preuzete iz JSON formata i potpore od strane giganata na web sceni, poput Googlea, JSON-LD sveprisutan je, te pretvorbom nestrukturiranih u strukturirane podatke čini web boljim mjestom.

Pohrana JSON-LD dokumenata trivijalna je, no isto tako stvar vrijedna diskusije. Kako proizlazi iz područja semantičkog weba te služi kao format za serijalizaciju RDF-a, razlaganje JSON-LD dokumenata na RDF trojke te njihovo spremanje u RDF baze podataka nameće se kao efikasno i logično rješenje. Ali, isto tako, kako se radi o dokumentima koji izlažu JSON

sintaksu, za njihovo efikasno spremanje moguće je koristiti i prije spomenute dokument baze podataka. AllegroGraph na neki način objedinjuje navedene pristupe te se zbog toga u svojoj multi-model definiciji proširuje i na dokument baze podataka.

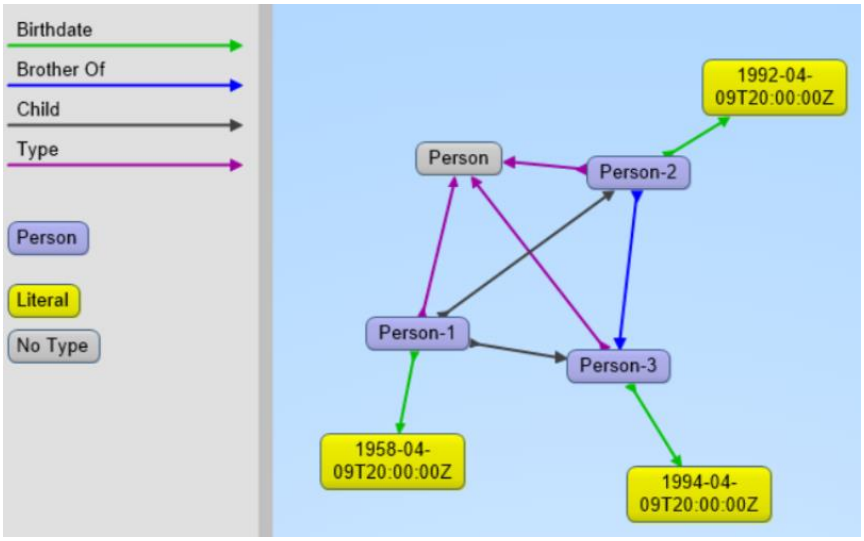
Prema [38] AllegroGraph sustav poželjno je i preporučljivo koristiti kao dokument bazu za spremanje podataka danih u JSON-LD formatu u slučajevima kada se planira kombinirati JSON-LD, ili sam JSON, dokumente sa grafovima te nad njima provoditi tipične operacije definirane nad modelom grafa. U dokument bazama podataka dokumenti se pohranjuju na način optimiziran za njihovo direktno dohvaćanje. Složeni upiti koji uključuju spajanje dokumenata prema ključevima nisu toliko efikasni, dok su operacije poput pronalaženja najkraćih puteva, što je u slučaju JSON-LD dokumenata u čijoj su srži upravo veze između podataka, skoro pa neizvedive [38].

Tako se primjerice, u AllegroGraph bazi podataka sljedeći JSON-LD dokument koji opisuje tri JSON objekta takozvanom *foaf* ontologijom, ontologijom koja opisuje ljude i odnose između njih, razlaže na RDF trojke povezane u RDF graf dan na Slici 15. Važno je napomenuti kako AllegroGraph omogućuje spremanje i cijelih JSON-LD dokumenata, no u tom slučaju oni su pohranjeni kao stringovi, čime se pak onemogućuje bilo kakvo složeno manipuliranje podacima pohranjenima u iste [38]. Velika prednost AllegroGrapha kao dokument baze podataka mogućnost je korištenja SPARQL upitnog jezika za postavljanje upita nad JSON-LD dokumentima.

```
p1 = {
    "@context": context,
    "@type":"foaf:Person",
    "@id":"foaf:person-1",
    "foaf:birthdate": "1958-04-09T20:00:00Z",
    "foaf:child": ['foaf:person-2', 'foaf:person-3']
}

p2 = {
    "@context": context,
    "@type":"foaf:Person",
    "@id":"foaf:person-2",
    "foaf:brotherOf": "foaf:person-3",
    "foaf:birthdate": "1992-04-09T20:00:00Z",
}

p3 = {"@context": context,
    "@type":"foaf:Person",
    "@id":"foaf:person-3",
    "foaf:birthdate": "1994-04-09T20:00:00Z",
}
```



Slika 15. Prikaz pohrane JSON-LD dokumenata u AllegroGraph bazi podataka (Izvor: https://franz.com/agraph/support/documentation/current/python/_images/person-graph.png)

5.3. Pristup podacima u AllegroGraph SUBP-u

Dohvaćanje podataka iz AllegroGraph sustava omogućeno je kroz različite načine. Najjednostavniji od njih, a ujedno i baza svakome od složenijih, svodi se na traženje trojki koje se preklapaju s danim uzorkom sastavljenim od subjekta, predikata, objekta i oznake grafa [32]. Svaki dio uzorka može se zadati kao točno preklapanje, raspon vrijednosti ili takozvani *wild card* (vrijednost nije važna za pretraživanje). Primjerice, temeljem uzorka

```
subject   : <http://www.example.com/people/gwking>
predicate: wild
object    : wild
graph     : <http://www.example.com/context/initial>
```

iz AllegroGraph baze podataka dohvatile bi se sve trojke o subjektu *gwking* iz grafa s imenom *initial.* Pretraživanje trojki na temelju raspona vrijednosti nekog od njihovih dijelova omogućeno je pohranom vrijednosti u određenom tipu podataka, na taj se način zaobilazi prije opisani rječnik stringova i ubrzava pretraživanje [32]. Pretraživanje baze ubrzava se i korištenjem indeksa, no o tome će biti riječi kasnije u radu.

O W3C standardnome upitnom jeziku za postavljanje upita nad RDF bazom podataka već je bilo riječi u jednom od prethodnih poglavlja te on kao takav predstavlja primaran način interakcije s AllegroGraph bazom podataka. AllegroGraph sustav od verzije 4.4 podržava sve mogućnosti unutar SPARQL 1.1 specifikacije uključujući sintaksu za dodavanje, ažuriranje i brisanje trojki.

Rad s AllegroGraph bazom podataka koristeći SPARQL upitni jezik biti će detaljno prikazan kroz primjere u praktičnome dijelu rada.

Kako je SPARQL standardni način postavljanja upita nad AllegroGraph bazom podataka, a Prolog uveden kao alternativni mehanizam, postavlja se pitanje u čemu se oni razlikuju po pitanju pretraživanja baze podataka. Prema [36] najvažnije razlike su sljedeće:

- Prolog nije upitni, već programski jezik opće namjene što korisnicima omogućuje pisanje pravila i definiranje izraza koji nisu zapisani u samim tvrdnjama (trojkama).
- SPARQL je s druge strane potpuno kompatibilan s RDF-om pa prema tome "zna" više o tipovima podataka te podržava potpuni set operatora i funkcija kojima se izvode operacije nad trojkama. U Prologu je te operacije moguće izvoditi u suradnji s Lisp programskim jezikom.
- Što se pak tiče same efikasnosti pretraživanja, kako Prolog ne podržava pretraživanje u širinu (engl. *breadth-first-search*), već samo pretraživanje u dubinu (engl. *depth-first-serach*), u slučajevima kada se pretraživanje u širinu nameće kao bolja opcija, Prolog gubi na efikasnosti.

Tablica 5 sumira opisane način postavljanja upita nad AllegroGraph bazom podataka

Tablica 5. Načini postavljanja upita nad podacima u AllegroGraph bazi podataka

Načini postavljanja upita nad podacima u AllegroGraph bazi podataka	
Preklapanje uzoraka	Najjednostavniji način dohvaćanja podataka iz AllegroGraph baze podataka. Služi za dohvaćanje trojki prema njihovim dijelovima.
SPARQL upitni jezik	Standardni način rada s podacima u AllegroGraph sustavu.
Prolog	Alternativni način rada s podacima u AllegroGraph sustavu u slučajevima kada upiti nisu posebice vezani uz samu shemu baze.

5.4. Indeksiranje AllegroGraph baze podataka

RDF baze podataka bazirane su na indeksima, što je vrlo čest argument u korist nativnih grafovskih baza podataka kao dominantnog pristupa pohrani podataka kao grafa. Pristup indeksiranju RDF baza specifičan je i temelji se na ideji multi-indeksiranja koja se svodi na optimiziranje indeksa na način da pokriju najčešće sheme upita koji se postavljaju nad bazom podataka. U AllegroGraph sustavu svaki indeks permutacija je oznaka *s, p, o* te *g* koje su skraćenice za prije opisana polja petorki. Kako je vrijednost identifikatora trojki *i* jedinstvena, on ne ulazi u permutacije te su stoga za indeksiranje AllegroGraph baze podataka dostupna

- dvadeset i četiri indeksa koja započinju sa oznakama *s, p, o, g* u bilo kojem redoslijedu i završavaju s identifikatorom trojke *i* te
- indeks nad samim identifikatorom trojke *i* [34]

Tako primjerice koristeći *spogi* indeks

1. trojke se prvo sortiraju prema subjektu (*s*), a sve one sa istom vrijednošću grupiraju se zajedno;
2. unutar grupa, trojke se sortiraju prema predikatu (*p*);
3. prateći isti princip, trojke se nadalje sortiraju prema objektu (*o*), nakon toga prema oznaci grafa (*g*) te naposljetku prema identifikatoru trojke (*i*)

Indeksiranje je omogućeno na bilo koji od ukupno dvadeset i pet dostupnih načina, no unatoč tome AllegroGraph za brzo pretraživanje baze podataka u pozadini, kod dodavanja trojki, automatski kreira sedam indeksa. Taj zadani set indeksa čine *spogi, posgi, ospgi, gspoi, gposi, gospi* i *i* indeks*.* Na korisniku je da na temelju zahtjeva svoje aplikacije eliminira indekse koje neće koristiti ili pak kreira indekse koji najbolje odgovaraju njegovim uzorcima upita. Primjerice, ako korisnik unutar svoje aplikacije ne planira koristiti pod-grafove, indeksi koji započinju sa *g,* oznakom grafa, koristit će se samo kao zadnja opcija zbog nepostojanja prikladnijih indeksa [32]. Indeksiranje baze podataka eliminacijom tih indeksa može se višestruko ubrzati, isto kao i pretraživanje tvrdnji.

U ovome poglavlju predstavljen je AllegroGraph SUBP na način da se većina njegovih karakteristika, bilo u kontekstu RDF, grafovske ili dokument baze podataka, povezala s teorijom u prethodnim poglavljima. Između ostalog, predstavljena je arhitektura sustava, podržanost određenih tehnologija te načini pohrane i pristupa podacima. Kako bi se do sad obuhvaćena opća i specifična teorija o samome sustavu zaokružila njihovom praktičnom upotrebom, u nastavku rada fokus će biti na razvoju baze podataka u AllegroGraph sustavu.

No, prije prelaska na praktični dio rada, valja ukratko sumirati povezanost AllegroGraph sustava sa raznim semantičkim tehnologijama, u Tablici 6, koje su detaljnije objašnjene u prethodnim poglavljima.

Tablica 6. Povezanost AllegroGraph sustava sa semantičkim tehnologijama

SEMANTIČKA TEHNOLOGIJA	DOMENA KORIŠTENJA
RDF	Grafovski model podataka AllegroGraph sustava.
RDFS	Jezik za modeliranje sheme baze podataka koji uvodi osnovne konstrukte za klasificiranje entiteta domene koji se pak koriste za zaključivanje nad podacima u bazi.
OWL	Jezik koji nadopunjuje RDFS te omogućuje razvoj ontologija kao shema RDF baza podataka. Omogućuje korištenje OWL alata za zaključivanje nad AllegroGraph bazom podataka.
SHACL	Tehnologija za provjeru konzistentnosti AllegroGraph baze podataka.
SPARQL	Upitni jezik za postavljanje upita te rad sa podacima u AllegroGraph bazi podataka.
JSON-LD	Jedan od formata za serijalizaciju RDF modela podataka, no AllegroGraph ga koristi i kao dokument model podataka.

U sljedećem poglavlju sve tehnologije dane u prethodnoj tablici biti će obuhvaćene primjerima te će se koristiti za izradu baze podataka te za rad s istima u AllegorGraph sustavu.

6. Modeliranje podataka u AllegroGraph-u

Poglavlje koje slijedi fokusirat će se na praktičnoj primjeni prije opisanih mogućnosti AllegroGraph sustava. Na temelju kreirane baze podataka, čiji je razvoj iz višestrukih razloga specifičan, primjerima će biti dočarane i pobliže objašnjene opće funkcionalnosti AllegroGraph sustava, poput unošenja podataka i postavljanja upita nad istima, kao i one specifičnosti koje proizlaze iz veze sustava s RDF modelom podataka, odnosno semantičkim webom.

6.1. Modeliranje podataka

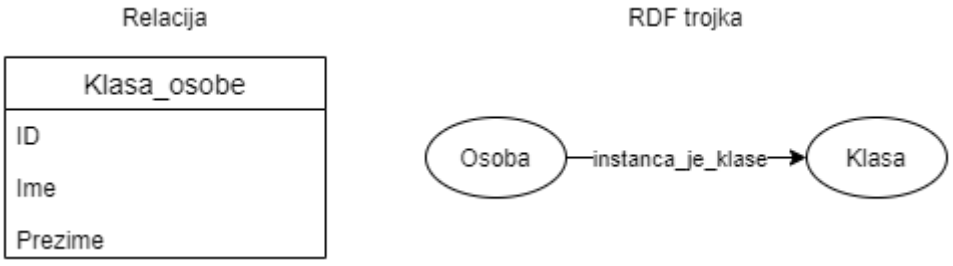
Kao što je do ove točke u radu više puta navedeno, jedna od karakteristika svake vrste baza podataka obuhvaćene definicijom AllegroGrapha kao multi-model sustava mogućnost je rada s istom bez prethodnog definiranja njene sheme odnosno strukture. No, prava prednost svake od njih zapravo je u opcionalnosti izrade sheme, što povlači činjenicu kako je u određenim slučajevima, ovisno o domeni i specifičnim potrebama, moguće i preporučljivo prije samog rada s bazom detaljno razraditi njenu strukturu.

Modeliranje podataka proces je određivanja i prikupljanja podatkovnih objekata, utvrđivanja veza između njih, kao i njihove ovisnosti o određenim poslovnim pravilima. Osnovni cilj, neovisno o tome koji se od tri osnovna modela podataka, bio on konceptualni, logički ili fizički, želi razviti u procesu modeliranja podataka, stvaranje je uvida u podatke koji se u nekoj domeni pojavljuju. Važno je odmah napomenuti kako u ovome kontekstu model podataka ne podrazumijeva implementacijski model podataka na kojem se temelje baze podataka, već konkretni opis podataka koji se nastoji pohraniti u bazu. Odabir modela podataka kao rezultata procesa modeliranja pretežno ovisi razini apstrakcije kojom se podaci nastoje opisati, pa se tako fizički model podataka, najniže razine apstrakcije, koristi upravo kao temelj za izradu baze podataka u konkretnome sustavu te ponekad sinonimno naziva shemom baze podataka.

AllegroGraph, sustav je koji objedinjuje čak tri (implementacijska) modela podataka na kojima se baziraju RDF, grafovske te dokument baze podataka, što u samome početku modeliranja podataka u istome predstavlja problem. Naime, odabir jedne od dostupnih baza podataka određuje način rada s podacima, kao i mogućnosti sustava koje će biti moguće prikazati u nastavku rada. Kako je sam AllegroGraph sustav prvotno osmišljen kao RDF baza podataka, a većina njegovih mogućnosti temelji se na RDF modelu podataka i na njega nadovezujućim tehnologijama, odlučeno je kako će se za potrebe praktičnog dijela rada razviti RDF baza podataka te da će shema, zbog istih potreba, biti preddefinirana.

6.2. Modeliranje ontologije

Definiranje sheme u RDF bazi podataka zapravo podrazumijeva razvijanje ontologije koja će na adekvatan način opisati željenu domenu. Sam proces razvoja, za razliku od razvoja sheme za relacijske baze podataka, nije standardiziran, no u većini dostupnih metoda svodi se na prepoznavanje i određivanje entiteta domene, veza između njih te njihovih svojstava, što je zapravo vrlo slično modeliranju relacijske sheme. Vrlo važna razlika između shema relacijske i RDF baze podataka je u tome što je shema RDF baze podataka, odnosno ontologija, implementirana u same podatke, odnosno nije definirana na višoj razini kao što je slučaj s relacijama u relacijskoj bazi podataka. Shema dana u samim podacima RDF bazi podataka omogućuje vrlo fleksibilne promjene, kao i postavljanje upita nad istom. Primjerice, uzmimo slučaj u kojem želimo saznati klasu to jest tip određene osobe pohranjene u bazi podataka. U relacijskoj bazi podataka ta informacije rijetko je predstavljena u obliku atributa u tablici, već je najčešće dana u definiciji same relacije. Kako bi došli do informacije o klasi osobe potrebno je poznavati samu shemu baze podataka. U RDF bazi podataka shema je dana u podacima, a upit o klasi osobe moguće je postaviti nad samom shemom.



Slika 16. Razlika u implementaciji sheme u relacijskim i RDF bazama podataka (Izvor: autorova izrada)

Na Slici 16, RDF trojka koja opisuje osobu kao instancu određene klase dana je vrlo apstraktno. Definiranjem vlastitih predikata, iako u svakome smislu moguće, u nekim slučajevima kao što je ovaj ograničava rad sa samom RDF bazom podataka zbog čega se u svrhu modeliranja njene sheme koriste za to standardizirani jezici RDFS i OWL. Između ostalog omogućuju uniformno sučelje za postavljanje upita, zaključivanje nad podacima te spajanje podataka iz različitih RDF baza podataka neovisno o njihovoj shemi.

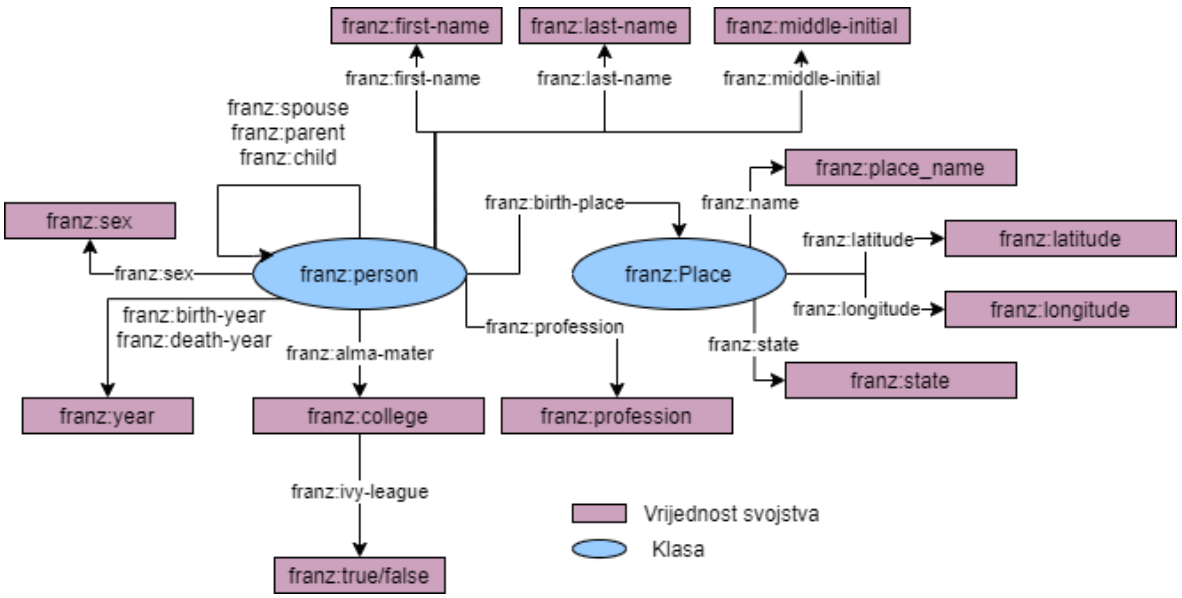
6.3. Opis domene i početne ontologije

Razvoj baze podataka nad kojom će se detaljno i jasno moći prikazati specifičnosti rada s podacima u AllegroGraph sustavu, kao što je to slučaj i s bilo kojom drugom bazom podataka, započinje definiranjem domene koju će ona obuhvatiti. Definiranjem domene određuju se podaci koji će se u bazu unositi, poslovna pravila i restrikcije koje podaci moraju zadovoljavati te operacije koje će se nad tom istom bazom moći provoditi.

Domena baze podataka koja će se implementirati u nastavku praktičnog dijela rada odnosi se na obiteljsko stablo bivšeg predsjednika SAD-a, Johna F. Kennedyja. Kako je obiteljsko stablo odličan primjer hijerarhije i više je nego dobra domena za razvoj ontologije kao sheme baze podataka i prikaz svih prije objašnjenih mogućnosti AllegroGraph sustava.

Kreirana baza podataka tako će se moći koristiti za obilaženje obiteljskog stabla, otkrivanje veza između pojedinih osoba bilo srodstvom ili pak nekim drugim čimbenikom poput pohađanog fakulteta. Baza će se također moći koristiti za izvlačenje informacija o osobama poput njihova zanimanja, a korištenjem semantičkome webu specifičnih mogućnosti poput zaključivanja nad pohranjenim podacima, moći će se izvoditi nove informacije koje se eksplicitno ne nalaze u samoj bazi.

Početna verzija ontologije, koja će se kasnije u radu doraditi, dostupna je je unutar samog AllegroGraph servera u prije spomenutom i prikazanom N-Triples formatu. Ontologija se sastoji od 1214 RDF trojki, a njena grafička reprezentacija dana je na Slici 17.



Slika 17. Početna verzija ontologije (Izvor: autorova izrada)

Ontologija obiteljskog stabla obitelji Kennedy sastoji se od dvije konkretne klase čije instance predstavljaju članove obitelji Kennedy (klasa *person*) te mjesta rođenja istih (klasa *Place*). Instancama klasa također je pridružen određen broj svojstava čije su vrijednosti predstavljene literalima tipa string (pr. vrijednosti svojstva *franz:birth-year*) ili pak resursima definiranim unutar *franz* područja imena (pr. vrijednosti svojstva *franz:alma-mater*) čije konkretne definicije nisu dane unutar N-Triples dokumenta te zbog toga nisu prikazane ni u ontologiji na prethodnoj slici.

Ovako definirana ontologija, iako validna i spremna za pohranu u bazu podataka, nije dostatna za prikaz svih mogućnosti AllegroGraph sustava te ju je potrebno nadopuniti konstruktima iz RDFS i OWL jezika. U prikazanoj početnoj verziji ontologije za sada se koristi samo jedno svojstvo koje nije korisnički definirano, *rdf:type,* kojim se resursima određuje pripadnost nekoj od spomenutih klasa. Primjer dijela definicije člana obitelji Kennedy u N-Triple formatu dan je u nastavku.

```
<http://www.franz.com/simple#person1>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.franz.com/simple#person> .

<http://www.franz.com/simple#person1>
<http://www.franz.com/simple#first-name>
"Joseph" .

<http://www.franz.com/simple#person1>
<http://www.franz.com/simple#has-child>
<http://www.franz.com/simple#person3> .
```

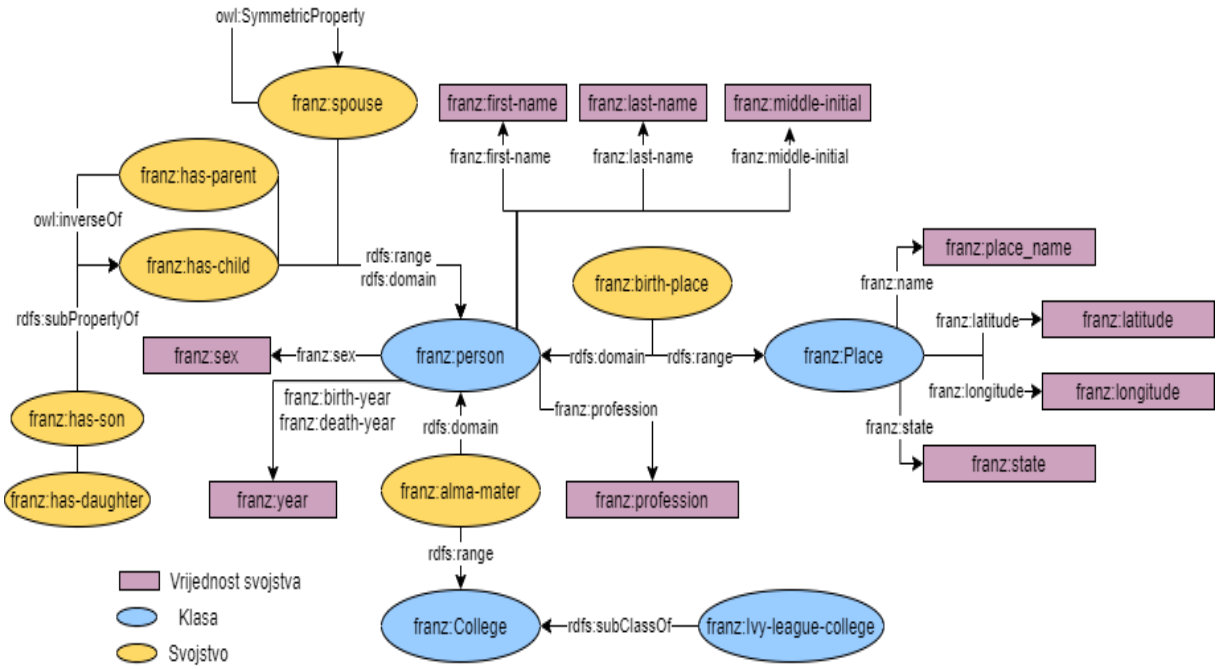
6.4. Prilagodba početne ontologije potrebama rada

Kao što je u prethodnom poglavlju navedeno, početna ontologija zbog svoje relativne jednostavnosti, ponajprije uslijed izostanka koncepata iz sintakse RDFS i OWL jezika, nije dostatna za detaljan prikaz mogućnosti manipuliranja podacima u AllegroGraph sustavu. Stoga će se ista, za potrebe ovog rada, nadopuniti i prilagoditi zahtjevima sustava imajući na umu određena ograničenja u vidu RDFS i OWL konstrukata koje je moguće koristiti kod provođenja zaključivanja nad podacima. Sljedeći koraci poduzeti su kako bi se ontologija prilagodila potrebama rada i pretvorila u shemu koja će se u nastavku koristiti za izradu baze podataka.

1. Definiranje klase *franz:College* koja će obuhvatiti fakultete koji su do sada bili definirani kao resursi bez tipa.

2. Definiranje klase *franz:Ivy-league-college* kao podklase *franz:College* klase korištenjem *rdfs:subClassOf* svojstva. Na taj način eliminirati će se potreba za korištenjem svojstva *franz:ivy-league* kao klasifikatora fakulteta.
3. Svojstvo *franz:alma-mater* dodatno će se ograničiti korištenjem svojstava *rdfs:range* i *rdfs:domain.* Tako će domena navedenog svojstva biti klasa *franz:person* dok će mu raspon biti klasa *franz:College.*
4. Domena i raspon također će se definirati i svojstvu *franz:birth-place* (domena: *franz:person,* raspon: *franz:Place*) te svojstvima *franz:spouse, franz:has-parent* i *franz:has-child* kojima će vrijednost svojstva domene i raspona biti klasa *franz:person.*
5. Svojstvo *franz:spouse,* koje definira brak između osoba u obiteljskom stablu, definirat će se kao simetrično korištenjem svojstva *owl:SymmetricProperty.* Time će se smanjiti broj RDF trojki jer se svojstvo *franz:spouse* neće morati definirati za obje osobe koje su u braku odnosno, veza u drugome smjeru izvest će se na temelju zaključivanja nad podacima u bazi podataka.
6. Svojstvo *franz:has-parent* definirat će se kao inverzno, *owl:inverseOf,* svojstvu *franz:has-child* čime će se informacija o roditeljima određene osobe izvoditi zaključivanjem na temelju postojanja svojstva *franz:has-child.*
7. Uvest će se dva nova svojstva *franz:has-son* i *franz:has-daughter* kao podsvojstva svojstva *franz:has-child* korištenjem svojstva *rdfs:subPropertyOf.* Time će se postići veća granularnost podataka te omogućiti postavljanje zanimljivijih upita.

Provedbom navedenih koraka dolazimo do konačne ontologije kao sheme baze podataka. Tako definirana ontologija pohranit će se, s određenim brojem instanci, u bazu podataka te će se koristiti za prikaz mogućnosti sustava. Grafički prikaz ontologije dan je na Slici 18.



Slika 18. Konačna verzija ontologije (Izvor: autorova izrada)

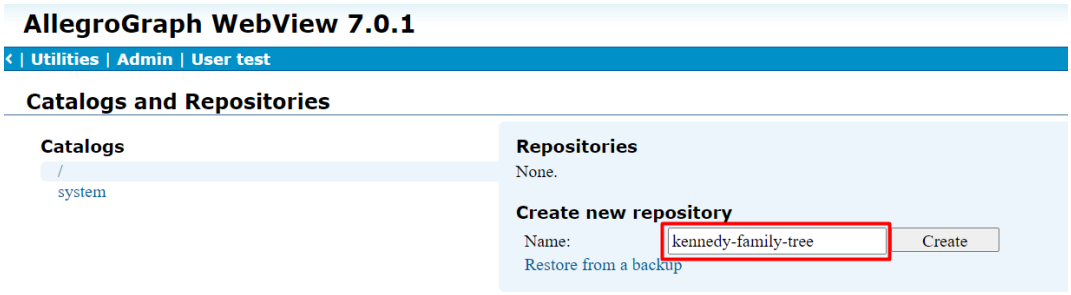
6.5. Izrada RDF baze podataka u AllegroGraph sustavu

Serverska strana AllegroGraph SUBP-a nativno se pokreće na Linux operacijskome sustavu, no moguće ju je pokretati, kroz razne dostupne načine od kojih je najpopularniji korištenjem Linux virtualnih mašina, kako na Windows tako i na Mac računalima [39]. Jedan od najnovijih pristupa korištenja besplatne verzije AllegroGraph sustava, koja dolazi s ograničenjem broja trojki u bazi podataka (maksimalno pet milijuna), pokretanje je istog u Docker kontejneru čija je Docker slika (engl. *Docker image*) dostupna na Docker Hubu (https://hub.docker.com/r/franzinc/agraph). Navedena Docker verzija AllegroGraph sustava obuhvaća i integrirano grafičko sučelje za rad sa AllegroGraph bazom podataka AGWebView te alat za vizualizaciju grafova Gruff. Navedeni alati, iako uključeni kao zadani načini pristupa AllegroGraph sustavu, nisu nametnuti od strane sustava te je moguće koristiti bilo koju klijentsku biblioteku specifičnog podržanog jezika, generalni program za izvršavanje naredbi u konzoli ili pak sam HTTP protokol za pristup serveru.

Nakon pokretanja AllegroGraph Docker kontejnera, serveru je moguće pristupiti preko porta *10035* [39].

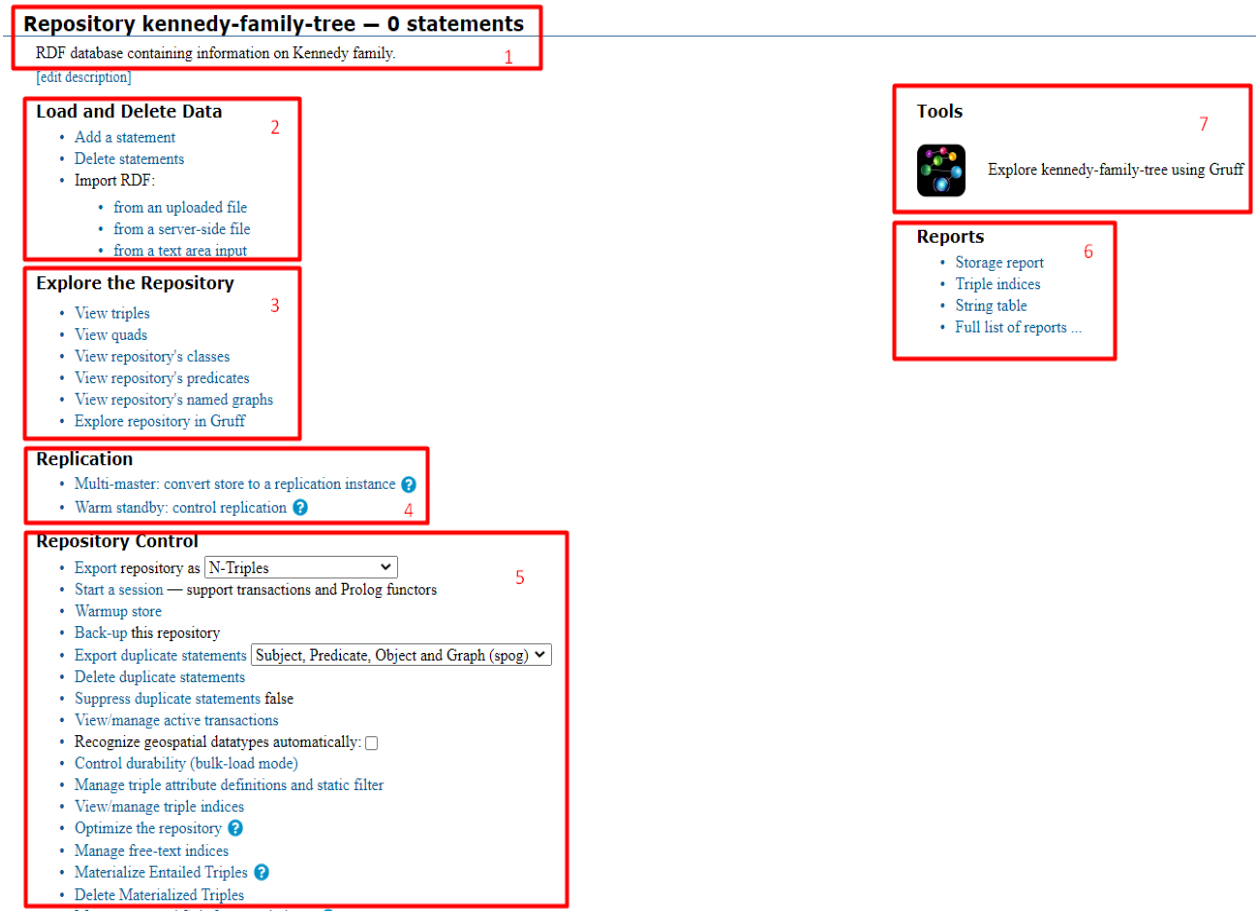
6.5.1. AGWebView alat

Na adresi *localhost:10035* dostupno je AGWebView grafičko sučelje za manipulaciju podacima u AllegroGraph bazi podataka koje je skoro u potpunosti dostatno za prikaz svih ranije opisanih mogućnosti rada s AllegroGraph sustavom, specifično RDF bazom podataka. Mogućnosti koje nisu uključene u AGWebView alat biti će prikazane korištenjem agtool alata za izvršavanje naredbi u konzoli. Nakon uspješne prijave u sustav, kreira se nova ili odabire postojeća baza podataka, Slika 19.



Slika 19. Kreiranje RDF baze podataka (Izvor: autorova izrada)

Nakon kreiranja nove ili odabira postojeće baze podataka, korisnik pristupa početnom zaslonu AGWebView alata koji je podijeljen u nekoliko funkcionalnih cjelina kao što je prikazano na Slici 20.



Slika 20. Početni zaslon AGWebView alata (Izvor: autorova izrada)

1. Naziv baze podataka sa kratkim opisom te informacije o broju pohranjenih trojki.
2. Mogućnosti za dodavanje i brisanje podataka u bazi bilo transakcijskim načinom ili učitavanjem iz datoteke u nekom od prije navedenih podržanih formata.
3. Načini jednostavnog pristupa podacima u bazi prema određenim dijelovima trojki. Moguće je filtrirati klase, predikate, grafove kojima trojke pripadaju ili pak same trojke.
4. Funkcionalnosti vezana uz replikaciju baze podataka.
5. Opcije koje omogućuju provedbu naprednijih operacija nad bazom podataka poput brisanja ponavljajućih trojki i stvaranja sigurnosnih kopija.
6. Prikaz raznih izvještaja o radu baze podataka.
7. Vizualizacija podataka u Gruff alatu.

6.6. Rad s RDF bazom podataka u AllegroGraph sustavu

Za primijetiti je kako se kreiranje baze podataka u AllegroGraph sustavu svodi na pridruživanje naziva istoj. Ne postoji potreba za definiranjem tablica na temelju preddefinirane relacijske sheme, veza između njih i sličnog. Podaci se u kreiranu bazu podataka mogu unositi bez ikakvih dodatnih definicija i bez uvažavanje sheme, ako je ona uopće definirana. U ovom slučaju definirana je i prikazana na Slici 18, a u bazu podataka učitati će se zajedno sa podacima što je i karakteristika RDF baza.

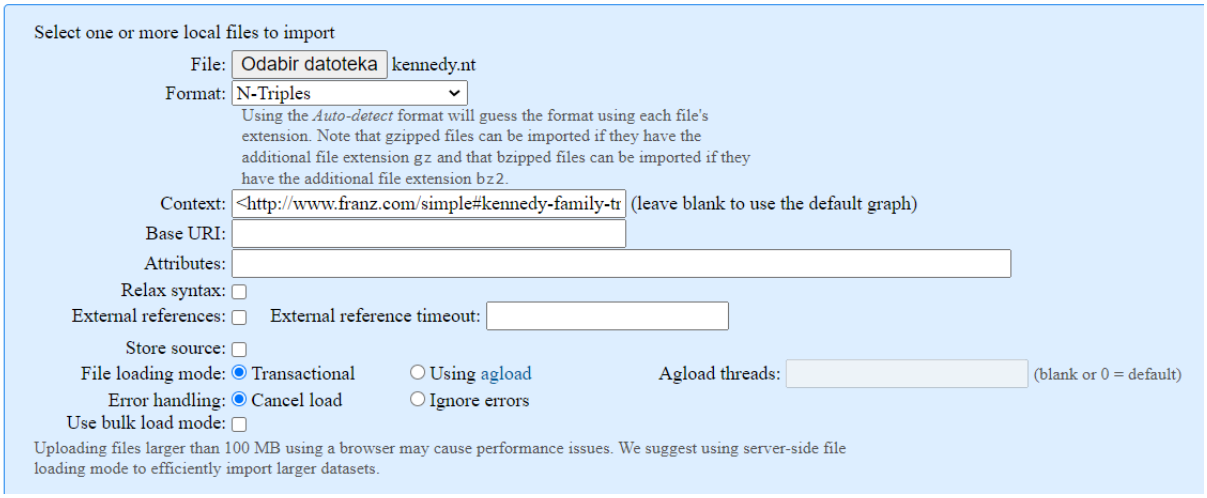
U nastavku rada prikazat će se sljedeće mogućnosti rada s AllegroGraph RDF bazom podataka koristeći alate AGWebView, Gruff te agtool.

1. Unos podataka u bazu podataka
    a. Koristeći učitavanje iz datoteke
    b. Pojedinačan unos trojki kroz AGWebView sučelje
    c. Unos podataka koristeći SPARQL
    d. Provjera konzistentnosti podataka provedbom SHACL validacije
2. Vizualizacija podataka koristeći Gruff
3. Postavljanje upita nad bazom podataka
    a. Postavljanje SPARQL upita
    b. Koristeći Prolog
4. Zaključivanje nad bazom podataka
    a. Koristeći RDFS++ alat
    b. Materijalizacija izvedenih trojki OWL2 RL alatom
5. Indeksiranje baze podataka

6.6.1. Unos podataka u bazu podataka

Prije samog početka prikaza raznih načina manipulacije podacima u AllegroGraph sustavu, potrebno je u kreiranu bazu podataka pohraniti prethodno prikazanu ontologiju nadopunjenu određenim brojem instanci danih klasa. Unos trojki u bazu podataka u AGWebView alatu omogućen je na nekoliko načina od kojih je za unos velikog broja trojki najpovoljniji učitavanjem podataka iz datoteke, danih u nekom od podržanih RDF formata za serijalizaciju. Za unos manjeg broja ili pak pojedinačnih trojki moguće je koristiti AGWebView editor te SPARQL INSERT naredbu.

6.6.1.1. Učitavanje podataka iz datoteke



Slika 21. Učitavanje podataka iz datoteke u bazu podataka (Izvor: autorova izrada)

Podaci u datoteci *kennedy.nt,* kao što ekstenzija .nt sugerira, serijalizirani su u N-Triples formatu. AllegroGraph sustav će, neovisno o formatu kojim su RDF podaci serijalizirani, podatke pohraniti i povezati u jedan ili više RDF grafova što naravno ovisi o vezama između trojki.

Kod samog učitavanja podataka iz datoteke, između ostalog moguće je definirati i kontekst podataka odnosno graf kojem pripadaju. Time dane trojke postaju četvorke što je vrlo korisno kada u istu bazu podataka nastojimo pohraniti više RDF grafova koji pripadaju različitim domenama. Na taj se način SPARQL upiti mogu efikasno lokalizirati, a u bilo kojem trenutku isti se na temelju novih informacija mogu povezati u jedinstveni RDF graf. Tako je kod unosa podataka iz dane datoteke definiran i graf kojem te iste trojke pripadaju:

- <http://www.franz.com/simple#kennedy-family-tree>

Polje grafa moguće je definirati i u samoj datoteci koja se učitava, a ona u tom slučaju mora biti u N-Quads formatu. N-Quads se od N-Triples formata razlikuje po uvođenje četvrtog elementa u opis trojke koji se odnosi na graf kojem ona pripada. Primjerice:

```
<http://www.franz.com/simple#person1>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.franz.com/simple#person>
<http://www.franz.com/simple#kennedy-family-tree> .
```

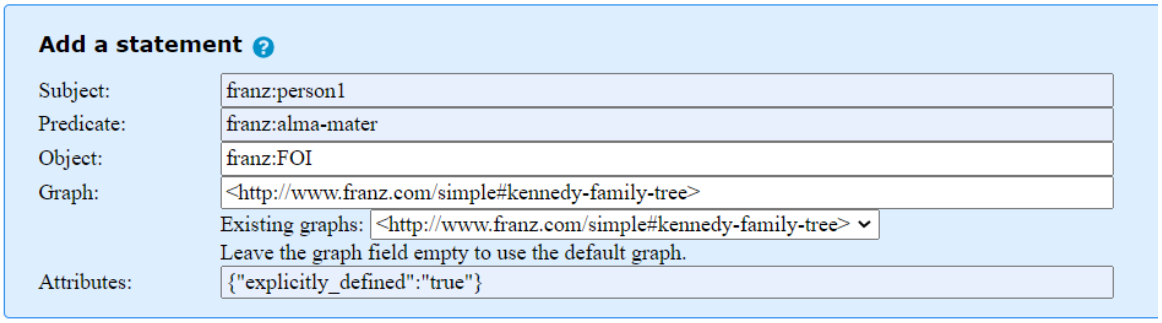
Također je moguće svim trojkama koje će se unijeti u bazu pridružiti atribute koji se mogu koristit u razne svrhe, primjerice kontrolu pristupa pojedinačnim trojkama.

Ovisno o broju trojki koje se nastoje unijeti u bazu podataka moguće je promijeniti način učitavanja iz transakcijskog koji podrazumijeva ACID svojstva u masovni način učitavanja (engl. *Bulk load mode*) koji uklanja transakcije, čime se učitavanje podataka može višestruko ubrzati.

6.6.1.2. Pojedinačan unos trojki u AGWebView sučelju

Podaci se u kreiranu bazu podataka mogu unositi i pojedinačno na nekoliko načina od kojih je zadani i najjednostavniji način kroz AGWebView sučelje.

Kako bi se pojednostavio unos trojki, kao i sve druge mogućnosti rada sa podacima, AllegroGraph omogućuje definiranje korisničkih prefiksa za područja imena te dolazi sa nekoliko unaprijed zadanih prefiksa za konstrukte iz RDF, RDFS i OWL sintakse. Definiranjem prefiksa *franz* za URI http://www.franz.com/simple# moguće je definirati novu trojku kao na Slici 22.



Slika 22. Unos trojke u bazu podataka koristeći AGWebView sučelje (Izvor: autorova izrada)

Tako definirana trojka dodat će se u *franz:kennedy-family-tree* graf kao zapis koji se odnosi na pohađani fakultet osobe *franz:person1.* RDF trojci bit će pridružen i atribut koji označava trojku kao eksplicitno unesenu u bazu podataka, a ne izvedenu temeljem zaključivanja. Atribut je potrebno prethodno definirati unutar samog sustava kako bi se mogao pridružiti trojkama kod unošenja bilo pojedinačnog ili masovnog učitavanjem iz datoteke.

6.6.1.3. Unos trojki koristeći SPARQL

Još jedan od načina unošenja trojki u bazu podataka korištenjem je INSERT naredbe SPARQL upitnog jezika što je vrlo pogodno ukoliko unos trojke ovisi o određenim preduvjetima koji se također mogu izraziti SPARQL sintaksom. Sljedeća SPARQL naredba će postojećem *franz:kennedy-family-tree* grafu pridružiti novu instancu klase *franz:person.*

```
INSERT DATA
      { GRAPH <franz:kennedy-family-tree>
        { franz:person112  rdf:type  franz:person }
      }
```

6.6.1.4. SHACL validacija podataka

Vratimo se na trenutak natrag na Sliku 22., na primjer u kojem se u bazu podataka unijela informacija kako je jedan od članova obitelji Kennedy završio fakultet *franz:FOI.* Svojstvo *franz:alma-mater* koje povezuje osobu sa fakultetom prema definiranoj shemi ograničeno je dvama svojstvima iz *rdfs* jezika: *rdfs:domain* koje definira subjekt trojke kao instancu klase *franz:person* te *rdfs:range* koje definira objekt trojke kao instancu klase *franz:College* ili neke od njenih podklasa (*franz:Ivy-league-college*).

Nažalost ili nasreću dana svojstva neće ograničiti vrijednosti subjekta kao ni objekta trojke na instance spomenutih klasa, što bi prema direktnoj interpretaciji svojstava zapravo bio njihov cilj, te samim time spriječiti ulazak baze u nekonzistentno stanje. Semantika danih svojstava zapravo je sljedeća: Ako definiramo trojke (*P rdfs:domain D i P rdfs:range R)* i trojka (*s P o)* postoji tada se provedbom zaključivanja iz podataka mogu izvesti trojke

- *s rdf:type D* i
- *o rdf:type R*

Kao što je već nekoliko puta kroz rad napomenuto, RDFS kao ni OWL ne ograničavaju unos trojki te ne služe za održavanje baze podataka u konzistentnome stanju već se koriste za definiranje sheme baze podataka te provedbu zaključivanja s ciljem izvlačenja novih informacija iz eksplicitno pohranjenih podataka. Kako bismo provjerili konzistentnost baze podataka AllegroGraph podržava SHACL jezik kroz alat agtool.

Validacija baze podataka koristeći SHACL provodi se usporedbom grafa podataka, odnosno samog RDF grafa pohranjenog u bazi, te takozvanog grafa uvjeta definiranog sintaksom iz SHACL jezika.

Kako bi zaista ograničili vrijednosti subjekt i objekta u trojci u kojoj je predikat *franz:alma-mater* na redom, instance klasa *franz:person* te *franz:College,* potrebno je u samu bazu podataka unijeti graf uvjeta kao u sljedećem primjeru.

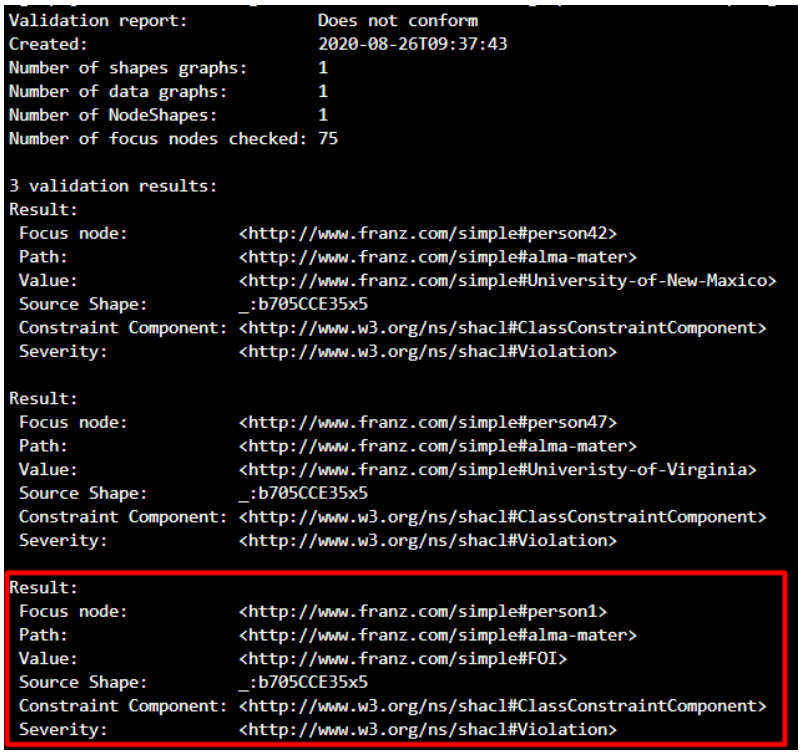
```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix franz: <http://www.franz.com/simple#> .
<http://franz.com#Shapes> {
<http://franz.com#AlmaMaterShape>
  a sh:NodeShape ;
   sh:targetClass franz:person ;
    sh:property [
         sh:path franz:alma-mater ;
         sh:class franz:College
    ] .
}
```

Nakon definiranja prefiksa za SHACL rječnik te područje imena koje obuhvaća podatke iz definirane i pohranjene ontologije i naziva grafa uvjeta *Shapes*, definiran je novi uvjet *AlmaMaterShape* tipa *sh:NodeShape* koji se odnosi na sve čvorove tipa *franz:person* što je zadano sa *sh:targetClass.* Svi takozvani ciljani čvorovi, čvorovi na koje se uvjet odnosi, moraju zadovoljavati ograničenje dano u *sh:property.* Prema definiranome ograničenju svojstvo *franz:alma-mater* čvora *franz:person* mora za vrijednost imati instancu klase *franz:College.*

Nakon unosa danog SHACL grafa uvjeta u postojeću bazu podataka moguće je provesti validaciju RDF grafa podataka koristeći agtool naredbu:

- *agtool shacl-validate --data-graph http://www.franz.com/simple#kennedy-family-tree --shapes-graph http://franz.com#Shapes kennedy-family-tree*

Rezultat validacije, dan na Slici 23, jasno upućuje na nekonzistentnost baze podataka jer resurs *franz:FOI* nije instanca klase *franz:College.* Isto tako, iz validacije ostalih 74 trojki u kojima se kao predikat pojavljuje svojstvo *franz:alma-mater,* pojavile su se još dvije trojke koje nisu u skladu s danim ograničenjem čemu su uzrok pravopisne greške. Vrlo važno je naglasiti kako provedena SHACL validacija nije ispravila nekonzistentnosti u bazi podataka već samo ukazala na iste. Na korisniku je da pogreške ispravi.

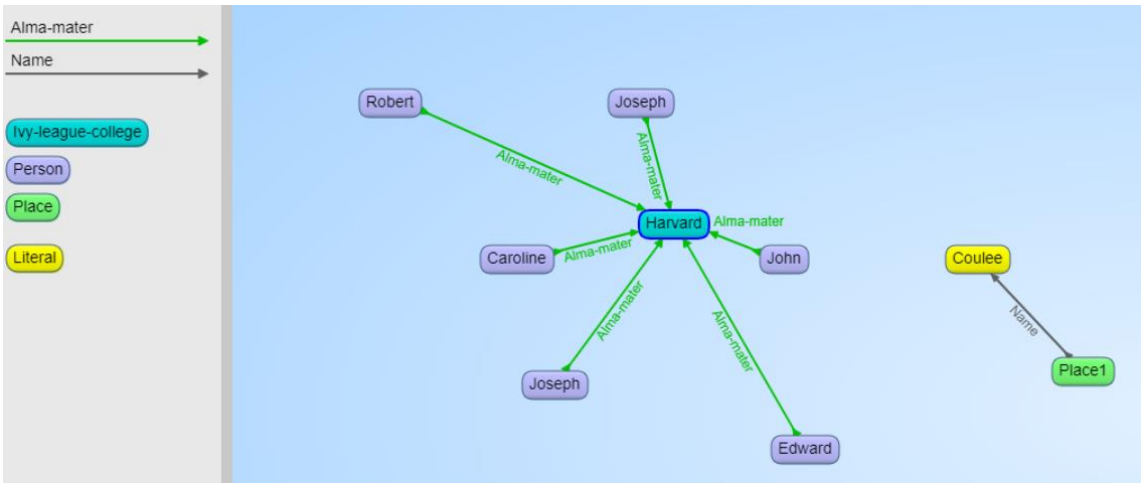


Slika 23. Rezultat SHACL validacije (Izvor: autorova izrada)

6.6.2. Vizualizacija podataka u Gruff alatu

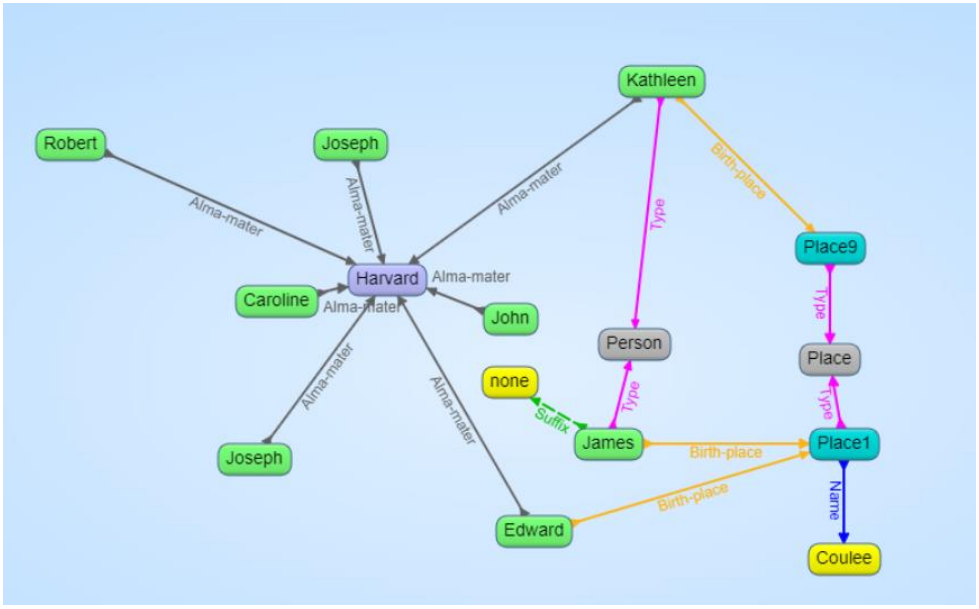
Gruff je alat za vizualizaciju grafa te grafičku izradu upita koji se može povezati s velikim broj SPARQL pristupnih točaka tako da nije isključivo vezan uz AllegroGraph sustav [40]. Nije razvijen za upravljanje bazama podataka već služi isključivo za rad s grafom, otkrivanje relevantnih veza između podataka te skrivenih uzoraka. Navedene mogućnosti povezane sa funkcionalnošću kreiranja SPARQL upita na vizualan način, dodavanjem čvorova i veza između njih grafu u obliku varijabli, čine Gruff vrlo praktičnim načinom pregleda te otkrivanja informacija u grafu.

Graf se u Gruff alatu može konstruirati na mnogo načina od kojih su najčešći temeljem SPARQL upita te traženjem najkraćih putanja između čvorova, to jest traženjem poveznica između podataka. Sljedeći primjer prikazuje kako se dva grafa u Gruff alatu povezuju temeljem pretrage najkraćih putanja između čvorova *Robert* i *place1.*



Slika 24. Vizualizacija nepovezanih podgrafova RDF grafa pohranjenog u bazi podataka (Izvor: autorova izrada)

Na Slici 24 prikazana su dva podgrafa ontologije pohranjene u bazi podataka čiji su čvorovi i veze opisani klasama i svojstvima u gornjem lijevom kutu slike. Jedan od načina povezivanja danih grafova pretragom je najkraće putanje između njihovih čvorova. Pretragom najkraće putanje između čvorova *Robert* klase *Person* i *place1* klase *Place* dobivamo rezultantni graf dan na Slici 25.



Slika 25. Vizualizacija grafa dobivenog pretragom najkraće putanje (Izvor: autorova izrada)

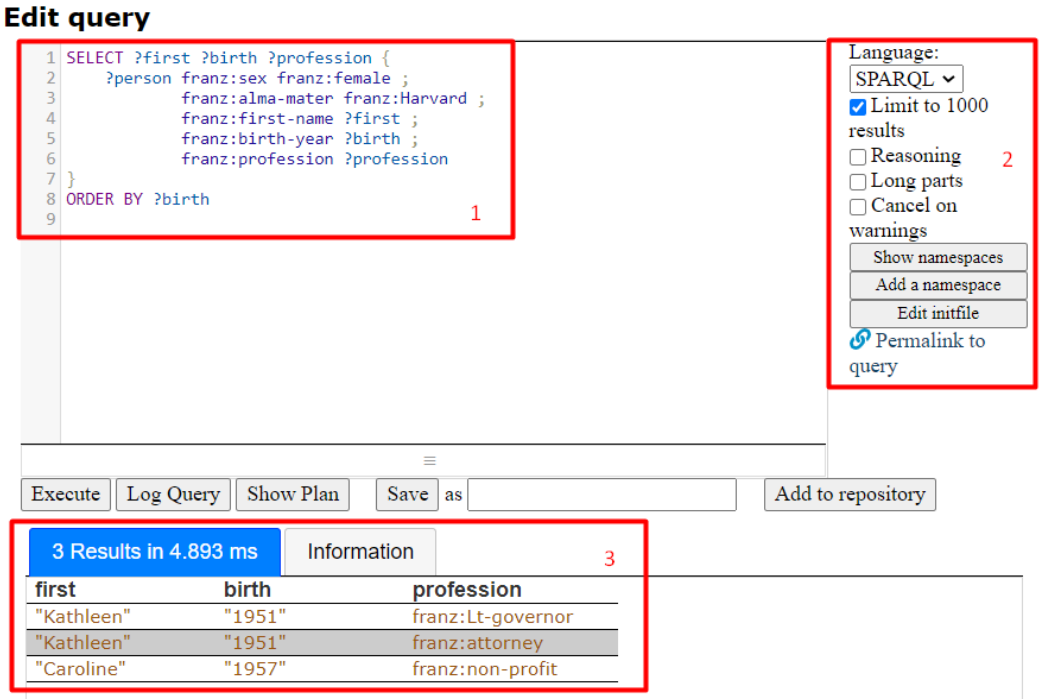
Pronađene su tri putanje kojima se povezuju prije dani čvorovi od kojih je najzanimljivija veza iz koje saznajemo kako je Edward Kennedy, koji je pohađao sveučilište Harvard baš kao i njegov rođak Robert, rođen u gradu Colueeu.

6.6.3. Postavljanje upita nad bazom podataka

Baš kao što SQL jezik u relacijskim bazama podataka predstavlja temeljni pristup podacima, tako se kompleksne analize podataka u AllegroGraph sustavu vrše standardiziranim SPARQL upitnim jezikom. Za potrebe rada sa podacima koje nadilaze mogućnosti SPARQL jezika korisnicima je omogućen rad s bazom koristeći Prolog. U sljedećim poglavljima prikazat će se manipuliranje podacima u bazi koristeći upravo ta dva navedena pristupa.

6.6.3.1. SPARQL upiti

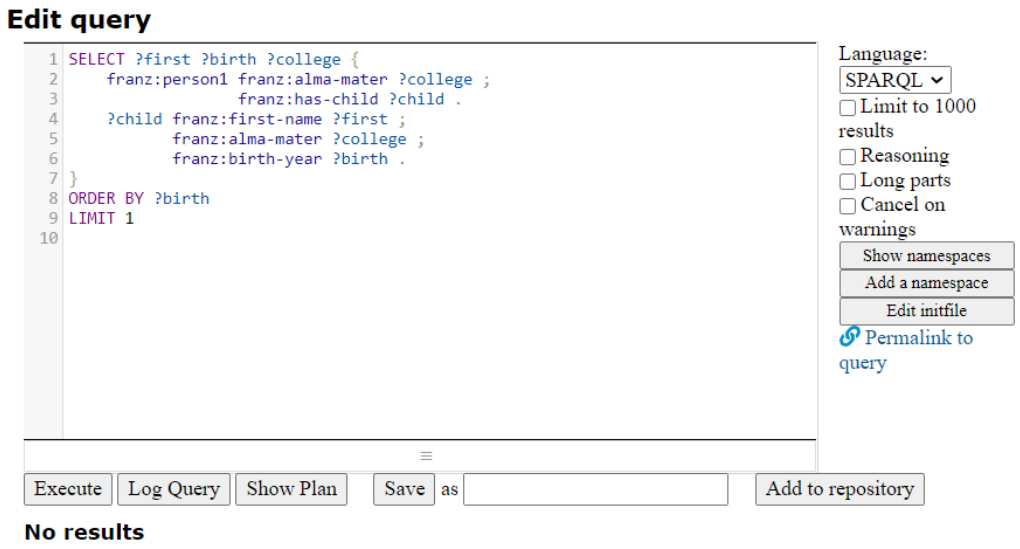
Na slici 26 prikazan je relativno jednostavan SPARQL SELECT upit (okvir 1) izveden unutar AGWebView sučelja za postavljanje upita.



Slika 26. Prikaz jednostavnog SPARQL upita (Izvor: autorova izrada)

Rezultat upita vraća podatke o ženama iz obitelji Kennedy koji su pohađali sveučilište Harvard te njihovim zanimanjima, sortirano prema njihovoj godini rođenja, te je prikazan u obliku tablice (okvir 3). Prozor za postavljanje SPARQL upita u AGWebViewu alatu nudi i dodatne mogućnosti (okvir 2) prebacivanja jezika na Prolog, uključivanja RDFS++ zaključivanja te prikaza prostora imena u rezultatima.

Na Slici 27 dan je još jedan SPARQL upit kojim se žele pronaći najstarije dijete patrijarha obitelji Kennedy, Josepha Kennedyja, koje je studiralo na istome sveučilištu kao i on. Zanimljivo je kako upit, koji bi prema podacima u bazi trebao vratiti rezultat, ne rezultira podacima. Razlog tome je u definiciji same sheme baze, odnosno ontologije i biti će detaljno objašnjen nakon sljedećeg poglavlja u kojem će se prikazati postavljanje upita nad bazom koristeći Prolog.



Slika 27. SPARQL upit bez uključenog zaključivanje nad bazom podataka (Izvor: autorova izrada)

6.6.3.2. Prolog upiti

Prolog predstavlja alternativan način postavljanja upita nad AllegroGraph bazom podataka koji se može iskoristiti u slučajevima kada korisnikove potrebe nije moguće zadovoljiti mogućnostima SPARQL jezika. AGWebView omogućuje direktan unos Prolog koda u prije prikazani editor.

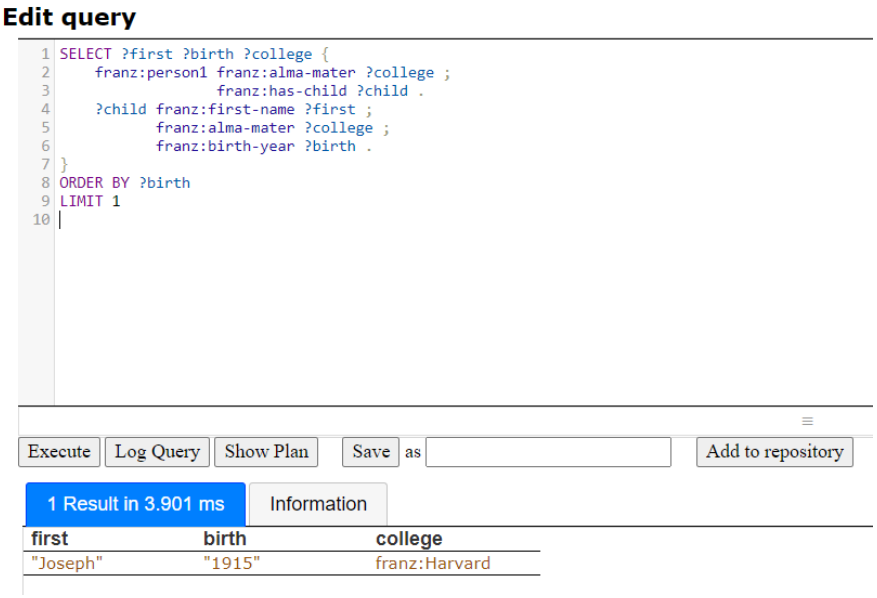
Sljedeća Prolog naredba vraća sve sinove osobe *franz:person1.*

```
(select (?son)
        (q- !franz:person1 !franz:has-son ?son))
```

Sintaksa Prolog upita prilagođena je radu sa RDF trojkama definiranjem posebnih predikata. Tako svaki *select* Prolog upit, nakon što se definiraju varijable koje će se vratiti u rezultatu upita, sadrži određen broj klauzula koje određuju na koji će se način provesti zamjena varijabli stvarnim vrijednostima trojki. Unutar klauzula pojavljuju se spomenuti predikati. Predikat *g-* tako služi za pridruživanje vrijednosti varijabli *?son* koje odgovaraju objektima trojki sa subjektom *franz:person1* te predikatom *franz:has-son.* Uskličnik prije prefiksa određenog područja imena sintaksno je pravilo korištenja istih u Prolog upitima i kodu [41].

6.6.4. Zaključivanje nad bazom podataka

U prethodnom poglavlju u kojem se pokazala mogućnost postavljanja SPARQL upita postavljen je upit koji nije vratio očekivanu RDF trojku. Razlog tome je to što u bazi podataka nije eksplicitno definirana ni jedna trojka u kojoj bi osoba *franz:person1* bila povezana s nekom drugom osobom kroz svojstvo *franz:has-child.* Roditelji su sa svojom djecom u bazi podataka povezana svojstvima izvedenim iz svojstva *franz:child, franz:has-son* i *has-daughter.* Kako su navedena svojstva definirana koristeći *owl:subPropertyOf* pronalazak djece određene osobe ostvarit će se korištenjem alata za zaključivanje. RDFS++ alat za zaključivanje na temelju eksplicitno pohranjenih trojki oblika: <*a*> <*franz:has-son*>/<*franz:has-daughter*> <*b*> izvodi trojke oblika: <*a*> <*has-child*> <*b*>. Uključivanje alata za zaključivanje kod postavljanja prethodno prikazanog upita dovodi do sljedećih očekivanih rezultata danih na Slici 28. Naravno, korisnik u svakome trenutku može i eksplicitno pridružiti djecu roditeljima koristeći *has-child* predikat.



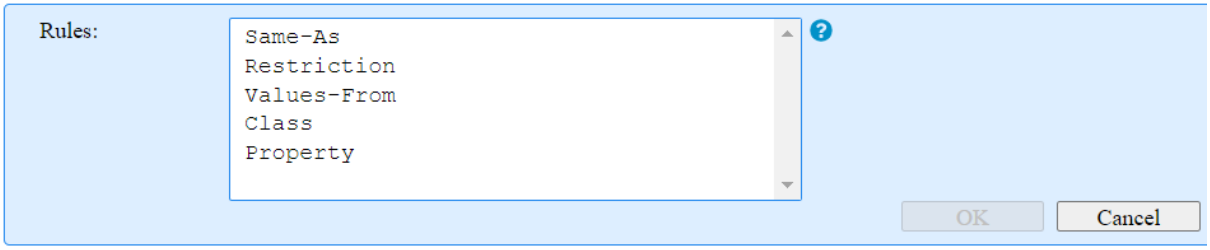
Slika 28. SPARQL upit s uključenim RDFS++ zaključivanjem nad bazom podataka (Izvor: autorova izrada)

6.6.4.1. Materijalizacija podataka izvedenih zaključivanjem

Trojke proizašle iz zaključivanja nad RDF grafom korištenjem RDFS++ alata za zaključivanje neće se trajno pohraniti u bazi podataka. U svrhu materijalizacije trojki moguće je koristiti OWL2 RL alat za zaključivanje, dostupan unutar AGWebView alata.

OWL2 RL alat koristi set pravila za zaključivanje za izvođenje novih trojki te ih dodaje u bazu podataka. Nakon toga izvedene trojke moguće je koristiti no, dodatnim modifikacijama baze, bilo uslijed operacija brisanja, ažuriranja ili dodavanja trojki, moguće je da trojke izvedene zaključivanjem neće vrijediti te će se cijeli postupak zaključivanja trebati ponoviti što iziskuje određeno vrijeme kao i računalne kapacitete ako se radi o bazi podataka s velikim brojem trojki.

AllegroGraph prije provođenja materijalizacije izvedenih trojki, OWL2 RL alatom za zaključivanje, od korisnika zahtijeva da samostalno odabere setove pravila, definirane unutar OWL2 RL specifikacije i detaljno sistematizirane u AllegroGraph dokumentaciji [42], koji će se koristiti za zaključivanje, Slika 29.



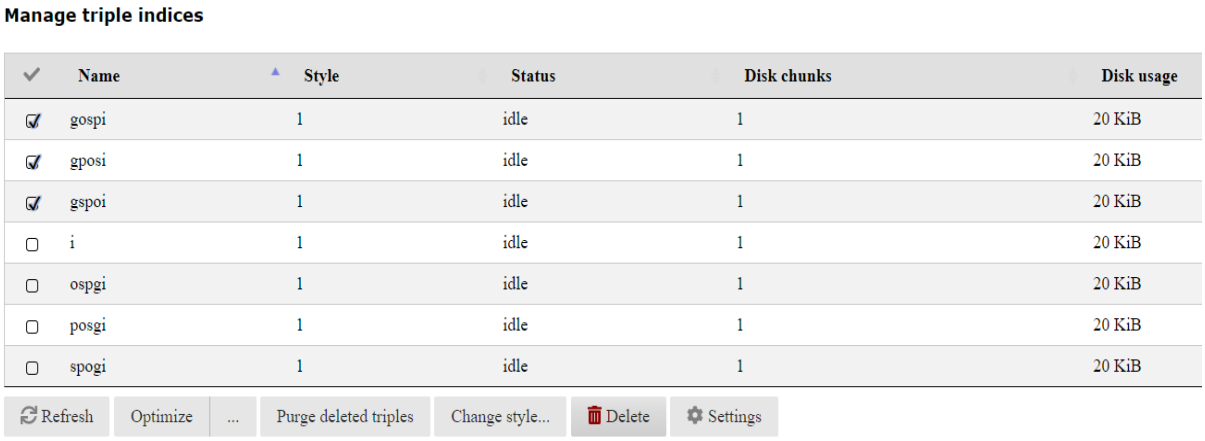
Slika 29. Odabir setova pravila za OWL2 materijaliziranje trojki (Izvor: autorova izrada)

Prema implementiranoj ontologiji svojstvo *franz:has-parent* definirano je kao inverzno svojstvu *franz:has-child* te kao takvo nije eksplicitno pridruženo ni jednoj trojki u bazi podataka. Kako bi se informacija o roditeljima određene osobe izvela na temelju informacija o djeci potrebno je provesti zaključivanje nad bazom bilo RDFS++ ili OWL2 RL alatom. Ukoliko želimo materijalizirati izvedene trojke koristit ćemo OWL2 RL alat što će rezultirati eksplicitno pohranjenim trojkama sa predikatom *franz:has-parent* kao na Slici 30. Za primijetiti je kako RDFS++ zaključivanje nije uključeno.

Slika 30. Prikaz rezultata OWL2 RL materijalizacije (Izvor: autorova izrada)

6.6.5. Indeksiranje baze podataka

Svrha indeksiranja, kao što je već navedeno, je ubrzavanje izvršavanja upita nauštrb smanjenja brzine unosa podataka u bazu, kao i zauzeća memorijskog prostora. Kako je kreirana baza podataka relativno mala, njenim indeksiranjem ne ostvaruju se bitni pozitivni učinci na brzinu izvršavanja upita, no ono je ipak izvedeno kroz sedam zadanih indeksa unutar AllegroGraph sustava.



Slika 31. Prikaz indeksa kreiranih nad bazom podataka (Izvor: autorova izrada)

Na Slici 31 vidljivo je svih sedam zadanih indeksa kojima je moguće upravljati. Primjerice, kako se ne predviđa korištenje polja grafa u upitima koji će se postavljati nad bazom, indeksi sa oznakom grafa *g* mogu se obrisati čime će se (u slučaju baza podataka s milijunima trojki) bitno smanjiti zauzeće memorijskog prostora te ubrzati unos u bazu podataka.

Prethodno poglavlje nadovezalo se na teoretski dio rada te se na jednostavan i razumljiv način prikazalo na koji način AllegroGraph sustav koristi tehnologije semantičkog weba za efikasno rukovanje podacima. Primjeri su izvedeni nad proširenom ontologijom obiteljskog stabla obitelji Kennedy, odnosno shemom baze podataka, koristeći alate sadržane u AllegroGraph serveru: AGWebView, Gruff te agtool, koji su samo jedan od načina rada s AllegroGraph bazom podataka.

7. Zaključak

AllegroGraph je multi-model sustav koji objedinjuje tri različita pristupa pohrani podataka te se može koristiti kao grafovska, dokument te RDF baza podataka. Strukturiranje rada na način da se prije samog povezivanja navedenih pristupa pohrani podataka AllegroGraph sustavom ostvari dublje razumijevanje istih kroz teoretsku analizu imalo je vrlo pozitivan utjecaj na shvaćanje kako samog sustava tako i pojma multi-model baza podataka. Naime, danas vrlo česti pristup objedinjavanja većeg broja modela podataka unutar jednog sustava, barem u slučaju AllegroGraph sustava, nikako nije zasnovan na iskorištavanju relativno malih razlika između modela podataka već na kontinuiranom radu na poboljšanju postojećeg modela. Upravo kao što je u radu prikazano, grafovska strana sustava proizašla iz RDF modela podataka, kao i spremanje podataka u dokument modelu temeljeno na jednome od formata za serijalizaciju RDF-a, JSON-LD, konstantno su unapređivani sa ciljem stvaranja razlika između tehnologija te neovisnosti između njih. Upravo zbog navedenog AllegroGraph, iako temeljen na standardima proizašlim iz područja semantičkog weba, omogućuje korisnicima da sustav koriste bez poznavanja cijele hrpe njima nevažnih tehnologija.

Unatoč tome, jedan od ciljeva proizašlih iz same motivacije za obradu ove teme rada, bio je prikaz korisnosti, upotrebljivosti te samog mjesta u području manipulacije podacima koje zauzimaju te iste tehnologije. Kroz praktični dio rada u kojem se, u AllegroGraph sustav integriranim alatima za upravljanje bazama podataka AGWebView te vizualizaciju grafa Gruff, razvila RDF baza podataka nad kojom su nakon toga provedene i prikazane tipične operacije manipulacije podacima poput postavljanja upita te one, RDF svijetu karakteristične, operacije poput zaključivanja nad podacima, došlo se do pozitivnih zaključaka o većini njih. Ostvario se pogled na RDF kao vrlo jednostavan, ali u isto vrijeme vrlo moćan model podataka koji u tandemu s jezicima koji ga nadopunjuju, RDFS i OWL, te jezikom koji omogućuje pretraživanje istog to jest SPARQL, dokazuje kako je ideja semantičkog weba u kojem svako od nas, kao i svaka druga opipljiva ili apstraktna stvar, na webu ima svoj vlastiti jedinstveni API jednostavno previše privlačna da bi se ikada napustila.

Na kraju ovog rada potrebno je naglasiti kako je AllegroGraph SUBP vrlo kompleksan sustav koji je pod stalnim stanjem razvoja i prilagođavanja svijetu u kojem su podaci najveće strateško oružje bilo koje domene ljudskog djelovanja te će svojim jedinstvenim pristupima pohrani i manipulaciji podacima uvijek imati značajnu ulogu na tržištu.

Popis literature

[1] Two-Bit History (29.12.2017.) *Important Papers: Codd and the Relational Model* [Na internetu]. Dostupno: https://twobithistory.org/2017/12/29/codd-relational-model.html [pristupano 15.09.2020.]

[2] Christof Strauch, *NoSQL Databases*. Hochschule der Medien, Stuttgart, 2019, Dostupno: https://christof-strauch.de/nosqldbs.pdf *[pristupano 12.04.2020.].*

[3] Bryce Merkl Sasaki (25.10.2018.) *Graph Databases for Beginners: Why We Need NoSQL Databases* [Na internetu]. Dostupno: https://neo4j.com/blog/why-nosql-databases/?ref=blog [pristupano 13.04.2020.]

[4] Adam Keys (bez dat.) *It's not NoSQL, it's post-relational* [Na internetu]. Dostupno: https://therealadam.com/2009/08/31/its-not-nosql-its-post-relational/ [pristupano 13.04.2020.]

[5] Renzo Angles i Claudio Gutierrez, *Survey of Graph Database Models*. Universidad de Chile, Chile, 2008, Dostupno: https://users.dcc.uchile.cl/~cgutierr/papers/surveyGDB.pdf *[pristupano 12.04.2020.].*

[6] M. Besta, E. Peter, R. Gerstenberger, M. Fischer, M. Podstawski, C. Barthels, G. Alonso, T. Hoefler, *Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries,* 04.03.2020., Dostupno: https://arxiv.org/abs/1910.09017 [Na internetu]. [pristupano 13.04.2020.]

[7] Ian Robisnon, Jim Webber i Emil Eifrem, Graph Databases. USA: O'Reilly Media, Inc. 2015. izd. 2.

[8] K. Rabuzin i M. Šestak, *Grafovske baze podataka – pregled istraživanja i budućih trendova*. Sveučilište u Zagrebu. Fakultet organizacije i informatike, Varaždin, Republika Hrvatska, Dostupno: https://bib.irb.hr/datoteka/940636.Grafovske_baze_podataka__pregled_istrazivanja_i_buduci h_trendova.pdf *[pristupano 12.04.2020.].*

[9] Vojtěch Kolomičenko, *Analysis and Experimental Comparison of Graph Databases* [Master thesis]. Charles University in Prague Faculty of Mathematics and Physics, Prag, 2013, Dostupno: *www.ksi.mff.cuni.cz/~holubova/dp/Kolomicenko.pdf [pristupano 15.04.2020.].*

[10] Packt (bez dat.) *Storage – native graph storage versus non-native graph storage* [Na internetu]. Dostupno: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781784393441 /1/ch01lvl1sec10/storage-native-graph-storage-versus-non-native-graph-storage [pristupano 15.04.2020.]

[11] mongoDB (bez dat.) *What is a document database?* [Na internetu]. Dostupno: https://www.mongodb.com/document-databases [pristupano 22.06.2020.]

[12] Marc Clifton (25.10.2014.) *Semantic Database: Concept, Architecture and Implementation* [Na internetu]. Dostupno: https://www.codeproject.com/Articles/832959/Semantic-Database-Concept-Architecture-and-Impleme [pristupano 13.04.2020.]

[13] TwoBitHistory (19.06.2020.) *Whatever Happened to the Semantic Web?* [Na internetu]. Dostupno: https://twobithistory.org/2018/05/27/semantic-web.html [pristupano 19.06.2020.]

[14] D. C. Faye, O. Cure, G. Blin, *A survey of RDF storage approaches*. Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées, 2012, Dostupno: https://hal.inria.fr/hal-01299496/document *[pristupano 15.04.2020.].*

[15] ontotext (2014.) *THE TRUTH ABOUT TRIPLESTORES* [Na internetu]. Dostupno: https://ontotext.com/wp-content/uploads/2014/07/The-Truth-About-Triplestores.pdf [pristupano 13.04.2020.]

[16] Jesus Barrasa (18.08.2017.) *RDF Triple Stores vs. Labeled Property Graphs: What's the Difference?* [Na internetu]. Dostupno: https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/ [pristupano 13.04.2020.]

[17] Parthenos (2020.) *What is a Formal Ontology?* [Na internetu]. Dostupno: https://training.parthenos-project.eu/sample-page/formal-ontologies-a-complete-novices-guide/what-is-a-formal-ontology/
[pristupano 13.04.2020.]

[18] W3C (25.02.2014.) *RDF Schema 1.1* [Na internetu]. Dostupno: https://www.w3.org/TR/rdf-schema/ [pristupano 13.04.2020.]

[19] Cambridge Semantics (bez dat.) *RDFS vs. Owl* [Na internetu]. Dostupno: https://www.cambridgesemantics.com/blog/semantic-university/learn-owl-rdfs/rdfs-vs-owl/ [pristupano 13.04.2020.]

[20] W3C (2015.) *Inference* [Na internetu]. Dostupno: https://www.w3.org/standards/semanticweb/inference [pristupano 13.04.2020.]

[21] Cambridge Semantics (bez dat.) *Flavors of OWL* [Na internetu]. Dostupno: https://www.cambridgesemantics.com/blog/semantic-university/learn-owl-rdfs/flavors-of-owl/ [pristupano 13.04.2020.]

[22] SPIN RDF (bez dat.) *SHACL and OWL Compared.* [Na internetu]. Dostupno: https://spinrdf.org/shacl-and-owl.html [pristupano 28.06.2020.]

[23] W3C (20.07.2017.) *Shapes Constraint Language (SHACL)* [Na internetu]. Dostupno: https://www.w3.org/TR/shacl/ [pristupano 28.06.2020.]

[24] Angus Addlesee (10.10.2018.) *Understanding Linked Data Formats* [Na internetu]. Dostupno: https://medium.com/wallscope/understanding-linked-data-formats-rdf-xml-vs-turtle-vs-n-triples-eb931dbe9827 [pristupano 13.04.2020.]

[25] Mark Watson, *Practical Semantic Web and Linked Data Applications*, 2011, Dostupno: https://www.markwatson.com/opencontent_data/book_java.pdf *[pristupano 15.04.2020.].*

[26] ontotext (bez dat.) *What is SPARQL* [Na internetu]. Dostupno: https://www.ontotext.com/knowledgehub/fundamentals/what-is-sparql/ [pristupano 18.06.2020.]

[27] AllegroGraph Documentation (19.06.2020.) *AllegroGraph 7.0.1 Reasoner Tutorial* [Na internetu]. Dostupno: https://franz.com/agraph/support/documentation/current/reasoner-tutorial.html [pristupano 19.06.2020.]

[28] Reference for Business (bez dat.) *Franz Inc. - Company Profile, Information, Business Description, History, Background Information on Franz Inc.* [Na internetu]. Dostupno: https://www.referenceforbusiness.com/history2/33/Franz-Inc.html [pristupano 22.06.2020.]

[29] DB-ENGINES (Lipanj 2020.) *DB-Engines Ranking.* [Na internetu]. Dostupno: https://db-engines.com/en/ranking [pristupano 22.06.2020.]

[30] AllegroGraph.com (bez dat.) *AllegroGraph Overview.* [Na internetu]. Dostupno: https://allegrograph.com/products/allegrograph/ [pristupano 22.06.2020.]

[31] AllegroGraph Documentation (19.06.2020.) *AllegroGraph 7.0.1 HTTP.* [Na internetu]. Dostupno: https://franz.com/agraph/support/documentation/current/http-protocol.html [pristupano 22.06.2020.]

[32] AllegroGraph Documentation (19.06.2020.) *AllegroGraph 7.0.1 Introduction.* [Na internetu]. Dostupno: https://franz.com/agraph/support/documentation/current/agraph-introduction.html#ai-overview [pristupano 28.06.2020.]

[33] AllegroGraph Documentation (19.06.2020.) *AllegroGraph 7.0.1 Data Import* [Na internetu]. Dostupno: https://franz.com/agraph/support/documentation/current/agload.html#specifying-sources [pristupano 22.06.2020.]

[34] AllegroGraph Documentation (19.06.2020.) *AllegroGraph 7.0.1 Triple Indices* [Na internetu]. Dostupno: https://franz.com/agraph/support/documentation/current/triple-index.html [pristupano 22.06.2020.]

[35] AllegroGraph Documentation (19.06.2020.) *AllegroGraph 7.0.1 and SHACL* [Na internetu]. Dostupno: https://franz.com/agraph/support/documentation/current/shacl.html [pristupano 22.06.2020.]

[36] AllegroGraph Documentation (19.06.2020.) *SPARQL API reference* [Na internetu]. Dostupno: https://franz.com/agraph/support/documentation/current/sparql-reference.html [pristupano 22.06.2020.]

[37] JSON-LD (bez dat.) *JSON for Linking Data* [Na internetu]. Dostupno: https://json-ld.org/ [pristupano 22.06.2020.]

[38] AllegroGraph Documentation for Python Client (bez dat.) *Example 19: Using JSON-LD* [Na internetu]. Dostupno: https://franz.com/agraph/support/documentation/current/python/tutorial/example019.html [pristupano 22.06.2020.]

[39] AllegroGraph Documentation (19.06.2020.) *AllegroGraph 7.0.1 Quick Start* [Na internetu]. Dostupno: https://franz.com/agraph/support/documentation/current/agraph-quick-start.html [pristupano 22.06.2020.]

[40] AllegroGraph Documentation (19.06.2020.) *Gruff in AllegroGraph 7.0.1* [Na internetu]. Dostupno: https://franz.com/agraph/support/documentation/current/gruff-in-ag.html [pristupano 22.06.2020.]

[41] AllegroGraph Documentation (19.06.2020.) *Using Prolog with AllegroGraph 7.0.1* [Na internetu]. Dostupno: https://franz.com/agraph/support/documentation/current/prolog-tutorial.html [pristupano 25.08.2020.]

[42] AllegroGraph Documentation (19.06.2020.) *AllegroGraph 7.0.2 Materialized Reasoner* [Na internetu]. Dostupno: https://franz.com/agraph/support/documentation/current/materializer.html#Rule-Sets [pristupano 25.08.2020.]

Popis slika

Popis tablica