UDC: 004.65 Info M: str. 4-10

MODELOVANJE I IMPLEMENTACIJA RODBINSKIH VEZA KORIŠĆENJEM GRAF MODELA PODATAKA
MODELING AND IMPLEMENTING SERBIAN FAMILY RELATIONSHIPS USING A GRAF DATA MODEL

Svetozar Bajčev – Betware d.o.o,
Slađan Babarogić – Fakultet organizacionih nauka Univerziteta u Beogradu

REZIME: U ovom radu opisane su osnovne osobine grafovskih baza podataka, jedne od kategorija unutar široke oblasti NOSQL baza podataka. Ukratko su navedeni i opisani osnovni koncepti Neo4j sistema za upravljanje grafovskim bazama podataka, kao i njegov graf upitni jezik Cypher. U centralnom delu rada dat je praktičan primer primene grafovske baze podataka u rešavanju problema modelovanja i implementacije rodbinskih veza u porodičnom stablu.

KLJUČNE REČI: grafovske baze podataka, rodbinske veze, Neo4j

ABSTRACT: In this paper we talk about basics of graph databases, one of the categories in large set of NOSQL databases. Elementary concepts of the Neo4j graph database management system have been briefly explained, and also an introduction of the Neo4j's structured graph query language, Cypher, has been made. At the end of this paper, we are presenting one practical example of how to use graph data model, Neo4j and Cypher to model and implement serbian family relationships in fast and easy way.

KEY WORDS: graph databases, family relationships, Neo4j

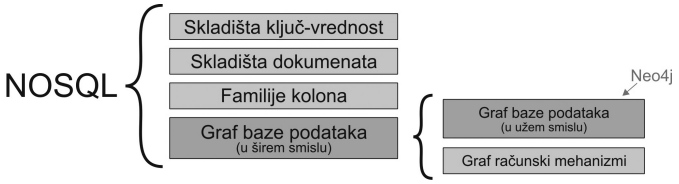
1. UVOD

Nagli porast, kako količine, tako i tempa skladištenja i obrade podataka u poslednjih par decenija, doveo je do pojave i procvata oblasti NOSQL baza podataka. Ovo je mlad pravac u oblasti baza podataka, koji nudi neka nova i drugačija rešenja u odnosu na dobro poznate i široko zastupljene relacione baze podataka. Osnovna ideja ovog pravca jeste da se rešenja pokušaju iznaći u okviru nekih drugih modela podataka, različitih od relacionog modela. Zato se ova oblast može nazvati i **oblast nerelacionih baza podataka**.

Podela NOSQL baza podataka koja je danas najviše prihvaćena obuhvata četiri široke kategorije NOSQL baza podataka. To su:

– Skladišta podataka ključ-vrednost (eng. *Key-value stores*)
– Skladišta dokumenata (eng. *Document stores*)
– Familije kolona (eng. *Column family*)
– Grafovske baze podataka (eng. *Graph databases*)

Svaka od ove četiri navedene kategorije nosi naziv po modelu podataka na kome je zasnovana. Granice među ovim kategorijama nisu strogo definisane i nisu nepoznati slučajevi da za pojedine baze podataka nije uvek lako odrediti kojoj od navedenih kategorija pripadaju.



Slika 1: *Podela NOSQL i grafovskih baza podataka*

2. GRAFOVSKE BAZE PODATAKA

Govoreći o grafovskim bazama podataka, I. Robinson, J. Webber i E. Eifrem u svojoj knjizi *Grafovkse baze podataka*[1] uvode pojam **sveukupnog grafovskog prostora**. Ovim pojmom obuhvaćena su sva tehnološka rešenja u računarstvu i informatici koja funkcionišu na principu grafova, odnosno koja za osnovu imaju neki model podataka oblika grafa i sve operacije koje se obavljaju nad podacima su operacije nad graf modelom podataka. Sveukupni grafovski prostor se može nazvati i **grafovskim bazama podataka u širem smislu**. One se dalje mogu podeliti na dve podkategorije:

1. **Grafovske baze podataka u užem smislu** (eng. *graph databases*) – u koje spadaju tehnologije koje prvenstveno omogućavaju direktno čuvanje podataka u obliku grafa (engl. *online graph persistence*) i kojima se najčešće pristupa iz neke aplikacije, direktno u realnom vremenu. Ekvivalentom ove kategorije u relacionom svetu se mogu smatrati OLTP sistemi,
2. **Grafovski računski mehanizmi** (engl. *graph compute engines*) – u koje spadaju tehnologije koje omogućavaju odloženu (offline) analizu podataka u obliku grafova i obično se izvode u paketima odnosno serijama (engl. *series of bach steps*). Ovo je ista kategorija tehnologija kao i sve druge tehnologije koje se koriste za analizu velike količine podataka (kao što je data mining, ili OLAP).

Grafovske baze podataka u užem smislu su baze podataka u pravom smislu te reči, odnosno sistemi za upravljanje bazama podataka. Pošto se u nastavku fokusiramo na ovu podkategoriju, u daljem tekstu će se pod pojmom *grafovske baze podataka* podrazumevati grafovske baze podataka u užem smislu. Neo4j je jedan od sistema koji pripada ovoj podkategoriji.

Model podataka na kojem su zasnovani ovi sistemi je **graf model podataka**. Opšta definicija grafa je jednostavna: *graf predstavlja kolekciju **temena** i **ivica***. To su ujedno i dva osnovna koncepta u graf modelu podataka. Njihovi nazivi mogu da variraju, pa tako često srećemo i termine **čvorovi** i **veze** (eng. *nodes and relationships*) koje ćemo i ovde koristiti.

1 *Graph Databases*, Ian Robinson, Jim Webber, Emil Eifrem

Ono što graf model podataka odmah na početku izdvaja od drugih modela podataka jeste činjenica da su jedino u ovom modelu **veze između entiteta jednako važne kao i sami entiteti**. U svim drugim modelima podataka, veze između entiteta se ostvaruju posredno, uključivanjem veštačkih elemenata poput spoljnih ključeva kod relacionih baza podataka ili *out-of-band* procesiranja u map-reduce sistemima. Jedino su u graf modelu veze zasebni elementi, isto kao i čvorovi.

Ova osobina graf modela podataka se savršeno poklopila sa danas preovlađujućim trendom da težište u obradi podataka više nije na samim podatacima, već na vezama između tih podataka. Najuticajnije svetske kompanije današnjice, kao što su Google, Facebook, Twitter i druge, svoj poslovni uspeh bazirale su upravo na ovim principima. Umesto da pokuša da pronađe vrednost unutar zasebnih internet stranica, kompanija Google je uočila da je mnogo veća vrednost u vezama koje postoje između tih stranica. Ova kompanija prva je ovladala takozvanim *mrežnim internet grafom* (eng. *web graph*). Na sličan način ljudi iz Facebooka su uočili da, iako postoji vrednost u diskretnim podacima njihovih korisnika (imenima, zanimanjima, stvarima koje vole i dr.), još veća vrednost leži u njihovim međusobnim vezama. Facebook je uspešno ovladao *grafom društvenih odnosa* (engl. *social graph*). Uspevajući da među prvima uoče i ovladaju grafovima, ove kompanije su stekle ključnu konkurentsku predost i izrasle u svetske divove, svaka u svojoj oblasti poslovanja.

Postoji više graf modela podataka, ali su danas u praksi najčešće korišćena sledeća tri:

– grafovi sa atributima ili property grafovi (engl. *property graph*),
– RDF (Resource Description Framework) tripleti i
– hipergrafovi

3. GRAF MODEL PODATAKA

Za naš rad najintersantniji je property graf model podataka. **Property grafovi** (eng. *property* = svojstvo, atribut), su dobili naziv po ključnoj osobini, da njihovi osnovni elementi (čvorovi i veze) sadrže atribute. Svaki čvor ili veza može da ima kolekciju atributa, najčešće oblika složenog zapisa (eng. *record*), a svaki atribut u tom zapisu predstavlja par ključ-vrednost. Ključ je jedinstven na nivou čvora ili veze i jednoznačno određuje vrednost koja mu je dodeljena.

Veze u property grafu mogu biti jednosmerne (usmerene od početnog ka krajnjem čvoru) ili neusmerene. Ukoliko se property graf sastoji samo od usmerenih veza, onda se takav graf zove **usmereni property graf**.

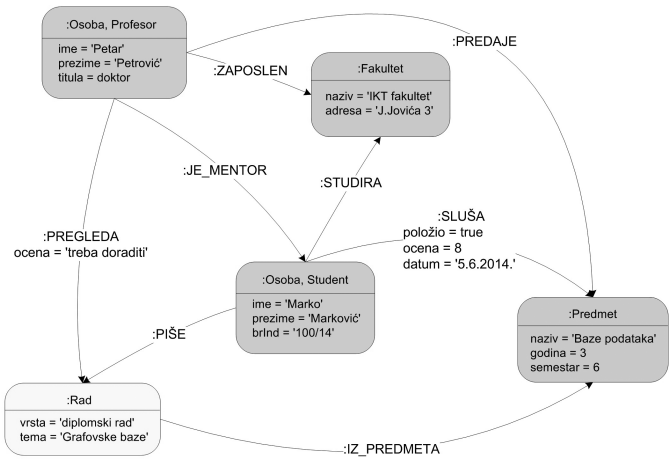
Uvedimo sada pojam labele. **Labele** (eng. *label* = oznaka, etiketa) su oznake koje se dodeljuju čvorovima, i na taj način se prave imenovane grupe čvorova. Svi čvorovi koji imaju istu labelu pripadaju skupu čvorova koji je određen tom labelom.

Imajući sve ovo u vidu, graf model podataka na kojem je zasnovan Neo4j sistem za upravljanje bazama podataka nazivamo **usmereni property graf sa labelama**[2].

Na osnovu svega prethodno navedenog, zaključujemo da su osnovni koncepti modela podataka u Neo4ju:
– čvorovi (eng. *nodes)*
– veze (eng. *relationships*)
– atributi (eng. *properties*)
– labele (eng. *labels*)

Primer graf modela podataka u Neo4j-u dat je na *slici 2*. Model u ovom primeru je prikazan korišćenjem **graf dijagrama**. Na slici se, pored osnovnih koncepata, mogu uočiti i neke vrlo bitne karakteristike graf modela podataka. Naime, graf model podataka predstavlja vrlo zgodan način **vizuelnog prikazivanja** podataka. Graf dijagrami su vrlo **čitljivi i lako razumljivi za običnog čoveka.** Zbog tih svojih osobina grafovi su najčešći način prikazivanja podataka u svakodnevnom životu. Mape, skice, šeme, šabloni, su samo neki od primera graf dijagrama koje koristimo svaki dan.

Još jedna interesantna osobina graf modela podataka, jeste da se ovi modeli radije prikazuju kroz konkretne instance nekih entiteta, nego preko nekih apstraktnih klasa i arhitipova. Graf se najčešće opisuje preko nekog **reprezentativnog primera**, odnosno određenog izdvojenog dela celokupnog grafa, koji na dobar način ilustruje sve osoboine koje su sadržane i u ostatku grafa. Ovakav način predstavljanja grafa se nazva **specifikacija pomoću primera** (eng. *specification by example*).



Slika 2: *Primer Neo4j graf modela podataka*

Navedimo sada i neka pravila koja u Neo4j graf modelu podataka uvek moraju biti ispoštovana:

Najpre o vezama: Veze u Neo4ju uvek imaju **naziv**. Svaka veza može imati jedan i samo jedan naziv i on se po konvenciji piše velikim slovima i kao jedna reč (ukoliko je potrebno, koristi se donja crta "_"). Naziv veze se može smatrati i tipom veze. Veza takođe uvek mora imati **polazni** i **krajnji** čvor, odnosno ne smeju postojati takozvane "landarajuće veze" (eng. *dangling relationships*). Čvor može imati vezu sa samim sobom, tj. dozvoljeno je da jedan isti čvor bude i polazni i krajnji

2 Labele su u modelu podataka Neo4j sistema uvedene od verzije 2.0.

čvor neke veze. Dalje, veze uvek moraju biti usmerene, što za posledicu ima da se za svaki čvor zna koje su njegove veze **izlazne** (one veze kojima je taj čvor polazni čvor), a koje su veze **ulazne** (one veze za koje je taj čvor krajnji čvor).

Atributi: Kao što je već rečeno, i čvorovi i veze mogu imati atribute. Atribut predstavlja par ključ-vrednost, gde je ključ tipa *string*, a vrednost može biti bilo koji predefinisani tip[3] podatka ili niz čiji su elementi predefinisani tipovi. Ključ svakog atributa mora biti jedinstven za dati čvor ili vezu.

Čvorovi: Čvorovi mogu imati 0 ili više veza. Čvor koji ima makar jednu ulaznu ili izlaznu vezu ne sme biti uklonjen iz grafa dok se prethodno ne uklone sve njegove veze. Ovo obezbeđuje poštovanje ograničenja o nepostojanju "landarajućih veza" u grafu.

Labele: Labele se dodeljuju samo čvorovima. Kao što je rečeno, labele grupišu čvorove u imenovane skupove. Jedan čvor može imati 0 ili više labela. Veze ne mogu imati labele.

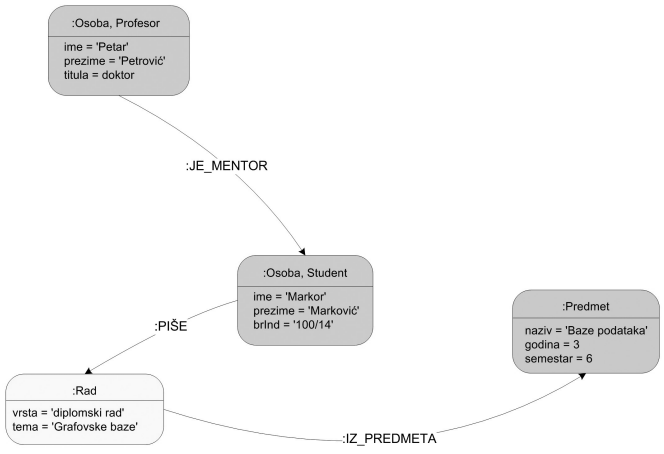
U Neo4ju, postoje još dva koncepta, koja nisu vizuelno uočljiva kao prethodna četiri, ali su podjednako važna. Radi se o sledećim konceptima:

– prolazak kroz graf, odnosno traverziranje (eng. *traverse*)
– putanje (eng. *path*)

Traverziranje ili prolazak kroz graf predstavlja "putovanje" kroz određeni deo grafa na taj način što se polazi od određenog čvora u grafu, prati se neka njegova izlazna veza koja vodi do sledećeg čvora, pa se iz tog čvora prati sledeća veza do sledećeg čvora i tako dalje. Traverziranjem se zapravo „čita" graf ili neki njegov deo.

Putanja predstavlja niz naizmenično poređanih čvorova i veza. Pri tome čvor koji prethodi nekoj vezi u tom nizu je polazni čvor te veze u grafu, a čvor koji sledi toj vezi je njen krajnji čvor u grafu. Putanja je rezultat traverziranja kroz graf.

Prolaskom kroz deo grafa sa *slike 2* mogli bismo pročitati da je: *Prof. Dr Petar Petrović mentor studentu Marku Markoviću i da Marko piše diplomski rad na temu "Grafovske baze" iz predmeta Baze podataka*. Putanja koja bi odgovarala ovom prolasku bi bila:



Slika 3: *Primer putanje u Neo4ju*

3 Neo4j je implementiran u Javi, tako da se pod predefinisanim tipovima podrazumevaju svi predefinisani tipovi Java programskog jezika.

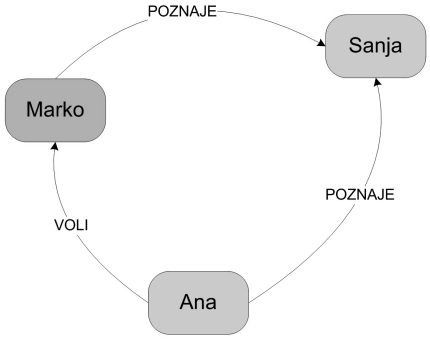
4. CYPHER – GRAF UPITNI JEZIK SISTEMA NEO4J

Jedna od najbitnijih komponenti Neo4j sistema je njegov upitni jezik Cypher. Cypher je deklarativni graf upitni jezik koji se u Neo4ju koristi za rad sa podacima grafovske baze podataka. Cypher je i napravljen sa idejom da bude jednostavan i intuitivan i zbog toga je deo sintakse Cyphera "ukraden" iz nekih popularnih upitnih jezika poput SQL i SPARQL. Tako su se u vokabularu Cyphera našle klauzule poput: `START, MACH, RETURN, WHERE, ORDER BY, DISTINCT` i slične.

Paralelno uz ovu "krađu" reči i konstrukcija iz postojećih upitnih jezika, Cypheru su dodate i neke specifične, originalne osobine i koncepti. Jedna od takvih osobina odnosi se na opise grafovskih konstrukcija u Cypheru. Ovi opisi, koji se u Cypheru nazivaju **paterni**, zapisuju se onako kako bi se i prirodno zapisivali da se koristi običan ljudski jezik. Već smo rekli da ljudi za opisivanje grafova, odnosno njihovog sadržaja i konstrukcija, najčešće koriste crteže, odnosno graf dijagrame.

Primer 1

Pretpostavimo da imamo podatke o tri osobe: Marku, Ani i Sanji, i znamo da Marko poznaje Sanju i da Ana voli Marka i poznaje Sanju. Najlakši način da to predstavimo bio bi graf dijagram na *slici 4*.



Slika 4: *Prikaz podataka pomoću graf dijagrama*

Cypher omogućava korisniku (ili aplikaciji na strani korisnika), da uputi upit bazi za pronalaženje podataka koji se poklapaju sa odgovarajućim paternom zadatim u okviru upita. Prosto rečeno, korisnik pomoću Cyphera kaže bazi: *Pronađi mi sve podatke slične ovom paternu*. Glavni problem je bio kako opisati *ovaj patern*, a koristiti samo karaktere, jer računarski jezik ne poznaje ništa drugo. Graf patern mora, dakle, da se opiše karakterima, a konstatovali smo da je paterne u grafu najprirodnije opisati crtežom!? U ovom trenutku na scenu stupa domišljatost autora Cyphera i fenomenalna, a opet jednostavna i logična ideja, da za "crtanje" paterna u Cypheru iskoriste upravo ono čime raspolažu – same karaktere. Naime, dosetili su se da paterne "iscrtavaju" koristeći *ASCII art*. Termin *ASCII art* je dobro poznat u svetu računarstva i označava umetnost crtanja pomoću karaktera iz ASCII kodnog rasporeda.

Ako graf iz prethodnog primera, proglasimo za patern, onda bismo pomoću ASCII arta u Cypheru mogli ovaj primer opisati na sledeći način:

```
(Ana)-[:VOLI]->(Marko)-[:POZNAJE]->(Sanja),
(Ana)-[:POZNAJE]->(Sanja)
```

Dakle, koristili smo sledeće simbole:

`( )` – za čvor,
`-[]->` – za vezu između dva čvora.

Pri tome se unutar zagrada upisuju određena svojstva čvorova i veza (ime čvora, tip veze, atributi...).

Prethodni ASCII kod je zapravo ***putanja***, i ona povezuje čvor *Ana* sa čvorom *Marko*, čvor *Marko* sa čvorom *Sanja* i čvor *Ana* sa čvorom *Sanja*. Naravno, morali smo se poslužiti malim trikom, da bismo nadomestili činjenicu da upitni jezik ima samo jednu dimenziju (smer čitanja – s leva na desno), dok se graf dijagram, u našem slučaju, prostire u dve dimenzije. Zato smo patern, koji je grafom prikazan kao jedna celina, morali razdvojiti zarezom u dva podpaterna. Ali namera je i dalje ostala ista: paterni u Cypheru prirodno prate paterne onako kako bismo ih iscrtali na tabli ili papiru.

Da bismo što bolje objasnili kako izgledaju upiti u Cypheru, u *primeru 2* dajemo poređenje jednog istog upita u SQLu i u Cypheru.

Primer 2

Pretpostavimo da imamo podatke o pojedinim osobama i mestima stanovanja. Ako bismo te podatke čuvali u obliku relacija, imali bismo sledeće dve tabele:

Osoba

id	ime	mestoID
1	Marko	11000
2	Ana	11000
3	Sanja	21000

Mesto

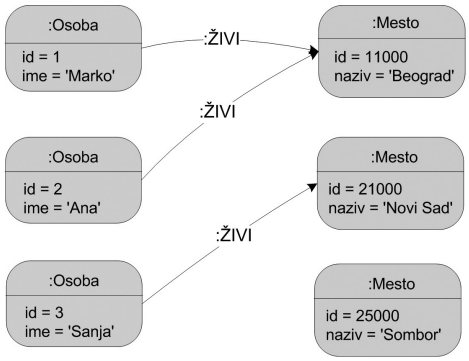
id	naziv
11000	Beograd
21000	Novi Sad
25000	Sombor

Ako iz datih podataka želimo da vidimo ko sve živi u Beogradu, dobro poznati SQL upit bi glasio:

```
SELECT Osoba.*
FROM Mesto
JOIN Osoba ON Mesto.id = Osoba.mestoID
WHERE Mesto.naziv = 'Beograd'
```

Spajanjem dve tabele po spoljnom ključu *mestoID* iz tabele Osoba i primarnom ključu *id* iz tabele Mesto, dobijamo tražene podatke.

Pogledajmo sada isti ovaj primer u slučaju kada su podaci dati u obliku grafa. Imali bismo sledeći graf:



a odgovarajući upit u Cypheru bio bi sledećeg oblika:

```
START mesto=node:Mesto(naziv = 'Beograd')
MATCH (mesto) <-[:ŽIVI]- (osoba)
RETURN osoba
```

U grafu su veze date eksplicitno te nije potrebno nikakvo spajanje, ali da bi se započela pretraga podataka unutar grafa, mora se zadati jedan ili više **polaznih čvorova**. Ovo se obično postiže vezivanjem promenljivih u upitu za pojedine čvorove u grafu, korišćenjem START klauzule. U našem primeru za promenljivu mesto vezani su svi čvorovi u grafu koji imaju labelu Mesto i atribut sa ključem naziv i vrednošću 'Beograd'.

Sledeća stvar koju svaki Cypher upit mora da sadrži jeste **patern** po kojem će se vršiti pretraga. Patern se navodi u okviru MATCH klauzule, na način koji smo ranije opisali. Korišćenjem ASCII art simbola, "crta" se putanja od polaznih čvorova, koja sadrži određene veze i dodatne čvorove. Samo one veze i čvorovi u grafu koje se **poklapaju** (eng. *match*) sa datim paternom biće uvršćeni u rezultat upita. U našem primeru, dat je patern: `(mesto) <-[:ŽIVI]- (osoba)`, kojim opisujemo da želimo da, među našim polaznim čvorovima, pronađemo one koji imaju ulaznu vezu `ŽIVI` i da polazne čvorove za tu vezu vežemo za promenljivu osoba. To će nam omogućiti da pronađemo sve osobe koje žive u Beogradu, te ćemo ih na kraju vratiti kao rezultat upita, korišćenjem klauzule `RETURN`. Ukoliko je potrebno, dodatno filtriranje u Cypher upitima postiže se korišćenjem klauzule `WHERE` (nije korišćena u ovom primeru).

I u slučaju SQL upita, i u slučaju Cypher uputa, kao rezultat dobijamo tabelu:

id	ime	mestoID
1	Marko	11000
2	Ana	11000

Ovakav način kreiranja upita i pretraživanja grafova ima određene prednosti, koje se ogledaju prvenstveno u veoma brzom prolasku kroz graf, te vrlo efikasnom pretraživanju gusto povezanih podataka. Međutim, Cypher poseduje još jednu vrlo bitnu prednost koja se odnosi na rekurzivne upite, vrlo čest i složen problem u relacionim bazama. Cypher ovom problemu

pristupa kroz **paterne proizvoljne dužine**. U okviru Cypher sintakse, postoji specijalni karakter *****, kojim se mogu definisati paterni proizvoljne dužine.

Primer 3

Pretpostavimo da imamo graf osoba između kojih postoje jednosmerne veze `JE_PRIJATELJ` i da, za određenu osobu, želimo da pronađemo prijatelje njenih prijatelja, te prijatelje tih prijatelja i tako u dubinu sve dok postoje prijatelji na sledećem nivou. U Cypheru se ovaj zadatak rešava sledećim jednostavnim upitom:

```
START osoba=node:Osoba('ime: ImeOsobe')
MATCH osoba-[:JE_PRIJATELJ*]->prijatelj
RETURN prijatelj.ime
```

Naravno, pošto često nije praktično nemati kontrolu nad dubinom pretraživanja, Cypher omogućava postavljanje donje i gornje granice. Sledeći upit daće samo prijatelje od drugog do petog nivoa:

```
START osoba=node:Osoba('ime: ImeOsobe')
MATCH osoba-[:JE_PRIJATELJ*2..5]->prijatelj
RETURN prijatelj.ime
```

5. MODELOVANJE I IMPLEMENTACIJA RODBINSKIH VEZA KORIŠĆENJEM GRAF MODELA PODATAKA, NEO4J-A I CYPHER-A

Osim nesumnjivih prednosti koje grafovske baze donose, kako u vidu značajno boljih performansi u pretraživanju i obradi gusto povezanih podataka (eng. *highly connected data*), tako i pri rešavanju specifičnih problema karakterističnih za grafove, ove baze takođe donose i određene novine u pogledu pristupa modelovanju podataka i projektovanju i implementaciji sistema zasnovanim na graf modelu podataka. Inženjeri koji razvijaju Neo4j često ističu prilagođenost Neo4ja početnim fazama projektovanja, odnosno fazama skiciranja modela na tabli za crtanje. Oni tu osobinu nazivaju ***whiteboard friendly.*** Bilo da koristimo UML dijagram klasa, MOV ili neki drugi sličan model Neo4j omogućava da se modeli sistema, koji se obično crtaju na papiru ili tablama za crtanje, vrlo jednostavno prevedu u odgovarajući fizički model, odnosno u konkretnu bazu podataka. Uostalom i UML dijagram klasa i MOV jesu neka vrsta graf modela podataka.

U nameri da i sami isprobamo ovu osobinu, pristupili smo modelovanju rodbinskih veza u porodičnom stablu. Ovaj problem pokazao se kao vrlo zgodan iz dva razloga. Prvi je bila sama struktura porodičnog stabla koja, iako najavljuje strukturu stabla, dodavanjem dodatnih rodbinskih veza između osoba u stablu, zapravo poprima oblik pravog grafa. Druga zgodna osobina je proistekla iz nesumnjivo velokog bogatstva koje srpski jezik ima u pogledu brojnosti rodbinskih veza i njihovih naziva.

Nakon prikupljanja i pobrojavanja većine rodbinskih veza u srpskom jeziku, pristupili smo njihovoj sistematizaciji. Sve rodbinske veze podeljene su na **osnovne** i **izvedene** rodbinske veze. Osnovnih rodbinskih veza bilo je svega šest:

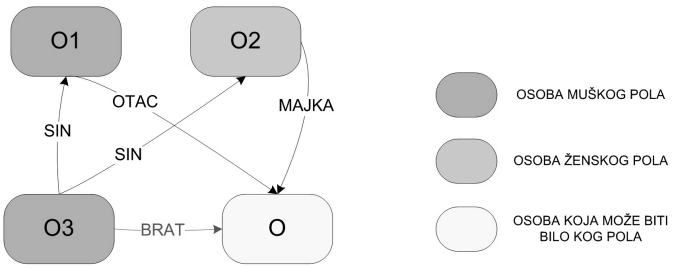
– MAJKA – OTAC
– SIN – KĆERKA
– MUŽ – ŽENA

Sve ostale rodbinske veze, počev od veza nastalih iz krvnog srodstva (poput rodbinskih veza: BRAT, SESTRA, DEDA, BABA, PRABABA, PRADEDA, TETKA, STRIC, UJAK i tako dalje), pa do veza nastalih iz tazbinskog srodstva (poput rodbinskih veza: SVEKAR, SVEKRVA, TAŠTA, TAST, ZET, SNAJA i tako dalje), su proglašene izvedenim rodbinskim vezama.

Ono što je potom uočeno, jeste da u ovakvom sistemu postoje jasna pravila po kojima se svaka izvedena rodbinska veza može objasniti nekom kombinacijom osnovnih rodbinskih veza. Na primer: "BRAT je osoba muškog pola sa kojom imamo zajedničkog OCA i MAJKU" ili "DEDA je OTAC našeg OCA ili OTAC naše MAJKE" i tako dalje. Naravno, što je izvedena veza "dalja", to je potrebno više osnovnih veza da bi se ona opisala, a takođe i njihova kombinacija je složenija.

Kako rodbinske veze predstavljaju jasne usmerene veze u graf modelu podataka, te pošto se one veoma lepo mogu prikazati pomoću graf dijagrama, upotrebili smo graf model podataka kao **sredstvo logičkog zaključivanja**[4], kojim smo uspeli da pomoću grafova definišemo pravila za opis izvedenih rodbinskih veza pomoću osnovnih.

U slučaju izvedene rodbinske veze BRAT, imali smo sledeću situaciju:



Slika 5: *Izvedena rodbinska veza BRAT*

Na *slici 5* je jasno prikazano da je osobi O, ma kog pola ona bila, muška osoba O3 BRAT ako i samo ako osoba O3 ima istog OCA i MAJKU kao i osoba O. Nakon što smo napravili ovakvo pravilo, odnosno graf model, preostalo je samo da ga prevedemo na upitni jezik Neo4ja, Cypher. Odgovarajući upit je glasio:

4 Ideju za korišćenje grafova kao sredstva logičkog zaključivanja dao nam je naučni rad koji je grupa autora iz Francuske objavila pod nazivom: *Knowledge Acquisition with a Pure Graph-Based Knowledge Representation Model*, *Application to the Sisyphus-I Case Study*, LIRMM France, 1999

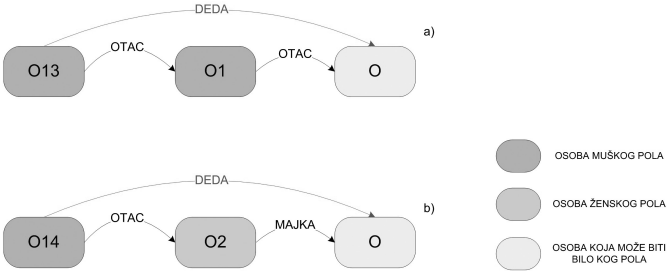
```
MATCH osoba<-[:OTAC]-otac<-[:SIN]-brat,
      osoba<-[:MAJKA]-majka<-[:SIN]-brat
WHERE id(osoba)<>id(brat)
CREATE UNIQUE brat-[:BRAT]->osoba
```

Ako bismo pretpostavili da imamo gotovu grafovsku bazu podataka porodičnog stabla u kojoj postoje sve osnovne rodbinske veze, onda bi izvršavanjem prethodnog upita nad takvom bazom bile kreirane rodbinske veze BRAT između svih osoba u grafu između kojih bi one po definiciji trebalo da postoje.

Sličan postupak smo potom ponovili i za ostale rodbinske veze.

Interesantan slučaj je bio definisanje pravila za izvedene rodbinske veze DEDA, PRADEDA, ČUKUNDEDA i tako dalje u dubinu sve do 14. kolena, do kojeg smo uspeli da pronađemo nazive. Primećeno je da kod kreiranja ovih veza, broj kombinacija osnovnih veza eksponencijalno raste sa povećavanjem kolena, ali da se u svim kombinacijama koriste samo veze OTAC i MAJKA. Zapravo, potrebno je da u prvom koraku paterna, kojim se opisuje odgovarajuće pravilo, postoji rodbinska veza OTAC, dok u je ostalim koracima u paternu svejedno da li je u pitanju OTAC ili MAJKA. Pogledajmo kako je to izgledalo u konkretnim slučajevima:

Za izvedenu rodbinsku vezu DEDA imali smo sledeći graf dijagram:



Slika 6: *Izvedena rodbinska veza DEDA*

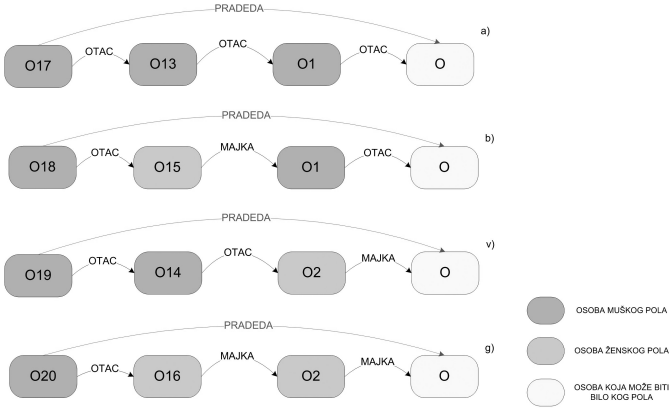
na osnovu kojeg smo kreirali odgovarajući upit u Cypheru:

```
MATCH deda-[:OTAC]->roditelj
      roditelj[:OTAC|MAJKA]->osoba
CREATE UNIQUE deda-[:DEDA]->osoba;
```

Možemo uočiti da je u MATCH paternu u prvom koraku uvek veza OTAC, dok je u drugom koraku svejedno da li je OTAC ili MAJKA.

Graf dijagram izvedene rodbinske veze PRADEDA prikazan je na *slici 7*, a na osnovu njega smo kreirali odgovarajući upit:

```
MATCH pradeda-[:OTAC]->potomak
      potomak-[:OTAC|MAJKA*2]->osoba
CREATE UNIQUE pradeda-[:PRADEDA]->osoba;
```



Slika 7: *Izvedena rodbinska veza PRADEDA*

Kao što možemo uočiti, i ovde, između prvog i drugog čvora u MATCH paternu, mora postojati veza OTAC, dok između ostalih čvorova može biti bilo koja od veza OTAC ili MAJKA. Prethodni upit je u Cypheru moguće i kraće zapisati na sledeći način:

```
MATCH pradeda-[:OTAC]->potomak
      potomak-[:OTAC|MAJKA*2]->osoba
CREATE UNIQUE pradeda-[:PRADEDA]->osoba;
```

gde smo dve veze [:OTAC|MAJKA] u paternu zamenili jednom oznakom [:OTAC|MAJKA*2] i na taj način rekli da nam je bitno samo da između drugog i poslednjeg čvora u paternu imamo 2 veze tipa OTAC ili MAJKA, u bilo kojoj kombinaciji.

Ovo nije urađeno samo iz estetskih razloga. Naprotiv, ovo se pokazalo veoma bitnom osobinom jer je upravo ovakav upit, sa ograničenom dužinom putanje, iskorišćen za uopštavanje kreiranja upita preostalih izvedenih rodbinskih veza, na većim dubinama. Sve što je bilo potrebno jeste zamena broja 2 u prethodnom upitu sa odgovarajućim brojem dubine. Na primer, za sledeću izvedenu rodbinsku vezu ČUKUNDEDA, odgovarajući upit bi glasio:

```
MATCH cukundeda-[:OTAC]->potomak
      potomak-[:OTAC|MAJKA*3]->osoba
CREATE UNIQUE cukundeda-[:PRADEDA]->osoba;
```

i tako dalje. Ista priča se potom ponovila i za izvedene rodbinske veze BABA, PRABABA, ČUKUNBABA..., UNUK, PRAUNUK, ČUKUNUNUK... i UNUKA, PRAUNUKA, ČUKUNUNUKA...

6. ZAKLJUČAK

Oblast grafovskih baza podataka u proteklih nekoliko godina odlikuje veliki rast popularnosti. Grafovi se danas uočavaju u mnogim oblastima života i poslovanja (iako su verovatno sve vreme bili tu). I mnoge svetske kompanije se polako upoznaju sa grafovskim bazama i uočavaju prednosti koje im njihova primena donosi.

Jedna od vodećih svetskih logističkih kompanija koristi garf baze podataka kako bi bila u mogućnosti da u svakom trenutku, u realnom vremenu, proračuna optimalne putanje za isporuku svojih pošiljki. Zatim imamo aviokompaniju koja je svoje multimedijalne metapodatke prebacila u oblik grafa. Tu je i vodeća evropska telekomunikaciona kompanija koja je morala preći na grafovsku bazu podataka, kako bi bila u stanju da u prihvatljivom vremenu odgovori na zahteve velikog broja svojih korisnika. Osim ovih, grafovske baze se od skora pojavljuju i u mnogim drugim oblastima u kojima su do nedavno bile potpuno nepoznate. Tako su se grafovske baze pojavile u oblastima: zdravstvene nege, maloprodaje, nafte i gasa, medija, video igara, igara na sreću i mnogim drugim, a svakim danom broj ovih oblasti se iznova uvećava.

Ovo je pre svega posledica jednog snažnog makroskopskog poslovnog trenda koji danas preovlađuje, a to je potreba da se tržišni prodor i takmičarska prednost u oštrom poslovnom okruženju, ostvare korišćenjem čak i složenih i dinamičnih veza iz gusto povezanih poslovnih podataka. Sposobnost kompanija da razumeju, analiziraju i upotrebe velike grafove gusto povezanih podataka, biće ključ uspeha u sticanju prednosti tih kompanija nad svojim konkurentima u godinama koje dolaze.

7. LITERATURA

[1] I. Robinson, J. Webber, E. Eifrem, *Graph Databases,* O'Reilly, 2013.

[2] J. Baget, D. Genest, M. Mugnier, *Knowledge Acquisition with a Pure Graph-Based Knowledge Representation Model*, *Application to the Sisyphus-I Case Study*, LIRMM France, 1999.

[3] Dokumentacija Neo4j, http://docs.neo4j.org, 2013.

[4] D. McCreary, A. Kelly, *Making Sense of NoSQL*, Manning Publications, 2013.

[5] P. J. Sadalage, M. Fowler, *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*, Addison Wesley, 2012.

[6] Gaurav Vaish, *Getting Started with NoSQL*, Packt Publishing, 2013.

[7] *The Competitive Dynamics of the Consumer Web: Five Graphs Deliver a Sustainable Advantage*, https://www.gartner.com/doc/2081316, Gartner, [preuzeto] jul 2012.

[8] G. Malewicz, M. Austern, A. Bik, J. Dehnert, I. Horn, N. Leiser, G. Czajkowski, *Pregel: A System for Large-Scale Graph Processing*, SIGMOD '10, 2010.

[9] Enciklopedija Wikipedia, http://sr.wikipedia.org/wiki/Српски_сроднички_односи, [preuzeto] januar 2014.

Svetozar Bajčev – Betware d.o.o
Kontakt: netcrush@gmail.com
Oblast interesovananja: Grafovske baze podataka, Java, Poslovni informacioni sistemi, Softversko inženjerstvo

Slađan Babarogić – docent, Fakultet organizacionih nauka Univerziteta u Beogradu
Kontakt: babarogic.sladjan@fon.bg.ac.rs
Oblast interesovananja: Razvoj IS vođen modelima, Poslovna analiza i Modelovanje poslovnih procesa, Metodologije razvoja informacionih sistema i Baze podataka

infoM

UPUTSTVO ZA PRIPREMU RADA

1. Tekst pripremiti kao Word dokument, A4, u kodnom rasporedu 1250 latinica ili 1251 ćirilica, na srpskom jeziku, bez slika. Preporučen obim – oko 10 strana, single prored, font 11.

2. Naslov, abstakt (100-250 reči) i ključne reči (3-10) dati na srpskom i engleskom jeziku.

3. Jedino formatiranje teksta je normal, bold, italic i bolditalic, VELIKA i mala slova (tekst se naknadno prelama).

4. Mesta gde treba ubaciti slike, naglasiti u tekstu (Slika1...)

5. Slike pripremiti odvojeno, VAN teksta, imenovati ih kao u tekstu, radi identifikacije, u sledećim formatima: rasterske slike: jpg, tif, psd, u rezoluciji 300 dpi 1:1 (fotografije, ekranski prikazi i sl.), vektorske slike – cdr, ai, fh,eps (šeme i grafikoni).

6. Autor(i) treba da obavezno priloži svoju fotografiju (jpg oko 50 Kb), navede instituciju u kojoj radi, kontakt i 2-4 oblasti kojima se bavi.

7. Maksimalni broj autora po jednom radu je 5.

Redakcija časopisa Info M