Sveučilište u Rijeci
University of Rijeka
http://www.uniri.hr

Polytechnica: Journal of Technology Education, Volume 4, Number 2 (2020)
Politehnika: Časopis za tehnički odgoj i obrazovanje, Volumen 4, Broj 2 (2020)

Politehnika
Polytechnica
http://www.politehnika.uniri.hr
cte@uniri.hr

DOI: https://doi.org/10.36978/cte.4.2.5

Pregledni članak
Review article
UDK 004.652

Istraživanja u području grafovskih baza podataka

Kornelije Rabuzin, Robert Kudelić
Fakultet organizacije i informatike
Sveučilište u Zagrebu
Pavlinska 2, 42000 Varaždin
kornelije.rabuzin@foi.unizg.hr, robert.kudelic@foi.unizg.hr

Sažetak

Kako smo unatrag nekoliko godina dobili određena sredstva (tzv. sveučilišne potpore) kojima smo istraživali određene aspekte grafovskih baza podataka, odlučili smo rezultate istraživanja predstaviti i hrvatskoj akademskoj i stručnoj zajednici. Smatramo kako su istraživanja bila kvalitetna i da rezultati istih imaju potencijal za širu primjenu i implementaciju u dostupnim sustavima za upravljanje grafovskim bazama podataka. Također, postoje i određene mogućnosti za proširenje implementiranih rješenja, kao i prostor za dodatna istraživanja i preslikavanja relacijskih u koncepte grafovskih baza podataka.

Ključne riječi: *grafovske baze podataka; Neo4j; integritetna ograničenja; nasljeđivanje; Cypher; Gremlin.*

1. Uvod

Teorija grafova područje je djelovanja u kojemu se putem strukture grafa modelira odnos objekata. Kao takva, prožima se u mnoga područja znanosti, od računalnih znanosti i društvenih znanosti, pa sve do lingvistike i kemije (Wilson, 1996). Spektar je područja gdje teorija grafova ima svoje mjesto zaista zapanjujući, pa je tako, možda i ne iznenađujuće, teorija grafova stupila svoju nogu i u baze podataka. Autori su objavili neke radove i iz područja teorije grafova, primjerice (Kudelić, 2016), te (Kudelić & Ivković, 2019).

Velike količine podataka koje se svakodnevno generiraju nadilaze mogućnosti klasičnih, relacijskih sustava za upravljanje bazama podataka. Količina podataka koja se generira u dva dana odgovara količini podataka koju smo prije nešto više od desetljeća generirali u toku jedne cijele godine. Uz to, podaci su heterogeni (tekst, slika, video zapis, itd.) te je brzina generiranja velika. U literaturi se stoga spominje skraćenica 3V (eng. Volume, Velocity, Variety) koja upravo opisuje spomenute probleme vezane uz velike količine podataka. Kako „veliki podaci“ predstavljaju određene izazove u smislu pohrane i procesiranja istih, javila se potreba i za novim rješenjima. Jedno od rješenja su i NoSQL baze podataka.

Kad govorimo o NoSQL bazama podataka, razlikujemo sljedeće vrste baza: stupčaste, grafovske, ključ-vrijednost i dokumentne. Svaki od spomenutih tipova počiva na svom modelu, no ono što je karakteristika svim spomenutim tipovima je da ne koriste SQL kao standardizirani upitni jezik. Iako su i ostale vrste zasigurno interesantne, ovaj rad primarno se bavi grafovskim bazama podataka pa će o njima biti više riječi u nastavku. Također, samostalno smo istražili i implementirali određena proširenja o kojima će također nešto više riječi biti u nastavku.

Grafovske baze podataka postale su popularne u posljednjih desetak godina. Niz velikih poznatih poslovnih sustava koristi prednosti grafovskih baza podataka, a domene primjene su raznolike: otkrivanje

prijevara, sustavi preporuke, sastavnica proizvoda, kontrola pristupa, itd. Nije više dovoljno prikupljati podatke, važno je otkriti i veze među njima.

Što se samog termina tiče, „graph databases“, kad smo započeli s radom na ovoj temi postavilo se pitanje koji prijevod na hrvatski je ispravan. Ako bi termin preveli kao „graf baze podataka“, to naime nije dobro jer netko može misliti da se radi o grafu koji reprezentira neku bazu podataka (slično je i kod pojma „graf funkcije“ gdje imamo graf za određenu funkciju). Kontaktirali smo i institut za jezikoslovlje u Zagrebu, pravilna tvorba i prijevod na hrvatski je „grafovska baza podataka“, pa ćemo stoga taj termin koristiti u nastavku.

Grafovske baze podataka počivaju na idejama koje su poznate u teoriji grafova. Graf se sastoji od čvorova i bridova između čvorova. Analogno, grafovska baza podataka služi pohrani podataka i informacija, s time da podatke pohranjujemo u čvorove i veze između čvorova. Čvorovi imaju svoja svojstva, te ćemo na primjerima kasnije demonstrirati kako se čvorovi i svojstva mogu implementirati i kako se kasnije mogu postavljati upiti. U tu ćemo svrhu koristiti sustav Neo4j koji je trenutno najpopularniji sustav za upravljanje grafovskim bazama podataka.

Grafovski sustavi u počecima nisu podržavali napredne karakteristike kakve su primjerice podržane kod relacijskih sustava (napredna ograničenja, nasljeđivanje, okidače…), no stvari polako idu na bolje. Kako smo rano identificirali nedostatke, upravo iz tog razloga smo samostalno implementirali određena proširenja o kojima ćemo govoriti u ostatku rada. Radova u kojima se također uvode proširenja ili se istražuju grafovske baze ima podosta: (Kaliyar, 2015), (Thompson, Personick, & Cutcher, 2014), (Bayerl & Granitzer, 2017), (Reutter, Romero, & Vardi, 2017), (Robinson, Webber, & Eifrem, 2013), (Jeon, Khosiawan, & Hong, 2013), (Chen, Song, Zhao, & Li, 2020), (de Oliveira, de Souza, Moreira, & Seraphim, 2020), itd.

Za početak će biti govora o implementaciji integritetnih ograničenja u relacijskim i grafovskim sustavima. I dok su kod relacijskih sustava integritetna ograničenja dobro istražena i lako se mogu implementirati, u grafovskim bazama podataka podrška za iste u početku je izostala. Stoga smo sami prionuli implementaciji te smo uspješno implementirali integritetna ograničenja UNIQUE i CHECK.

Što se jezika za grafovske baze podataka tiče, za rad s najpopularnijim sustavom Neo4j možemo koristiti jezike Cypher i/ili Gremlin. Oni imaju svoju sintaksu, no možemo reći kako u pojedinim klauzulama Cypher sliči SQL-u koji je standardni jezik za rad s relacijskim bazama podataka. No kako razlike u jezicima postoje, prionuli smo implementaciji vizualnog sučelja koje bi trebalo omogućiti korištenje grafovske baze podataka na način gdje bi se koristila ideja Query By Example (QBE; upit kroz primjer) jezika koji je primjerice podržan u sustavu MS Access. QBE omogućava vizualno postavljanje upita prema bazi podataka, bez poznavanja (naprednih) klauzula jezika SQL. Na istim principima implementirali smo sučelje Gremlin By Example koje omogućava rad s bazom podataka na način gdje se upiti u Gremlinu specificiraju vizualno, bez poznavanja pojedinih klauzula samog jezika.

Kod objektno-relacijskih sustava od velike važnosti je koncept nasljeđivanja, tj. mogućnost da tablica naslijedi atribute (i ograničenja) neke druge tablice. Implementirano je proširenje i mogućnost da čvor grafovske baze naslijedi neki drugi, već ranije kreirani čvor.

Kod relacijskih baza podataka od velike su važnosti i okidači (eng. trigger). Temeljem iskustva dobivenog u implementaciji jezika Gremlin By Example, napravljeno je i određeno proširenje koje omogućava vizualnu specifikaciju okidača u grafovskoj bazi.

Za sve gore navedeno trebalo je riješiti i određene tehničke izazove i pronaći kako spomenuta proširenja implementirati. To je bio izazov koji smo također uspješno riješili.

U nastavku ovog rada slijedi kratak pregled korištenih sustava za upravljanje grafovskim bazama podataka i mali primjer kreiranja grafovske baze podataka u sustavu Neo4j. Nakon toga pozabavit ćemo se implementacijom integritetnih ograničenja i jezicima koji se koriste za rad s grafovskim bazama. Tu ćemo predstaviti Gremlin By Example i mogućnost nasljeđivanja, a nakon tog slijedi opis proširenja koje omogućuje vizualnu specifikaciju okidača u samoj bazi. Na kraju cijelog rada slijedi zaključak.

Za istraživanje smo koristili financijska sredstva koja smo dobili u sklopu sveučilišne potpore „Novi jezik za grafovske baze podataka“ u trajanju od 2015. - 2017. godine.

2. Sustavi za upravljanje grafovskim bazama podataka

Što se sustava tiče, najpopularniji sustav za upravljanje grafovskim bazama podataka je Neo4j (Slika 1).

Rank Sep 2020	Rank Aug 2020	Rank Sep 2019	DBMS	Database Model
1.	1.	1.	Neo4j	Graph
2.	2.	2.	Microsoft Azure Cosmos DB	Multi-model
3.	3.	↑ 4.	ArangoDB	Multi-model
4.	4.	↓ 3.	OrientDB	Multi-model
5.	5.	5.	Virtuoso	Multi-model
6.	6.	↑ 7.	Amazon Neptune	Multi-model
7.	7.	↓ 6.	JanusGraph	Graph
8.	8.	↑ 17.	FaunaDB	Multi-model
9.	9.	↑ 10.	Dgraph	Graph
10.	↑ 11.	↑ 13.	Stardog	Multi-model
11.	↓ 10.	↓ 8.	GraphDB	Multi-model
12.	12.	12.	TigerGraph	Graph
13.	13.	↓ 9.	Giraph	Graph

Slika 1. Dostupni sustavi (https://db-engines.com/en/ranking/graph+dbms)

Što se same pohrane podataka u grafovskim bazama podataka tiče, generalno možemo reći kako razlikujemo sustave koji su implementirani na način da grafove spremaju kao grafove, te sustave koji koriste relacijski ili neki drugi model za pohranu grafovskih podataka (eng. native vs. non-native). Možemo reći kako su primarno grafovski sustavi izgubili na popularnosti u zadnjih nekoliko godina. Naime, postoje klasični relacijski sustavi koji danas ne podržavaju samo relacijski model podataka, već podržavaju više modela, uključujući i grafovski. Kao primjer svakako valja izdvojiti MS SQL Server i Oracle, dva iznimno popularna, nekad primarno relacijska sustava.

Kad govorimo o relacijskim sustavima, shema tablice je fiksna i svaki redak koji dodajemo u tablicu trebao bi imati specificirane vrijednosti za sve stupce tablice. Iznimno, ako vrijednost nije poznata, može se upisati NULL. Kako ćemo kasnije vidjeti, kod grafovskih baza podataka to nije slučaj. Upravo to je jedna od bitnih odrednica cijelog NoSQL pokreta koji se javio kao alternativa relacijskim bazama podataka.

3. Neo4j i Cypher

Grafovska baza podataka sastoji se od čvorova i veza između čvorova. Svaki čvor, a i veza, mogu imati drugačija svojstva, odnosno shemu. Za kreiranje čvorova i veza u sustavu Neo4j koristi se jezik Cypher i naredba CREATE.

Kreirajmo dva čvora odnosno dvije osobe u bazi; prvi se čvor zove Petar te ima oznaku (labelu) Osoba (istu koristimo za grupiranje odnosno klasificiranje čvorova). Nakon toga imamo svojstvo *ime* čija vrijednost je *Petar*:

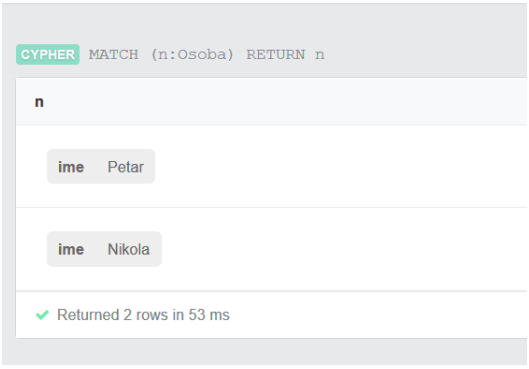
```
CREATE (Petar:Osoba { ime: "Petar" })
```

Kreirajmo i drugu osobu odnosno čvor u bazi:

```
CREATE(Nikola:Osoba { ime: "Nikola" })
```

Provjerimo jesu li čvorovi stvarno kreirani. MATCH klauzulu koristimo za definiranje uzorka koji se traži u bazi i ona često ide u paru s klauzulom WHERE u kojoj možemo definirati i dodatne uvjete koji moraju biti zadovoljeni. U ovom upitu ne koristimo klauzulu WHERE, no u klauzuli MATCH koristimo labelu Osoba kako bi upit vratio samo čvorove s tom labelom (Slika 2):

```
MATCH (n:Osoba)
RETURN n
```



Slika 2. Osobe u bazi

Ako ne specificiramo labelu (Osoba), upit bi vratio sve čvorove iz baze podataka:

```
MATCH n
RETURN n
```

Kreirajmo vezu između dva čvora kojom ćemo označiti od kada su Petar i Nikola prijatelji. MATCH klauzulu koristimo kako bi se referencirali na kreirane čvorove u bazi, a između kojih onda kreiramo i vezu (CREATE):

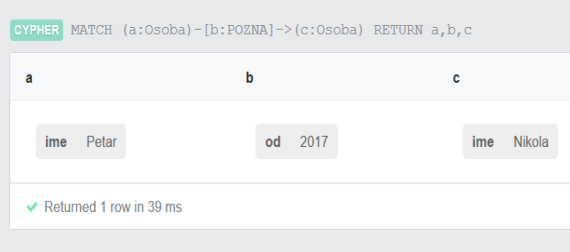
```
MATCH (Petar:Osoba {ime:"Petar"}), (Nikola:Osoba {ime:"Nikola"})

CREATE (Petar)-[:POZNA {od:2017}]->(Nikola)
```

Kako bismo provjerili sadržaj baze, koristimo sljedeći upit; u osnovi, tražimo čvorove koji su povezani putem veze 'pozna':

```
MATCH (a:Osoba)-[b:POZNA]->(c:Osoba)
RETURN a, b, c
```

Vidimo kako upit vraća podatke o čvorovima te godinu od kada se Petar i Nikola međusobno poznaju:



Slika 3. Osobe i poznanstva

Moguće je da čvorovi kao i kreirane veze imaju različita svojstva, a isto vrijedi i za veze.

Ovaj primjer je samo poslužio kako bismo stekli dojam kako se radi sa sustavom Neo4j i jezikom Cypher. Naredba ima puno i njima se ne možemo detaljnije baviti, no svakako upućujemo pogledati dokumentaciju sustava.

4. Integritetna ograničenja

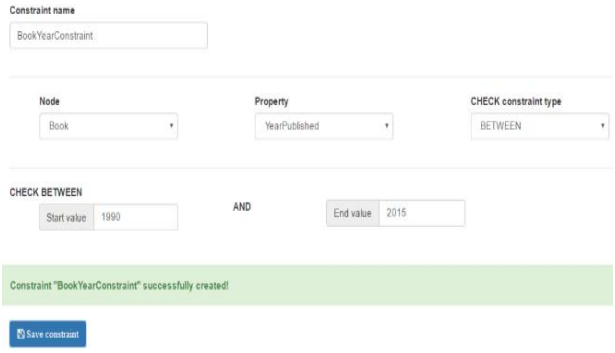
Integritetna ograničenja koristimo kako bi osigurali kvalitetu podataka u samoj bazi i kako bi izbjegli upisivanje nekonzistentnih i potencijalno nekorektnih vrijednosti u bazu podataka. Kod relacijskih baza podataka tu svakako valja istaknuti:

- PRIMARY KEY – vrijednost stupca mora biti jedinstvena i mora biti upisana,
- NOT NULL – vrijednost stupca mora biti upisana,
- UNIQUE – vrijednost stupca mora biti jedinstvena,
- CHECK – vrijednost stupca mora odgovarati definiranom uvjetu,
- FOREIGN KEY – vrijednost stupca mora odgovarati vrijednosti stupca u drugoj (moguće istoj) tablici na koju se referenciramo.

Ako su vrijednosti koje se upišu u samu bazu podataka „loše“, odnosno nisu konzistentne a niti korektne, tada puno vremena moramo potrošiti kako bi se riješili problemi s takvim podacima, što je pogotovo problem kod izrade sustava poslovne inteligencije.

Kod grafovskih baza podataka prije svega nekoliko godina, spomenuta ograničenja nisu bili podržana. Tako smo samostalno razvili specifikaciju prema kojoj vrijednost atributa mora biti upisana, specifikaciju prema kojoj vrijednost atributa mora biti jedinstvena, te specifikaciju prema kojoj vrijednost atributa mora zadovoljavati određeni uvjet.

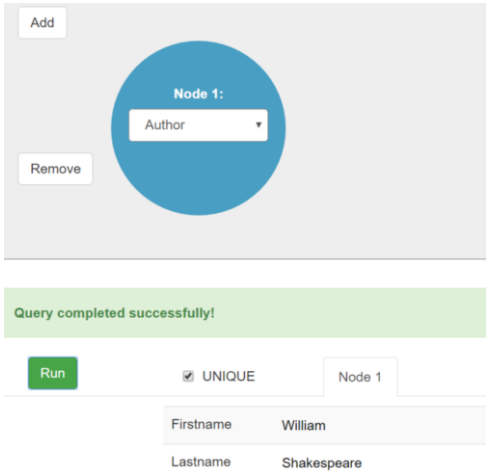
Sljedeća slika prikazuje specifikaciju ograničenja CHECK u samoj bazi. Prilikom unosa, vrijednost atributa (svojstva) *YearPublished* za čvor *Book* mora biti između 1990 i 2015:



Slika 4. Implementacija ograničenja CHECK

Ako vrijednost atributa nije specificirana kako je to definirano ograničenjem, neće se moći upisati u bazu podataka. Detalji implementacije dani su u (Rabuzin, Konecki, & Šestak, 2016).

Nadalje, realizirali smo i implementaciju ograničenja UNIQUE:



Slika 5. Implementacija ograničenja UNIQUE

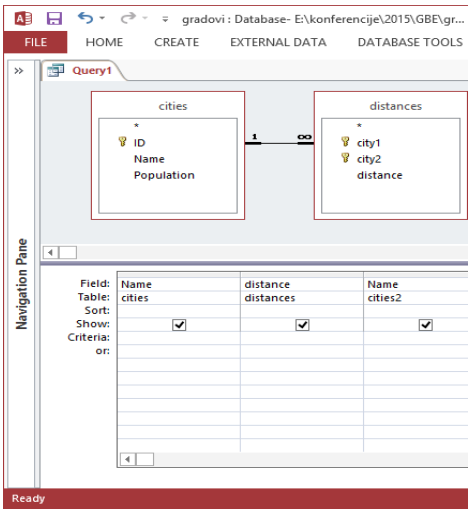
Na slici vidimo kako je dodan novi autor, William Shakespeare, a vidimo i kako je kod dodavanja čvora uključena opcija UNIQUE. Ponovni pokušaj dodavanja istog autora (William Shakespeare) završava neuspjehom. Detalji implementacije dani su u (Rabuzin, Šestak, & Konecki, 2016).

5. Jezici za rad s grafovskim sustavima

Kad govorimo o relacijskim bazama podataka, primarno se koristi jezik SQL. Za grafovske baze podataka popularni su Cypher i Gremlin, no sustavi kao što su Oracle i MS SQL Server koriste verzije koje više sliče jeziku SQL.

Iako je SQL standardizirani jezik za rad s bazama podataka, on svakako nije jedini. U sustavu MS Access moguće je koristiti i Query By Example, koji

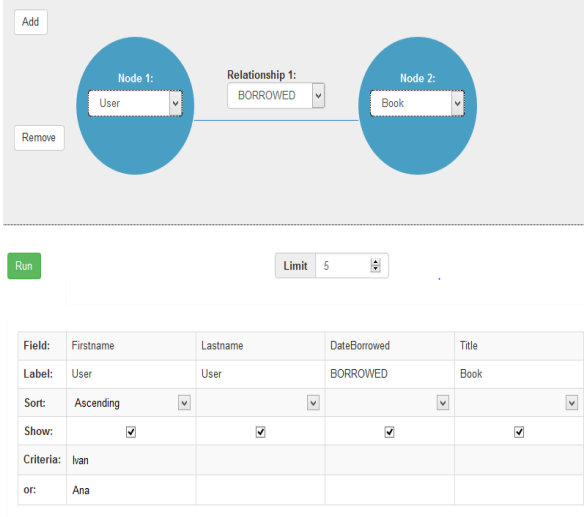
omogućava vizualno postavljanje upita bez poznavanja SQL klauzula (Slika 6).



Slika 6. QBE u sustavu MS Access

U gornjem dijelu slike 6 specificiramo tablice iz kojih želimo dohvatiti podatke, a u donjem dijelu popunjavamo koje atribute iz odabranih tablica želimo u odgovoru, koji uvjeti pri tome moraju biti zadovoljeni, te kako će se rezultat sortirati.

U nastavku ćemo demonstrirati sučelje koje smo nazvali Gremlin By Example (Gremlin kroz primjer). Domena su knjige i posudbe knjiga. Pri dizajniranju rješenja iskorištena je ideja jezika Query By Example, a razvijeno sučelje može se koristiti prilikom rada sa sustavom Neo4j:



Slika 7. Gremlin By Example (Gremlin kroz primjer)

Na Slici 7 (gornji dio) vidi se da smo dodali dva čvora, *User* i *Book*, a dodatno smo specificirali i vezu između njih, *BORROWED*. U donjem je dijelu slike moguće specificirati uvjete koji moraju biti zadovoljeni (tražimo istinitost izraza '*Firstname = Ivan*' ili '*Firstname = Ana*'). Moguće je specificirati kako će se rezultat sortirati i što će sve (koji stupci) biti prikazano u samom rezultatu.

Ovaj način zadavanja upita daleko je jednostavniji od pisanja upita u samom Gremlinu, a i razvijeno sučelje sliči QBE jeziku (Slika 6), dakle jeziku s kojim su korisnici upoznati od ranije, a što zasigurno olakšava rad s grafovskim bazama podataka.

6. Nasljeđivanje

Nasljeđivanje je, osim u programiranju, steklo popularnost i u bazama podataka. U programiranju je to način rada koji nam omogućuje temeljenje jednog objekta na drugom pri čemu implementacija ostaje slična. Putem hijerarhije objekata programer može iskoristiti prijašnji kod, napraviti novu implementaciju s istom funkcionalnošću, ali i proširiti prijašnji program novom funkcionalnošću.

U svijetu baza podataka objektno-relacijski pokret preuzeo je ideju nasljeđivanja i moguće je kreirati tablice koje nasljeđuju neke druge tablice. Pogledajmo sljedeći primjer:

```
CREATE TABLE grad
(naziv VARCHAR (100),
broj_stanovnika BIGINT)

CREATE TABLE glavni_grad
(drzava VARCHAR(100))
INHERITS (grad)
```

Vidimo kako druga tablica *glavni_grad* nasljeđuje tablicu *grad* (INHERITS grad), te je u tablici *glavni_grad* definiran novi atribut *drzava*, koji sadrži naziv države kojoj je taj grad glavni. Dakle kako svi gradovi nisu glavni, glavni gradovi dodatno imaju i atribut koji predstavlja državu kojoj je taj grad glavni. Ako postavimo upit

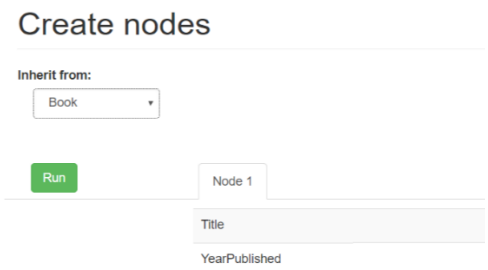
```
SELECT * FROM glavni_grad
```

u odgovoru ćemo dobiti sva tri atributa: *naziv*, *broj_stanovnika* i *drzava*.

Prilikom nasljeđivanja nasljeđuju se atributi tablice, no postoje i dodatne opcije, ali one izlaze izvan opsega ovoga rada pa se tako njima u ovom radu nećemo detaljnije baviti.

Na sličan način omogućili smo i specifikaciju nasljeđivanja u grafovskim bazama podataka; čvor koji se stvara može naslijediti svojstva odnosno atribute nekog već ranije stvorenog čvora. Na slici 8 u nastavku možemo vidjeti kako se kreira novi čvor, s time što isti

nasljeđuje svojstva već ranije kreiranog čvora *Book*. Novi čvor dakle ima dva dodatna svojstva, *Title* i *Year Published*.



Slika 8. Nasljeđivanje u grafovskim bazama podataka

Za detalje same implementacije upućujemo na (Rabuzin & Sestak, 2018).

7. Okidači

I dok je teorija aktivnih baza podataka dobro poznata i razvijena za relacijske sustave, to se nije moglo reći za grafovske. Osnovu aktivnih baza podataka čini mehanizam automatskog reagiranja; kada nastupi neki događaj, automatski se mogu izvršiti određene akcije u ili pak van same baze podataka kao reakcija na nastali događaj. Taj mehanizam može poslužiti kako bi pratili promjene u samoj bazi, kako bi održavali kopije podataka, automatski naručivali određene proizvode, kod različitih kontrolnih sustava, itd.

Sukladno tome, razvili smo prototip za vizualno kreiranje okidača u samoj grafovskoj bazi podataka (Slika 9).

Slika 9. Specifikacija okidača u grafovskoj bazi

Na primjeru se može vidjeti kako nije moguće dodati novi čvor Job Posting ako već ne postoji čvor Agency sa svojstvom name = NYC HOUSING AUTHORITY. Više informacija o implementaciji može se pronaći u (Rabuzin & Šestak, 2019).

8. Daljnja istraživanja

Na početku svakako valja istaknuti kako za prikazana rješenja sigurno ima mjesta za daljnja poboljšanja i manje korekcije, a isto vrijedi i za poboljšanja oko performansi.

Što se daljnjih istraživanja tiče, zasigurno ćemo se pozabaviti preslikavanjem i implementiranjem ostalih mehanizama koji postoje kod relacijskih baza podataka, a smisleno ih je prilagoditi i za grafovske baze podataka.

Uputno bi bilo dodatno istražiti i značaj grafovskih baza podataka kod izgradnje sustava poslovne inteligencije (eng. Business Intelligence). Naime, u skladištima podataka koja čine osnovu sustava za poslovnu inteligenciju, u principu su smještene velike količine povezanih podataka i to u obliku dimenzijskih i činjeničnih tablica. Grafovske baze podataka imaju izuzetno dobre performanse kod analize podataka koji su međusobno povezani, a što je u skladištima sa podacima svakako slučaj.

Rudarenje podacima nad grafovskim bazama podataka (eng. graph mining) također je još jedna zanimljiva tema koju bi imalo smisla dodatno istražiti.

9. Zaključak

Grafovske baze podataka u osnovi počivaju na temeljnim principima teorije grafova. Kako su u teoriji grafova osnovni koncepti čvor i veza, tako i grafovske baze podataka podatke pohranjuju u čvorovima i vezama koje iste povezuju. Ideja kao takva se pokazala iznimno korisnom, a i niz velikih poslovnih sustava koristi grafovske baze podataka za različite namjene.

Unazad zadnjih nekoliko godina uspješno smo istražili određene aspekte grafovskih baza podataka, te smo napravili određena proširenja sustava Neo4j. Učinjena su proširenja u specifikaciji integritetnih ograničenja, u definiranju okidača, u definiranju novih sučelja odnosno grafičkih verzija upitnih jezika (Gremlin By Example), u mogućnostima nasljeđivanja, itd. Također, identificirana su i neka otvorena pitanja kojima ćemo se pozabaviti u budućim istraživanjima.

Istraživanje opisano u ovom radu uključilo je i rješavanje određenih tehničkih izazova oko prilagodbe i povezivanja različitih razvojnih okruženja, kako bi implementacija ovih proširenja uopće bila moguća.

Za potrebe istraživanje koristili smo financijska sredstva koja smo dobili u sklopu sveučilišne potpore

„Novi jezik za grafovske baze podataka“ u trajanju od 2015. - 2017. godine. S obzirom na raspoloživa sredstva, smatramo da smo doprinijeli razvoju grafovskih sustava i baza podataka, te da će naše ideje biti od pomoći svima koji se bave spomenutim područjem. Ovaj rad daje uvid u rezultate naših istraživanja, te smatramo da može poslužiti hrvatskoj akademskoj i/ili stručnoj zajednici.

Literatura

Bayerl, S., & Granitzer, M. (2017). Discovering, Ranking and Merging RDF Data Cubes. In *Proceedings - IEEE 11th International Conference on Semantic Computing, ICSC 2017*. https://doi.org/10.1109/ICSC.2017.56

Chen, J., Song, Q., Zhao, C., & Li, Z. (2020). Graph Database and Relational Database Performance Comparison on a Transportation Network. In *Communications in Computer and Information Science* (Vol. 1244 CCIS, 407–418). Springer. https://doi.org/10.1007/978-981-15-6634-9_37

de Oliveira, A. T., de Souza, A. D., Moreira, E. M., & Seraphim, E. (2020). Mapping and Conversion between Relational and Graph Databases Models: A Systematic Literature Review. In *Advances in Intelligent Systems and Computing* (Vol. 1134, 539–543). Springer. https://doi.org/10.1007/978-3-030-43020-7_71

Jeon, S., Khosiawan, Y., & Hong, B. (2013). Making a Graph Database from Unstructured Text. In Chen, J and Cuzzocrea, A and Yang, LT (Ed.), *2013 IEEE 16th Iternational Conference on Computational science and Engineering (CSE 2013)* (981–988). https://doi.org/10.1109/CSE.2013.144

Kaliyar, R. K. (2015). Graph Databases: A Survey. In Swaroop, A and Sharma, V (Ed.), International Conference on Computing, Communication & Automation *(ICCCA)* (785–790).

Kudelić, R. (2016). Monte-Carlo randomized algorithm for minimal feedback arc set. *Applied Soft Computing Journal*, *41*, 235–246. https://doi.org/10.1016/j.asoc.2015.12.018

Kudelić, R., & Ivković, N. (2019). Ant inspired Monte Carlo algorithm for minimum feedback arc set. *Expert Systems with Applications*, *122*, 108–117. https://doi.org/10.1016/j.eswa.2018.12.021

Maleković, M., Rabuzin, K., & Šestak, M. (2016). Graph Databases-are they really so new. *International Journal of Advances In Science Engineering and Technology*. Retrieved from http://bib.irb.hr/prikazi-rad?rad=843723

Rabuzin, K, Konecki, M., & Šestak, M. (2016). Implementing CHECK Integrity Constraint in Graph Databases. Proceedings of the 82nd IIER International Conference. Retrieved from http://bib.irb.hr/prikazi-rad?rad=836861

Rabuzin, K, Maleković, M., & Šestak, M. (2016). Gremlin By Example. *International Conference on Advances in Big Data Analytics*, 144–149.

Rabuzin, Kornelije, Konecki, M., & Šestak, M. (2016). Implementing CHECK Integrity Constraint in Graph Databases. In *IIER 105th International Conference on Recent Innovations in Engineering and Technology* (19–22). Berlin, Germany.

Rabuzin, Kornelije, & Sestak, M. (2018). Towards inheritance in graph databases. In *2018 International Conference on Information Management and Processing, ICIMP 2018* (Vol. 2018, 115–119). https://doi.org/10.1109/ICIMP1.2018.8325851

Rabuzin, Kornelije, & Šestak, M. (2019). Creating triggers with trigger-by-example in graph databases. In *DATA 2019 - Proceedings of the 8th International Conference on Data Science, Technology and Applications* (137–144). https://doi.org/10.5220/0007829601370144

Rabuzin, Kornelije, Šestak, M., & Konecki, M. (2016). Implementing UNIQUE Integrity Constraint in Graph Databases. In *The Eleventh International Multi-Conference on Computing in the Global Information Technology* (48–53).

Reutter, J. L., Romero, M., & Vardi, M. Y. (2017). Regular Queries on Graph Databases. Theory of computing Systems *61*(1), 31–83. https://doi.org/10.1007/s00224-016-9676-2

Robinson, I., Webber, J., & Eifrem, E. (2013). *Graph Databases*. *Information Management*. https://doi.org/http://dx.doi.org/10.1016/B978-0-12-407192-6.00003-0

Šestak, M., Rabuzin, K., & Konecki, M. (2016). Indexing in relational and graph databases as a mean of dealing with increasing amount of business data. *International Conference on Management, Business and Marketing,* 181-185.

Sestak, M., Rabuzin, K., & Novak, M. (2016). Integrity constraints in graph databases-implementation

challenges. Proceedings of Central European Conference on Information and Intelligent Systems. Retrieved from http://search.proquest.com/openview/aad715de2556dca67ef5e88a60976dc1/1?pq-origsite=gscholar&cbl=1986354

Thompson, B., Personick, M., & Cutcher, M. (2014). The Bigdata (R) RDF Graph Database. In Harth, A and Hose, K and Schenkel, R (Ed.), Linked Data Management (193–237).

Wilson, J. R. (1996). *Graph Theory*. UK: Addison Wesley.

Exploring graph databases

Abstract

We received certain funds (so-called University grants) a few years ago to explore certain aspects of graph databases. In this study, we decided to share the results of the research to the Croatian academic and professional community. We believe that the research was of high quality and that the results have the potential for wider application and implementation in the available graph database management systems. In addition, there are certain possibilities for expanding the implemented solutions, and there is space for additional research and mapping of relational into graph database concepts.

Keywords: *graph databases; Neo4j; integrity constraints; inheritance; Cypher; Gremlin*