Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/11BE12Jeremic

REALIZACIJA UPITA ISKAZANIH PUTEM OBDA SISTEMA SA RELACIONIM IZVORIMA PODATAKA

IMPLEMENTATION OF QUERIES EXPRESSED IN OBDA SYSTEMS OVER RELATIONAL DATA SOURCES

Aleksandar Jeremić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu predstavljene su osnove OBDA pristupa u upravljanju podacima. Dat je pregled izabranih OBDA tehnologija, i opisan je jedan metod realizacije upita iskazanih prema OBDA sistemima sa relacionim izvorima podataka.*

Ključne reči: *OBDA, ontologije, baze znanja, opisne logike, prevođenje SPARQL upita u SQL*

Abstract – *In this work, we present the foundations of OBDA data management method. Selected OBDA technologies are also presented, as well as one method for execution of queries expressed over OBDA systems with relational data sources.*

Keywords: *OBDA, ontologies, knowledge bases, description logics, SPARQL to SQL rewriting*

1. UVOD

Relacione baze podataka se već godinama unazad koriste kao glavni mehanizam za skladištenje podataka u većini informacionih sistema. Pored mnogih kvalitetnih osobina, relacioni sistemi ograničeni su modelom podataka koji podatke skladišti u relacijama, tj. tabelama. Ponekad relacije nisu idealan mehanizam za opis entiteta koji postoje u realnom svetu, kao i odnosa među njima. Hijerarhije nad entitetima, na primer, komplikuju strukturu šeme baze, ili rezultuju velikim brojem nedostajućih vrednosti obeležja. Isto tako, poznata je i neefikasnost baza podataka za potrebe odgovaranja na kompleksne upite, kada spajanje velikog broja tabela ima negativan uticaj na performanse.

Jedan od predloga za rešavanje navedenih problema relacionog modela jeste metod pristupa podacima zasnovan na ontologijama (eng. *Ontology-Based Data Access*, skr. OBDA). Ontologije, pored definicija struktura podataka, sadrže i definicije aksioma nad tim podacima. Ovi aksiomi definišu semantiku podataka u ontologiji, na način koji nije moguće replicirati u klasičnim relacionim sistemima. Ovo ne znači da OBDA sistemi uklanjaju potrebu za relacionim bazama podataka. Naime, OBDA sistemi obezbeđuju definisanje strukture podataka preko ontologija, dok sami podaci mogu biti smešteni u različitim izvorima. U OBDA sistemima najčešći izvori podataka za prikaz putem modela definisanih ontologijama jesu upravo relacione baze podataka.

U ovom radu biće analiziran problem prevođenja upita postavljenih nad ontologijama u odgovarajući oblik izvršiv nad relacionom bazom podataka koja se koristi kao izvor podataka za datu ontologiju. Definisaćemo korake u realizaciji upita, i demonstrirati sam proces na primeru. Cilj rada je da se na sistematičan način prikaže proces realizacije upita u OBDA sistemima, što će biti od koristi programerima u implementaciji daljih unapređenja u samom procesu prevođenja, ali i korisnicima OBDA sistema, u konstrukciji upita koji će bolje iskoristiti postojeće optimizacije u procesu prevođenja.

Pored uvoda i zaključka, ovaj rad sadrži još tri poglavlja. U poglavlju 2 objašnjene su osnove OBDA pristupa. Objašnjeni su pojmovi baza znanja, ontologija i opisnih logika. Poglavlje 3 daje pregled izabranih OBDA tehnologija koje se mogu koristiti za izgradnju OBDA sistema, dok je u poglavlju 4 predočen jedan metod realizacije upita postavljenih prema OBDA sistemima sa relacionim izvorima podataka.

2. OSNOVE OBDA

OBDA predstavlja skup tehnika, algoritama i sistema koji se koriste za predstavljanje podataka, ali i znanja, putem ontologija [1]. Ontologije posmatramo kao način da formalno predstavimo i opišemo entitete iz realnog sveta, i odnose medu njima. Pomoću ontologija, svaki tip entieta koji postoji u nekom sistemu možemo predstaviti putem odgovarajućeg imenovanog koncepta (eng. *concept*), a svaku pojavu tog tipa entieta putem odgovarajućih objekata koji su instance datog koncepta (eng. *individuals*). Veze izmedu entiteta predstavljaju se putem uloga (eng. *roles*), a na sličan način predstavljaju se i obeležja ovih entiteta. Takođe, ontologije mogu da sadrže i razne druge aksiome koji opisuju različite odnose izmedu entiteta, kao i veza koje se javljaju medu njima.

3.1. *Baze znanja i ontologije*

Sastavni deo OBDA sistema jesu baze znanja (eng. *knowledge bases*), koje sadrže znanje opisano putem neke ontologije. Bitno je ustanoviti razliku izmedu baza podataka i baza znanja. Baze podataka predstavljaju organizovane kolekcije podataka, i njima se pristupa u cilju čitanja podataka koji se čuvaju u bazi. Baze znanja sadrže podatke koji predstavljaju činjenice o realnom svetu, ali takođe sadrže i skup aksioma kojima su formalizovana pravila koja važe u realnom svetu. Pored čitanja podataka, bazama znanja pristupa se i sa ciljem

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Ivan Luković, red.prof.

donošenja određenih zaključaka o podacima. Ovaj postupak otkrivanja skrivenih zakonitosti u podacima naziva se rezonovanje.

Nad bazom znanja uočavaju se 2 nivoa – razlikuju se znanje na nivou intenzije, i znanje na nivou ekstenzije [2]. Na nivou intenzije, znanje se u bazi čuva u obliku aksioma o konceptima, sadržanih u komponenti s nazivom TBox. Na nivou ekstenzije, znanje u bazi se čuva u obliku tvrdnji o pojedinačnim objektima, sadržanih u komponenti pod imenom ABox. Znanje sadržano u TBox-u se generalno posmatra kao nepromenljivo, dok je znanje saržano u ABox-u podložno čestim promenama.

Ontologije, iako se često poistovećuju sa bazama znanja, ipak su samo formalizam za predstavljanje tog znanja. Ontologije su obično opisane preko jezika ontologija, a za ovu namenu se najčešće koriste jezici bazirani na opisnim logikama (eng. *description logics*).

2.1. Opisne logike

Opisne logike omogućavaju opisivanje problemskog domena ontologije putem aksioma u TBox-u ontologije i tvrdnji o pripadnosti u ABox-u ontologije. Aksiomi u TBox-u mogu biti definicije koncepata, uloga, kao i različitih odnosa koji postoje izmedu koncepata i uloga. Koncepti definišu skupove, osnosno klase objekata, dok se uloge koriste da predstave binarne relacije izmedu objekata, tj. instanci koncepata. Tvrdnje o pripadnosti u ABox-u govore o pripadnosti objekata koji predstavljaju entitete realnog sveta nekom konceptu.

3. OBDA TEHNOLOGIJE

Za potrebe realizacije OBDA sistema razvijen je velik broj alata i jezika, objedinjenih pod nazivom OBDA tehnologije. U nastavku su predstavljene izabrane OBDA tehnologije.

3.1. *Protégé* i *Ontop*

Protégé je softverski alat otvorenog koda nastao sa ciljem da podrži modelovanje velikih ontologija. *Protégé* takođe pruža i usluge rezonovanja nad ontologijama, postavljanja upita nad ontologijama, kao i izvoz modelovanih ontologija u raznim formatima. Arhitektura alata je zasnovana na komponentama, te su funkcionalnosti alata proširive raznim *plugin*-ovima. Ovo omogućava korišćenje alternativnih mehanizama za vizuelizaciju, sistema rezonovanja, podršku sistema za upravljanje verzijama, i druge funkcionalnosti.

Ontop je OBDA sistem otvorenog koda namenjen upravljanju virtuelnim grafovima, tj. bazama znanja. *Ontop* ima za cilj da korisniku pruži ontologiju kao konceptualni, virtuelni pogled nad relacionom bazom podataka. Kažemo da je ontologija virtuelna jer je njen ABox predstavljen podacima u bazi podataka, a ne preko tvdnji pripadnosti kakve se obično nalaze u ontologijama. Nad ovim virtuelnim ontologijama moguće je postavljati upite preko SPARQL jezika, i *Ontop* će ih prevesti u odgovarajući SQL (eng. *Structured Query Language*) oblik koji će se potom izvršiti nad bazom podataka.

3.2. RDF i OWL

RDF (eng. *Resource Description Framework*) je jezik namenjen opisivanju resursa na vebu. Resursi predstavljeni RDF-om identifikuju se putem IRI-ja (eng. *Internationalized Resource Identifier*) ili literala (stringovi sa pratećom specifikacijom tipa), a opisani su svojim obeležjima (eng. *properties*) i vrednostima tih obeležja. Povezani podaci u jednom RDF dokumentu formulišu graf (takozvani RDF graf), i nekada se taj graf predstavlja preko RDF trojki (eng. *triplets*). Svaka RDF trojka predstavlja dva čvora grafa povezana ivicom.

Na bazi RDF-a nastao je jezik OWL (eng. *Web Ontology Language*), namenjen opisivanju ontologija. Iako je originalno bio namenjen potrebama semantičkog veba, OWL se danas koristi u mnogim alatima za modelovanje ontologija, kao i u sistemima za rezonovanje (eng. *semantic reasoners*), u mnogim domenima primene.

3.3. R2RML

R2RML je deklarativni jezik za opis preslikavanja izmedu relacionih baza podataka i RDF dokumenata. Preslikavanja se definišu na nivou logičkih tabela (eng. *logical tables*), što mogu biti tabele ili pogledi u bazi podataka, ili proizvoljni SQL upiti, takozvani R2RML pogledi (eng. *R2RML views*).
Svako preslikavanje je opisano preko skupa pravila preslikavanja (eng. *triples map*). Ova pravila preslikavaju svaku pojedinačnu torku logičke tabele na odgovarajući skup RDF trojki.

3.4. SPARQL

SPARQL je jezik za pisanje upita nad RDF grafovima. Upiti se sastoje od šablona trojki sa promenljivama i vrednosti podataka iz RDF grafa koje, kada su dodeljene ovim promenljivama u šablonu, daju ispravnu formulu spram podataka u RDF grafu, predstavljaju rezultate izvršenja SPARQL upita.
Konjunkciju šablona torki u telu upita nazivamo osnovnim šablonom grafa (eng. *basic graph pattern*). Pored ovih jednostavnih konjunkcija šablona torki, u upitu se mogu naći i izrazi za filtriranje, uniranje, spoj, opcioni šabloni, i drugo.

4. FORMULISANJE I IZVRŠAVANJE SPARQL UPITA

U ovom poglavlju će biti objašnjen jedan metod realizacije SPARQL upita postavljenih nad ontologijama sa relacionih izvorima podataka, po uzoru na [3]. Ovaj metod je implementiran i u *Ontop* sistemu. U cilju objašnjenja procesa realizacije upita, koristi se primer šeme relacije `Radnik` koja opisuje radnika u sistemu:

Naziv kolone	Tip podataka	Dozvoljena `null` vrednost	Opis
`ID`	`NUMBER(38,0)`	ne	primarni ključ
`IME`	`VARCHAR(20)`	da	ime radnika
`PREZIME`	`VARCHAR(20)`	da	prezime radnika

U sistemu postoji i ontologija u čijem je TBox-u definisan koncept *Radnik*. Ovaj koncept je putem atributa *imaIme* i *imaPrezime* povezan sa vrednostima tipa `xsd:string`. ABox ove ontologije je prazan, jer se podaci o objektima nalaze u relacionoj bazi podataka.

Putem takozvane *Quest* sintakse R2RML jezika, definisano je sledeće preslikavanje *m* između šeme relacije `Radnik` i odgovarajućih koncepata i atributa u ontologiji:

```
select ID, IME, PREZIME from RADNIK
                  ⇝
 :Radnik-{ID} a :Radnik ;
         :imaIme {IME} ;
         :imaPrezime {PREZIME}.
```

U primeru, iskorišćena je vrednost primarnog ključa relacije `Radnik` za generisanje identifikatora objekata koncepta *Radnik*. SQL upit koji selektuje torke iz tabela predstavlja izvor (eng. *source*). Šablon trojki koji opisuje trojke koje će biti generisane preslikavanjem predstavlja odredište (eng. *target*).

Nad ontologijom, dat je sledeći SPARQL upit *q*, koji selektuje imena i prezimena svih radnika:

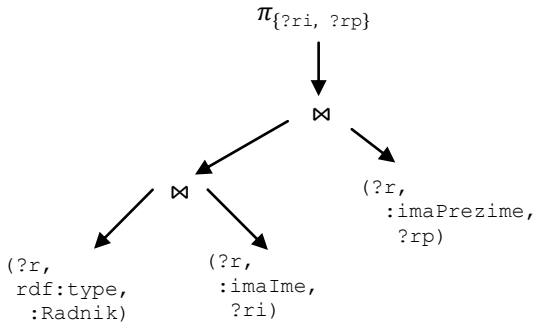
```
select ?ri ?rp
where {
    ?r rdf:type :Radnik ;
       :imaIme ?ri ;
       :imaPrezime ?rp .
}
```

U *Ontop*-u, proces odgovaranja na zadati SPARQL upit sastoji se od sledećih pet faza:

1. SPARQL upit se prepisuje u odgovarajući prepis (eng. *rewriting*) tog upita,
2. ovako prepisani SPARQL upit se potom prevodi u odgovarajući SQL upit,
3. ovaj SQL upit se optimizuje na nivou *Ontop* alata,
4. SQL upit se šalje na izvršenje RDBMS-u i
5. rezultat izvršenja SQL upita se prevodi u odgovor originalnog SPARQL upita.

Prepisivanje upita ima za cilj da pojednostavi polazni upit, oslanjajući se na aksiome u TBox-u. Navedeni upit *q* dovoljno je jednostavan da njegovo prepisivanje ne bi rezultovalo efikasnijim upitom, te se u ovom tekstu nećemo detaljnije baviti ovom fazom realizacije upita.

Druga faza podrazumeva prevođenje SPARQL upita u odgovarajući SQL oblik. Prvo, dati SPARQL upit *q* se raspisuje u sledeće stablo izraza SPARQL algebre:



U ovom stablu, simbol ⋈ označava spoj šablona trojki po promenljivama istog naziva, dok je sa π označena projekcija rezultujućeg skupa torki na obeležja definisana u zaglavlju upita.

Za potrebe prevođenja ovog upita u odgovarajući SQL oblik, *Ontop* generiše skup *T*-preslikavanja M_T na osnovu skupa preslikavanja koje je korisnik eksplicitno zadao. Ranije dato preslikavanje *m*, na primer, bilo bi iskorišćeno za kreiranje sledeća tri *T*-preslikavanja:

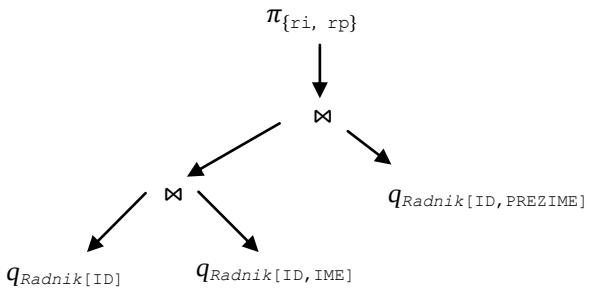
```
        select ID from RADNIK
                  ⇝
 :Radnik-{ID} a :Radnik .

select ID, IME from RADNIK
                  ⇝
 :Radnik-{ID} :imaIme {IME} .

select ID, PREZIME from RADNIK
                  ⇝
:Radnik-{ID} :imaPrezime {PREZIME} .
```

Više informacija o *T*-preslikavanjima čitalac može pronaći u master radu iz koga je proistekao ovaj rad, kao i u [4].

Ova tri *T*-preslikavanja odgovaraju osnovnim šablonima trojki koji se nalaze u listovima SPARQL stabla upita *q*. Šabloni trojki u listovima stabla upita koriste promenljive, dok se na istim mestima u šablonima trojki iz preslikavanja nalaze izrazi nad obeležjima šeme relacije. Stoga se dato SPARQL stablo prevodi u sledeće stablo relacione algebre:



U ovom stablu su sa $q_{Radnik[ID]}$, $q_{Radnik[ID,IME]}$, $q_{Radnik[ID,PREZIME]}$ označeni sledeći upiti:

```
select
   ":Radnik-" || cast(ID as varchar(4000)) as r
from (select ID from RADNIK)

select
   ":Radnik-" || cast(ID as varchar(4000)) as r,
   IME as ri
from (select ID, IME from RADNIK)
where IME is not null

select
   ":Radnik-" || cast(ID as varchar(4000)) as r,
   PREZIME as rp
from (select ID, PREZIME from RADNIK)
where PREZIME is not null
```

Ovi upiti generisani su na osnovu SQL upita u izvorima *T*-preslikavanja koja su korišćena za preslikavanje svakog šablona trojke u listovima SPARQL stabla upita *q*. Svaki od ovih upita samo primenjuje određene transformacije nad vrednostima koje vraćaju upiti u izvorima preslikavanja (ca ciljem generisanja identifikatora objekata opisanih preslikavanjem).

Simbolom ⋈ u ovom stablu označen je prirodni spoj relacija koje su rezultat podupita, dok je π projekcija na obeležja rezultata celog upita. Upit koji predstavlja ovo stablo nazivamo razvoj (eng. *unfolding*) upita *q* po skupu *T*-preslikavanja M_T, u oznaci $unf(q, M_T)$.

U trećoj fazi realizacije upita *q*, upit $unf(q, M_T)$ se optimizuje. *Ontop* sprovodi strukturalne i semantičke optimzacije. Stukturalne optimizacije čine upit efikasnijim tako što ispravljaju strukturalne nedostatke forme upita. Semantičke optimizacije u optimizaciji upita oslanjaju se na ograničenja u relacionoj bazi podataka koja se koristi kao ABox.

Jedna strukturalna optimizacija koja se može sprovesti nad upitom $unf(q, M_T)$ jeste izravnavanje (eng. *flattening*) podupita u listovima stabla. Ugnježdeni upiti koji prate

`from` klauzulu u ovim podupitima ne menjaju vrednosti koje selektuju iz tabela (niti vrše bilo kakvo filtriranje), te se ovi ugnježđeni upiti mogu zameniti nazivima tabela iz kojih se čitaju ti podaci.

Na primer, podupit $q_{Radnik[ID]}$ se modifikuje tako da postane:

```
select
   ":Radnik-" || cast(ID as varchar(4000)) as r
from RADNIK
```

Na sličan način se modifikuju i preostala dva podupita.

Ova tri upita čitaju vrednosti obeležja iste tabele `Radnik`, te se sva tri upita mogu objediniti u jedan. U pitanju je semantička optimizacija koja se oslanja na ograničenje primarnog ključa definisano nad obležjem `ID` šeme relacije `Radnik`. U uslovu selekcije ovog "objedinjenog" upita nalazi se konjunkcija uslova selekcije iz upita koji su objedinjeni. Na ovaj način se iz upita $unf(q, M_T)$ gube svi prirodni spojevi relacija. Nakon ove optimizacije, stablo relacione algebre upita $unf(q, M_T)$ izgleda ovako:

$$\pi_{\{ri,\ rp\}}$$

$$\downarrow$$

$$q_{Radnik[ID,IME,PREZIME]}$$

U ovom stablu, sa $q_{Radnik[ID,IME,PREZIME]}$ je označen sledeći upit:

```
select
   ":Radnik-" || cast(ID as varchar(4000)) as r,
   IME as ri,
   PREZIME as rp
from RADNIK
where IME is not null and PREZIME is not null
```

Poslednja optimizacija koja se može sprovesti nad upitom $unf(q, M_T)$ jeste objedinjavanje projekcije u korenu upita sa projekcijom u podupitu. Na ovaj način se iz rezultata upita kompletno uklanja kolona `r`, koja je inače sadržala identifikatore objekata.

Finalni oblik upita $unf(q, M_T)$ koji se dobija kao rezultat treće faze realizacije upita q jeste:

```
select IME as ri, PREZIME as rp
from RADNIK
where IME is not null and PREZIME is not nul
```

U četvrtom koraku SQL upit je već dostavljen na izvršavanje RDBMS-u. Peti korak podrazumeva preuzimanje rezultata od RDBMS-a i prevodenje rezultata u oblik koji odgovara promenljivama iz zaglavlja originalnog SPARQL upita. S obzirom da su svim kolonama rezultata SQL upita već dodeljeni aliasi spram naziva promenljivih iz SPARQL upita, ovaj korak prevodenja postaje trivijalan.

5. ZAKLJUČAK

U ovom radu je predstavljen OBDA pristup, kao i odabrane OBDA tehnologije koje se mogu koristiti u izgradnji OBDA sistema. Takođe je analiziran i proces realizacije upita postavljenih prema OBDA sistemima koji se oslanjaju na relacione izvore podataka.

U radu je dat i primer prevođenja jednog SPARQL upita u SQL oblik. Dati primer je relativno jednostavan, i odabran je sa ciljem demonstracije procesa prevođenja. U praktičnoj upotrebi OBDA sistema ipak treba očekivati da će prema sistemu biti postavljani i razni kompleksniji upiti. Na primer, izveštajna funkcija preduzeća oslanja se u velikoj meri na agregirane podatke o poslovanju, te je od izuzetnog značaja da informacioni sistem preduzeća podrži i upite sa agregacionim funkcijama. Ovo je kategorija u kojoj pristup opisan u ovom radu počinje pokazivati svoje nedostatke. *Ontop* sistem još uvek nema podršku za upite sa agregacionim funkcijama. Isto tako, prevođenje SPARQL naredbi za upravljanje podacima u SQL i dalje nije podržano, već se ove operacije moraju sprovoditi direktno nad podacima u relacionoj bazi podataka.

Poseban problem, ipak, prepoznat je u samoj ideji relalizacije upita u OBDA sistemima, na način kako je to opisano u ovom radu. Ontologija opisana putem OWL jezika zapravo je RDF graf. Jezik SPARQL namenjen je isključivo postavljanju upita nad RDF grafovima, te ne postoji način iskorišćenja semantike OWL konstrukcija kojima je ontologija opisana, pri konstruisanju upita. To takođe znači da ne postoji mogućnost korišćenja velikog broja konstruktora koje opisne logike pružaju, u konstruisanju upita (kakvi su, na primer, konstruktori egzistencijalnog i unverzalnog kvantifikatora, prisutni u većini opisnih logika). Ovaj problem bi se mogao rešiti proširenjem SPARQL jezika tako da omogući iskorišćavanje informacija koje pruža OWL struktura RDF dokumenta nad kojim se postavlja upit, ili, idealno, razvojem kompletnog novog jezika za postavljanje upita nad OWL ontologijama.

Opis faza prevođenja SPARQL upita u SQL dat u ovom radu može biti od pomoći u implementaciji novih optimizacija u *Ontop*-ov sistem za prevođenje, kao i u unapređenju postojećih optimizacija. U slučaju kreiranja novog softvera za realizaciju upita u OBDA sistemima, ili novih jezika za postavljanje upita nad OWL ontologijama, ovaj rad može pružiti informacije bitne za razumevanje zahteva sistema, i problema koji su prisutni u postojećim rešenjima.

6. LITERATURA

[1] Oscar Corcho, Freddy Priyatna & David Chaves-Fraga, Towards a new generation of ontology based data access, *Semantic Web Preprint (2019)*, pp. 1–8.

[2] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider & Daniele Nardi, *The description logic handbook: Theory, implementation and applications*, Cambridge university press, 2003.

[3] Dag Hovland, Davide Lanti, Martin Rezk & Guohui Xiao, OBDA constraints for effective query answering, *International Symposium on Rules and Rule Markup Languages for the Semantic Web*, Springer, 2016, pp. 269–286.

[4] Mariano Rodriguez-Muro, Roman Kontchakov & Michael Zakharyaschev, Ontology-based data access: Ontop of databases, *International Semantic Web Conference*, Springer, 2013, pp. 558–573.

Kratka biografija:

Aleksandar Jeremić rođen je u Rumi 1995. god. Bečelor rad iz oblasti Elektrotehnike i računarstva – Računarske nauke i informatika odbranio je 2018. godine. Master rad iz iste oblasti odbranio je 2020. god.