PRIMJENA ONTOLOGIJA U ORACLE INFORMACIJSKIM SUSTAVIMA

INCORPORATION OF ONTOLOGIES INTO ORACLE BASED INFORMATION SYSTEMS

Vedran Vrbanić, Martina Kocet

Koios savjetovanje d.o.o.

Radnička cesta 52, 10000 Zagreb

vedran.vrbanic@koios.hr; martina.kocet@koios.hr

SAŽETAK

Shema Oracle baze podataka opisuje strukturu objekata koji čine bazu. S druge strane, ontologija definira strukturu i sadržaj područja znanja, odnosno semantiku sadržanu u podacima. Ontologije se sastoje od klasa, instanci, pravila i hijerarhija i kao takve čine solidnu podlogu za definiciju domena poslovnog znanja. Oracle baza podržava rad s ontologijama ali definiranje i održavanje ontologija zahtijeva dodatan angažman. Kroz jednostavan poslovni primjer iz bankovne industrije ilustrirat ćemo dizajn ontologije i definiranje poslovnih pravila. Usporedit ćemo performanse upita nad podacima u relacijskim tablicama s performansama upita u ontologiju.

ABSTRACT

The Oracle database schema represents the structure of database objects. On the other hand, ontology defines the structure and content of an area of knowledge, i.e. data semantics. Ontologies consist of classes, instances, rules and hierarchies and therefore present the solid foundation for the description of business knowledge domains. The Oracle database supports the storage of semantic data and ontologies but the definition and maintenance of ontologies require additional engagement. By conducting a simple banking case study, designing of an ontology and incorporation of business rules within it will be presented. Performances of querying relational data will be compared to performances of querying ontologies.

1. UVOD

Velika količina podataka i raznolikost formata u kojima su zapisani predstavljaju izazov sustavima za upravljanje bazama podataka. Razumijevanje semantike, odnosno znanja pohranjenog u podacima, u velikoj mjeri doprinosi fleksibilnosti i korisnosti informacijskih sustava, posebno kod integracije podataka iz heterogenih izvora. Ukoliko je riječ o sustavima skladišta podataka, isti su zamišljeni kao transparentni i točni izvori informacija za sve poslovne korisnike poduzeća ili institucije. Ali situacija nije uvijek tako blistava, pogotovo za velike poslovne subjekte sa složenim poslovnim modelima. Najkompleksniji i najzahtjevniji izvještajni sustavi u pravilu su sustavi financijskih institucija. Velik broj izvora podataka, nekonzistentne vrijednosti podataka na izvorima i potreba za složenim transformacijama podataka pri pokušajima unifikacije istih u skladištu znaju rezultirati izrazito kompleksnim izvještajnim sustavima s velikim troškovima razvoja i održavanja (ovo se posebno odnosi na naknadne nadogradnje takvih sustava). Korisne poslovne informacije zapisane u ovakvim

sustavima često su skrivene u gomili podataka i njihovo značenje je često znano samo nekolicini najnaprednijih korisnika (poslovnog i/ili IT sektora).

Treća normalna forma relacijskih baza omogućuje pohranu bilo kojeg poslovnog podatka. Međutim, ER (*Entity Relationship*) model kojim je treća normalna forma opisana često se sastoji od kompleksne mreže tablica, ograničenja (*constraints*), stranih ključeva i ostalih objekata u bazi. Transformacija treće normalne forme u denormalizirane zvjezdaste ili pahuljaste modele koji se koriste u sustavima skladišta podataka rezultira uvođenjem novih kolona, zastavica ili pojavom normaliziranih struktura.

Poslovna pravila koja se koriste pri učitavanju, transformaciji i unifikaciji podataka najčešće ostaju zapisana samo unutar PL/SQL koda, iako bi deklarativno sav programski razvoj i izmjene morali biti zapisani i u dokumentaciji. Izmjene poslovnih potreba i procesa uvjetuju redizajn ili nadogradnje postojećeg izvještajnog sustava, a praćenje ovisnosti novih funkcionalnosti o postojećima i održavanje sustava postaje velik trošak poduzeća ili institucije. Navedeno rezultira nefleksibilnim i tromim izvještajnim sustavima koji ne ispunjavaju svoju zadaću u potpunosti, a to je brz, točan i transparentan prikaz informacija korisnicima.

Stoga ima smisla govoriti o sustavima koji bi implicitno donekle bili “svjesni“ semantike podataka koji su u njima pohranjeni.

2. TEHNOLOGIJE SEMANTIČKE POHRANE PODATAKA

Relacijske baze podataka sadrže tablice koje se sastoje od stupaca (atributa) u kojima su pohranjene određene vrijednosti. Koristeći SQL upite programeri mogu raditi s podacima zapisanim u tablicama. Bitno je primijetiti da se pri spajanju tablica ne moraju koristiti nikakva pravila koja bi podatke smještale u razuman kontekst. Na primjer, moguće je spojiti tablicu klijenata i tablicu ugovora po kriteriju da je iznos ugovora jednak godinama starosti klijenta. Iako legitiman, ovakav upit semantički nema smisla.

Dakle, relacijske baze podataka temelje se na podatkovnom modelu i omogućuju rad s podacima ali tim podacima ne dodjeljuju kontekst u kojem su smješteni. Ali ukoliko se pri pohrani podataka upotrijebe pravila koja određuju odnose između podataka zapisanih u repozitoriju, tada više nije riječ o pohrani podataka, nego o pohrani informacija. Kako bi omogućili smještanje podataka u odgovarajući kontekst i pohranu informacija, semantički repozitoriji temeljeni na ontologijama koriste usmjerene grafove. Kroz pregled tehnologija koje je nužno koristiti pri izgradnji semantičkog repozitorija prikazat ćemo evoluciju pohrane i korištenja semantičkih podataka te konačno i definirati što je to ontologija.

2.1. Resource Description Framework

Resource Description Framework (RDF) je model podataka razvijen s ciljem omogućavanja dijeljenja informacija putem usluge World Wide Weba (WWW). Model je definiran od strane organizacije W3C (skupina eksperata pod vodstvom izumitelja WWW-a, Tim Berners Lee-a, koja se bavi dizajnom i standardizacijom web tehnologija) i temelji se na zapisivanju informacija u obliku trojki. Trojka je logička izjava koju čine:

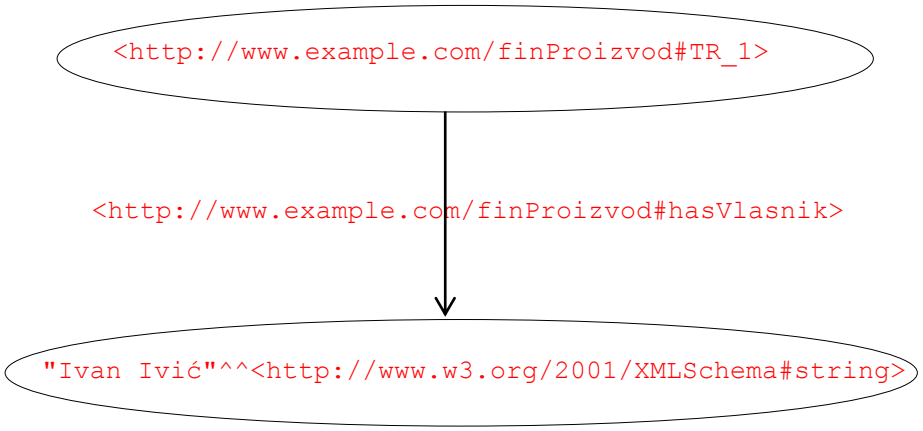
- subjekt – jedinstveni identifikator opisanog resursa (resurs je bilo koji objekt opisan RDF-om)
- predikat – označava relaciju između subjekta i objekta
- objekt – vrijednost dodijeljena subjektu

Semantička podrška Oracle baze podržava RDF. Slijedi primjer upisa trojke u tablicu pravila PRODUCT_TPL:

```
INSERT INTO PRODUCT_TPL VALUES
  (ID_SEQ.NEXTVAL,
  SDO_RDF_TRIPLE_S
       ('finProizvodModel',
       '<http://www.example.com/finProizvod#TR_1>',
       '<http://www.example.com/finProizvod#hasVlasnik>',
       '"Ivan Ivić"^^<http://www.w3.org/2001/XMLSchema#string>'
       )
  );
```

Tablicu pravila PRODUCT_TPL kreira programer, a sastoji se od ID-a i objektnog tipa SDO_RDF_TRIPLE_S. Isti se koristi za pohranu semantičkih podataka, odnosno trojki, a posljednja tri parametra pripadajućeg mu konstruktora odgovaraju subjektu, predikatu i objektu pravila. Primjerom je definirano da je Ivan Ivić vlasnik tekućeg računa TR_1.

U kontekstu RDF-a podrazumijeva se pohrana informacija, a ne podataka, i to u obliku trojki. Trojke čine usmjereni graf čiji su čvorovi subjekt i objekt dok je odnos između subjekta i objekta određen usmjerenom granom grafa, odnosno predikatom. U Oracle bazi se za pohranu trojki koristi SDO_RDF_TRIPLE_S objektni tip, ali se usmjereni graf također može zapisivati i u drugim formatima, od kojih je najpoznatiji RDF/XML.



Slika 2.1 Usmjereni RDF graf s dva čvora i jednom granom

Bitna stvar koju je moguće definirati unutar RDF-a je predikat *rdf:type* pomoću kojeg se bilo kojem resursu može dodijeliti određeni tip. Primjena ovog atributa bit će prikazana u sljedećem odlomku, u okviru pregleda RDFS-a (*Resource Descripion Framework Schema*), koji je nadogradnja RDF-a.

2.2. Resource Description Framework Schema

RDF skup specifikacija omogućuje zapisivanje informacija u širem smislu, što znači da nema ograničenja svojstava po vrstama objekata, odnosno korisnik može definirati bilo kakvo svojstvo za bilo koji resurs. Slično kao što *XML Schema* ograničava dozvoljenu strukturu XML dokumenata, *RDF*

Schema definira dozvoljeni vokabular RDF modela podataka. Osim toga, RDFS omogućuje definiranje klasa, nasljeđivanje i kreiranje instanci klasa po sličnom principu koji se koristi u objektno orijentiranom programiranju. Također je omogućeno definiranje domene i kodomene atributa klasa. Konstrukti RDFS-a koji se koriste za navedeno su:

- *rdfs:Class* – omogućuje definiciju klasa
- *rdfs:subClassOf* – omogućuje definiranje hijerarhije između klasa
- *rdfs:domain, rdfs:range* – koriste se za definiciju domene i kodomene predikata

Uporaba gore navedenih konstrukata bit će ilustrirana na primjeru u trećem poglavlju.

Nad formalnim jezikom RDFS također postoji nadogradnja naziva *Web Ontology Language* (OWL). OWL je također definiran od strane tijela W3C s ciljem razmjene informacija na WWW-u i obogaćuje RDFS uvođenjem kompleksnijeg vokabulara. Između ostalog, OWL omogućuje definiranje predikata koji su tranzitivni, inverzni ili jedinstveni, ograničenja kardinalnosti, definiranje međusobno isključivih klasa i tako dalje. Unutar OWL-a postoji daljnja podjela na verzije *OWL Lite, OWL DL i OWL Full* koje se razlikuju po kompleksnosti vokabulara, gdje je verzija *OWL Full* najkompleksnija. I RDFS i OWL su jezici za pohranu ontologija, a budući da su ukratko prezentirani u prethodnim odlomcima, u nastavku slijedi i formalna definicija ontologije.

2.3. Što je ontologija?

Ontologija je formalna, eksplicitna specifikacija kojom se opisuju koncepti iz određene domene. Sastoji se od klasa, atributa klasa, instanci i odnosa između klasa. Može se reći da je ontologija specifikacija konceptualizacije određene domene. Praktičnija definicija bi glasila da je ontologija uređen vokabular na temelju kojeg računalo raspoznaje značenje pojedinih vrijednosti (riječi, brojevi, datumi, ...) u različitim kontekstima.

Jezici RDFS i OWL omogućuju kreiranje dozvoljenog vokabulara kojim su definirani resursi, odnosno koncepti, neke poslovne domene, kao i relacije između tih resursa i to u obliku pravila zapisanih u trojkama. Stoga se RDFS i OWL koriste za pohranu ontologija, s tom razlikom da OWL nudi bogatiji vokabular za definiciju odnosa.

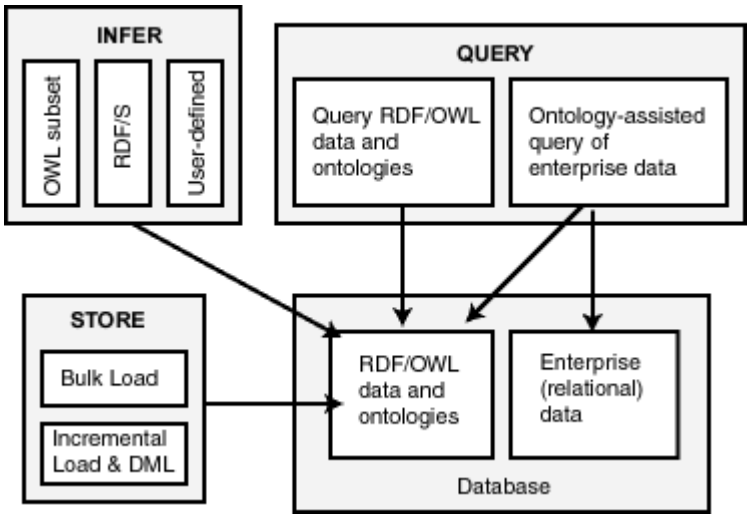
2.4. Rezoniranje (*inference*)

Bitna stvar kod ontologija je mogućnost rezoniranja, to jest razrješavanja postojećih i kreiranja novih pravila na temelju hijerarhijskih odnosa zapisanih u ontologiji kao i na temelju korisničkih pravila kojima se eksplicitno opisuju zakonitosti poslovne domene. Za razliku od isključivo sintaksnog spajanja podataka u jeziku SQL (*klijenti.postanski_broj = mjesta.postanski_broj*), rezoniranje koristi odnose između koncepata i zapisane podatke s ciljem dobivanja novih informacija.

Na primjer, neka podaci o dva poslovno različita bankovna proizvoda, depozitu i prekoračenju po tekućem računu, pristižu u skladište podataka iz različitih izvora. Prije nego se učitaju u skladište podataka, podaci se učitavaju u ontologiju za provjeru kvalitete podataka. U istoj je navedeno pravilo: “Ako je depozit vezan uz isti tekući račun kao i prekoračenje, tada i depozit i prekoračenje imaju istog vlasnika“. Ukoliko za proizvod depozit s izvora dođu nepotpuni podaci u ontologiju (bez popunjenog vlasnika), rezoniranje dotičnom depozitu može dodijeliti vlasnika ukoliko je vlasnik upisan uz proizvod prekoračenje po tekućem računu. Dakle, iako podatak o vlasniku depozita na izvoru nije bio popunjen, u ontologiji postoji.

2.5. Skica semantičkih mogućnosti Oracle baze

U poglavlju 2.1 već je naveden primjer umetanja trojke u tablicu pravila u Oracle bazi. Osim podrške spremanju trojki, odnosno ontologija, baza pruža mogućnosti upita nad semantičkim podacima koristeći upitni jezik SPARQL (više o njemu u primjerima koji slijede), rezoniranja ali i kombiniranja upita nad podacima zapisanim u ontologiji s podacima zapisanim u relacijskim tablicama. Ilustracija semantičkih mogućnosti Oracle baze prikazana je na slici 2.2:



Slika 2.2 Semantičke mogućnosti Oracle baze

U istoj instanci baze pohranjeni su i podaci u relacijskim tablicama i podaci u ontologijama. Trojke se mogu puniti iz vanjskih izvora *batch* obradom ili inkrementalno koristeći INSERT naredbe. *Inference engine* baze podržava RDFS u cijelosti, dio OWL konstrukata te razrješavanje pravila definiranih od strane korisnika. U nastavku referata slijedi poslovni primjer kroz koji će biti ilustrirana uporaba ontologije i konstrukata opisanih u prethodnim poglavljima.

3. UPORABA ONTOLOGIJA U PRAKSI

Za potrebe ilustracije uporabe ontologija na primjeru iz bankarske industrije, opisat ćemo nekoliko jednostavnih financijskih proizvoda - tekući račun, depozit, prekoračenje po tekućem računu i obročnu otplatu po tekućem računu. Slijedi detaljniji opis ovih proizvoda i poslovnih potreba koje treba modelirati.

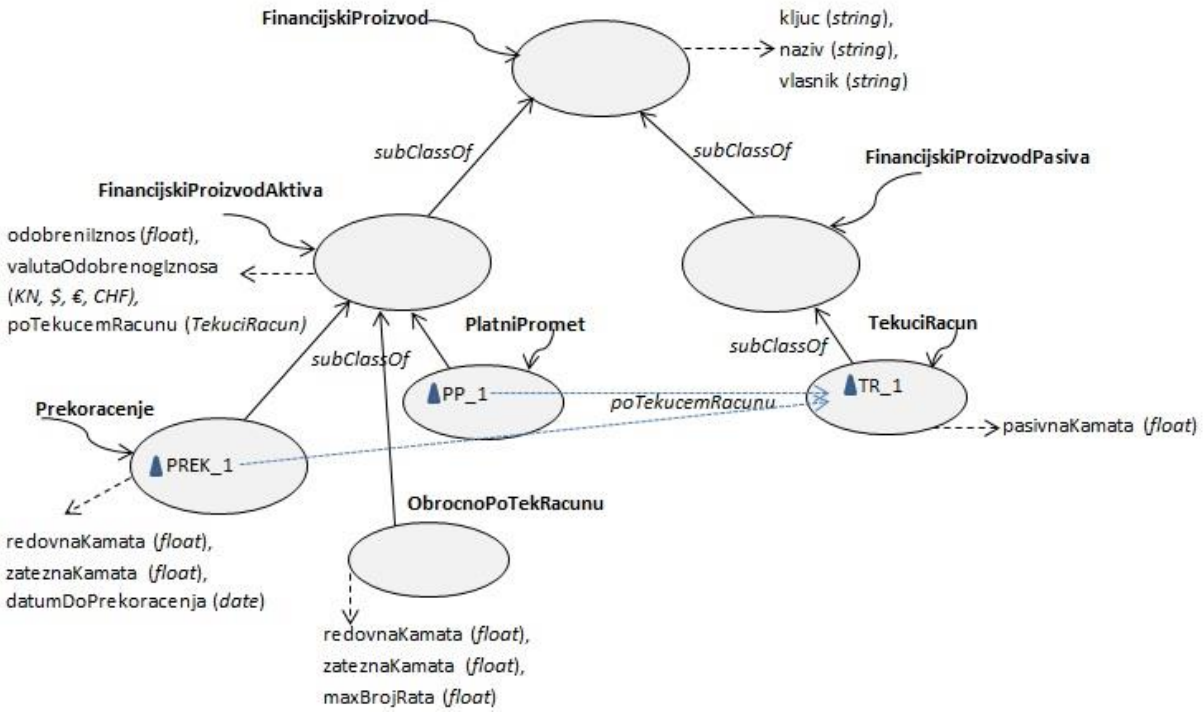
3.1. Opis poslovnog primjera

Poslovni primjer iz financijske industrije bit će opisan iz perspektive banke. Banka prikuplja novac po nižoj cijeni, plasira ga na tržište po višoj cijeni te zarađuje na razlici. Ukoliko klijent u banci ima položen novac, banka mu je taj novac dužna isplatiti u određenom trenutku i stoga takav proizvod (depozit) spada u pasivu banke. Ukoliko je banka klijentu pozajmila novac, očekuje od klijenta da taj novac i vrati. Stoga takav proizvod (kredit) spada u aktivu banke. Iako su različiti proizvodi, depozit i kredit dijele određena svojstva – na primjer, oba moraju imati vlasnika.

U okviru primjera razmatrat ćemo sljedeće proizvode:

- Tekući račun – primarno služi za polaganje novčanih sredstava (depozit).
- Prekoračenje – ukoliko klijent u banci ima otvoren tekući račun, po istom mu može biti odobreno prekoračenje, odnosno minus.
- Obročna otplata po tekućem računu – banka klijentu može odobriti i obročnu otplatu po tekućem računu, u visini određenog limita.
- Platni promet – budući da klijent koristi tekući račun za primanje uplata i obavljanje isplata, za tu uslugu mu banka obračunava platno prometnu naknadu. Za potrebe analitičkih izvještaja, platno prometne naknade se najčešće ne prikazuju po tekućem računu, nego se kreira "umjetni" proizvod naziva *platni promet*.

Hijerarhijski model opisanih proizvoda mogao bi izgledati ovako:



Slika 3.1 Hijerarhija proizvoda

Obojane elipse predstavljaju klase definirane unutar ontologije. Nazivi klasa označeni su valovitom strelicom, a atributi klasa crtkanom ravnom strelicom. *FinancijskiProizvod* je krovna klasa koja sadrži atribute zajedničke svim proizvodima (*kljuc, naziv, vlasnik*). Nasljeđuju je klase

FinancijskiProizvodAktiva i *FinancijskiProizvodPasiva*. U okviru primjera postoji samo jedna vrsta pasivnog proizvoda – tekući račun, ali treba primijetiti da je svaki proizvod aktive vezan uz određeni tekući račun, jer se na temelju otvorenog tekućeg računa otvaraju proizvodi opisani klasama *Prekoracenje, ObrocnoPoTekRacunu* i *PlatniPromet*. Stoga se u klasi *FinancijskiProizvodAktiva* nalazi atribut *poTekucemRacunu*.

Trokutići koji se nalaze unutar obojanih elipsi predstavljaju instance klasa. Vidljivo je da su instance PP_1 i PREK_1 vezane uz tekući račun TR_1.

3.2. Kodiranje modela u ontologiji

Za početak, potrebno je definirati klase. Na primjer, definicija klase *FinancijskiProizvod* glasi:

```
INSERT INTO PRODUCT_TPL VALUES --[1]
 (ID_SEQ.NEXTVAL, --[2]
 SDO_RDF_TRIPLE_S --[3]
       ('finProizvodModel', --[4]
       '<http://www.example.com/finProizvod#FinancijskiProizvod>', --[5]
       '<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>', --[6]
       '<http://www.w3.org/2002/07/owl#Class>' --[7]
       )
);
```

Linije koda imaju sljedeća značenja:

1. PRODUCT_TPL je tablica u koju se spremaju trojke. Kreirana je od korisnika, unutar određenog semantičkog modela, u ovom slučaju *finProizvodModel*.
2. Unos sekvence.
3. Konstruktor Oracle objektnog tipa podatka SDO_RDF_TRIPLE_S u kojem se čuvaju trojke.
4. Naziv semantičkog modela. Korisnik može definirati više modela koji sadrže različite i ne nužno vezane trojke.
5. URI (*Uniform Resource Identifier*) resursa. U ovom slučaju resurs je klasa *FinancijskiProizvod* i u ovoj trojci predstavlja subjekt.
6. URI predikata. Predikat u ovoj trojci označava tip resursa.
7. URI objekta. Označava da je *FinancijskiProizvod* klasa definirana u OWL jeziku. Specifikacija klase jedinstveno je određena URI-em http://www.w3.org/2002/07/owl#Class.

Kako bi se mogla definirati hijerarhija proizvoda, potrebno je koristiti *subClassOf* konstrukt jezika RDFS:

```
INSERT INTO PRODUCT_TPL VALUES --[1]
 (ID_SEQ.NEXTVAL, --[2]
 SDO_RDF_TRIPLE_S --[3]
       ('finProizvodModel', --[4]
       '<http://www.example.com/finProizvod#TekuciRacun>', --[5]
       '<http://www.w3.org/2000/01/rdf-schema#subClassOf>', --[6]
       '<http://www.example.com/finProizvod#FinancijskiProizvodPasiva>' --[7]
       )
);
```

Treba primijetiti da je konstrukt *type* definiran u sklopu pravila RDF (prva *insert* naredba, linija šest), dok je konstrukt *subClassOf* definiran u sklopu RDFS sheme (druga *insert* naredba, linija šest). Definicija klase je pak uzeta iz rječnika OWL (prva *insert* naredba, linija sedam).

Podklase razreda *FinancijskiProizvodAktiva* nasljeđuju svojstvo *poTekucemRacunu* na temelju kojeg je ostvarena veza proizvoda aktive s odgovarajućim tekućim računom. U tablici pravila ovaj odnos izgleda ovako:

Tablica I. Pravila za definiciju svojstva *poTekucemRacunu*

SUBJEKT	PREDIKAT	OBJEKT
example:poTekucemRacunu	rdf:type	owl:ObjectProperty
example:poTekucemRacunu	rdf:type	owl:FunctionalProperty
example:poTekucemRacunu	rdfs:range	_:dummyNode
_:dummyNode	rdf:type	owl:Restriction
_:dummyNode	owl:allValuesFrom	example:TekuciRacun
_:dummyNode	owl:onProperty	example:poTekucemRacunu

U tablici I. umjesto cijele putanje URI-a navedene su kratice *example: rdf:, rdfs:, owl*. Pri radu s trojkama moguće je radi preglednosti definirati kratice koje označavaju prefiks URI-a (takav prefiks naziva se *namespace*). Na primjer, kratica *example:* u ovom slučaju označava znakovni niz *http://www.example.com/finProizvod#.*

- OWL konstrukt *ObjectProperty* označava da je kodomena svojstva *poTekucemRacunu* neka klasa.
- OWL konstrukt *FunctionalProperty* određuje da neka instanca može imati samo jednu vrijednost svojstva *poTekucemRacunu.*
- Zadnja četiri retka u tablici definiraju kodomenu svojstva *poTekucemRacunu* – kodomena smije sadržavati samo objekte tipa *TekuciRacun*.

Prva dva pravila u tablici II. definiraju instance PREK_1 i TR_1. Treće pravilo dodjeljuje prekoračenju PREK_1 vezu na tekući račun TR_1.

Tablica II. Kreiranje instanci i pripadajuće veze

SUBJEKT	PREDIKAT	OBJEKT
example:TR_1	rdf:type	example:TekuciRacun
example:PREK_1	rdf:type	example:Prekoracenje
example:PREK_1	example:poTekucemRacunu	example:TR_1

3.3. Korisnički definirana pravila

Konstrukti (*rdf:type, owl:Class, rdfs:*subClassOf) prikazani u prethodnim primjerima definiraju svojstva i odnose između resursa. Ti su odnosi pohranjeni u Oracle objektu SDO_RDF_TRIPLE_S, u tablicama trojki. Međutim, isto tako korisnici mogu kreirati i specifična poslovna pravila koristeći SPARQL sintaksu.

SPARQL (*Simple Protocol and RDF Query Language*) je jezik za pretragu podataka u RDF grafu i pandan je SQL jeziku za pretragu podataka u relacijskim bazama. Poput SQL upita, i SPARQL upit sadrži klauzule SELECT i WHERE u kojima su navedene varijable koje predstavljaju elemente trojki (subjekt, predikat, objekt). Primjeri korištenja SPARQL-a slijede u nastavku.

Ukoliko je prekoračenje bez eksplicitno navedenog vlasnika povezano s tekućim računom (preko atributa *poTekucemRacunu*), tada dotičnom prekoračenju treba dodijeliti istog vlasnika. Pravilo izgleda ovako:

```
INSERT INTO mdsys.semr_proizvodi_rb VALUES( --[1]
  'has_vlasnik_prek', --[2]
  ' (?subjekt1 rdf:type :Prekoracenje) –[3]
   (?subjekt1 :poTekucemRacunu ?subjekt2) –[4]
   (?subjekt2 :hasVlasnik ?v1) ', --[5]
    null, --[6]
   '(?subjekt1 :hasVlasnik ?v1)', --[7]
    . . . .
```

Linije koda imaju sljedeća značenja:

1. Naziv Oracle tablice u MDSYS shemi u kojoj su pohranjena korisnička pravila.
2. Naziv korisničkog pravila.
3. Varijabla *subjekt1* mora biti tipa prekoračenje.
4. Varijabla *subjekt1* vezana je uz tekući račun koji je definiran varijablom *subjekt2*.
5. Tekući račun definiran varijablom *subjekt2* ima vlasnika definiranog varijablom *v1*.
6. Filter – nema ga.
7. Ako su ispunjeni uvjeti navedeni u točkama 2, 3, 4 i 5, tada dodijeli prekoračenju koje je definirano varijablom *subjekt1* vlasnika koji je zapisan u varijabli *v1*.

3.4. Zaključivanje novih pravila na temelju postojećih

U poglavlju 2.4 spomenuta je mogućnost rezoniranja, to jest kreiranja novih pravila na temelju postojećih. Na slici 2.1 definirano je da je Ivan Ivić vlasnik tekućeg računa TR_1. U poglavlju 3.3 definirano je korisničko pravilo po kojem prekoračenje ima istog vlasnika kao i tekući račun na temelju kojeg je odobreno.

Pozivom procedure SEM_APIS.CREATE_ENTAILMENT stvara se *rules index* – objekt kojeg Oracle baza koristi za pretragu i kombiniranje pravila.

```
exec SEM_APIS.CREATE_ENTAILMENT(
       'finProizvod_rix',
       SEM_Models('finProizvodModel'),
         SEM_Rulebases('owlprime','proizvodi_rb'),
         SEM_APIS.REACH_CLOSURE,
       null,
       'USER_RULES=T');
```

Parametrima procedure definira se naziv *rules indexa*, semantički model o kojem je riječ, te skupove pravila koje treba kombinirati – u ovom primjeru rezoniranje se izvršava na temelju konstrukata definiranih vokabularom *owlprime* u kombinaciji s korisnički definiranim pravilima u skupu pravila *proizvodi_rb*. Po završetku procedure CREATE_ENTAILMENT, svojstva prekoračenja PREK_1 mogu se dohvatiti sljedećom naredbom:

```
SELECT  sub,  pred,obj
FROM     TABLE(SEM_MATCH('
        {
          ?sub ?pred ?obj .
          filter(?sub = :PREK_1)
        }',
                SEM_MODELS('finProizvodModel'),
                SEM_RULEBASES('owlprime',  'proizvodi_rb'),
                SEM_ALIASES(SEM_ALIAS(null,'http://www.example.com/finProizvod#'),
                                SEM_ALIAS('owl','http://www.w3.org/2002/07/owl#')),
                null))
      ;
```

SEM_MATCH je Oracle funkcija za pretraživanje semantičkog repozitorija. Treba primijetiti da je to ujedno i tablična funkcija, odnosno da vraća *dataset* tip podataka. To znači da se rezultati iz semantičkog repozitorija mogu kombinirati s relacijskim podacima zapisanim u Oracle bazi, odnosno funkcija SEM_MATCH može se ugnijezditi u bilo koji SQL podupit, korištenjem ključne riječi TABLE. Rezultat ovog upita je:

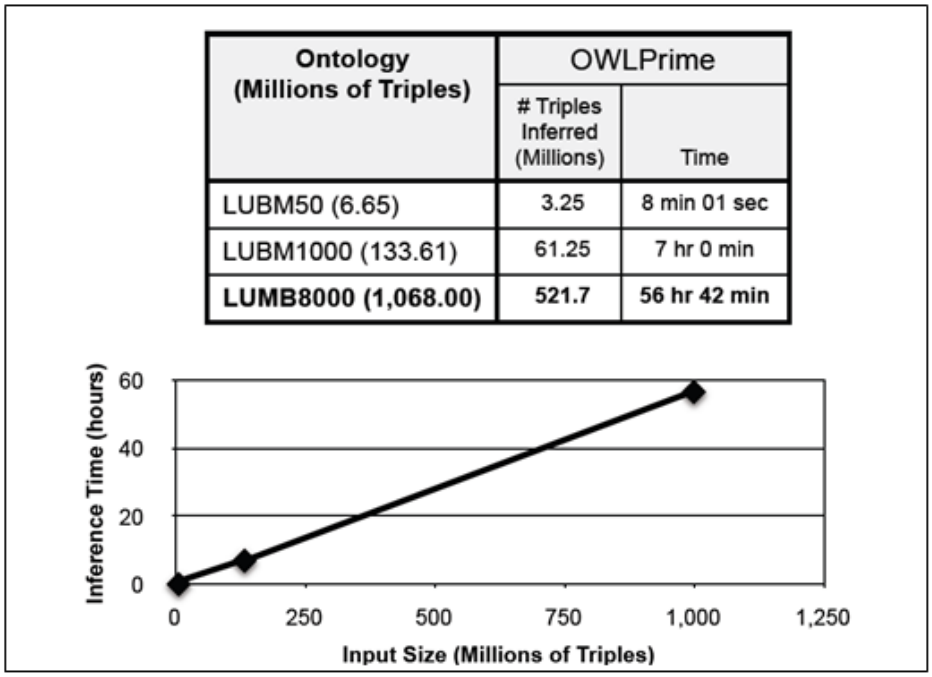
Tablica III. Svojstva prekoračenja PREK_1

SUB	PRED	OBJ
example:PREK_1	rdf:type	example:FinancijskiProizvod
example:PREK_1	rdf:type	example:FinancijskiProizvodAktiva
example:PREK_1	rdf:type	example:Prekoracenje
example:PREK_1	example:poTekucemRacunu	TR_1
example:PREK_1	example:hasVlasnik	Ivan Ivić

Treba primijetiti da činjenica da je Ivan Ivić vlasnik prekoračenja PREK_1 nije bila eksplicitno navedena, nego je rezultat OWL i korisnički definiranih pravila, navedenih prethodno u tekstu.

3.5. Performanse rezoniranja i upita nad semantičkim podacima

Autori Oracle prezentacije [7] su između ostalog proveli i eksperiment nad performansama rezoniranja koristeći LUBM (*Lehigh University Benchmark*) ontologije koje su kreirane u svrhu testiranja od strane SWAT (*The Semantic Web and Agent Technologies Lab*) laboratorija. Testna Oracle baza bila je instalirana na stroju s procesorom frekvencije 3 GHz i 4 GB RAM memorije, a performanse rezoniranja su:



Ontology (Millions of Triples)	OWLPrime	
	# Triples Inferred (Millions)	Time
LUBM50 (6.65)	3.25	8 min 01 sec
LUBM1000 (133.61)	61.25	7 hr 0 min
LUMB8000 (1,068.00)	**521.7**	**56 hr 42 min**

Slika 3.2 Performanse rezoniranja nad LUBM ontologijama

Riječ je o tri testne ontologije koje sadrže od 6.65 do 1068 milijuna trojki – vrijeme potrebno za rezoniranje je ovisno o broju trojki, kao i o kompleksnosti pravila, odnosno o broju novonastalih trojki rezoniranjem (stupac *Triples Inferred*).

I sami smo za potrebe referata napravili jednostavan eksperiment koristeći Oracle 11g bazu. Deset puta smo odradili spajanje (*inner join*) četiri tablice, gdje je svaka tablica reda veličine milijun redaka (prije svakog spajanja smo ispraznili *buffer cache*). Zatim smo dotične tablice zapisali u obliku trojki u ontologiju. Koristeći SEM_MATCH funkciju odradili smo i identično spajanje virtualnih tablica iz ontologije, deset puta. Upiti nad ontologijom trajali su u prosjeku za red veličine duže nego upiti nad relacijskim tablicama.

Brzine rezoniranja i baratanja s ontologijama općenito predstavljaju problem, a optimizacija semantičkih upita u Oracle bazi mogla bi biti tema nekog budućeg rada.

4. ZAKLJUČAK

Ontologije su formalizam predstavljanja znanja u računalnim sustavima. Opisujući koncepte i odnose među njima, ontologije podacima dodjeljuju kontekst u kojem su smješteni te su zamišljene kao sredstvo jednostavnog dijeljenja informacija između strojeva i ljudi. Rezoniranje pak omogućuje otkrivanje novih informacija na temelju postojećih odnosa zapisanih u ontologijama.

Međutim, u primjeni ontologija postoje problemi - razvoj i održavanje zahtijevaju dodatan angažman; ne postoji uniformno pravilo koje bi propisalo na koji način izgraditi ontologiju te često postoji više načina kojima se može opisati određeno poslovno područje, a da niti jedan u potpunosti ne odgovara svakoj primjeni; formalizmi za preslikavanje relacijskih baza podataka u ontologije nisu

jednostavni i upitno je može li se preslikavanje u potpunosti automatizirati; komplicirano je predvidjeti utjecaj izmjene postojećih i dodavanja novih pravila u ontologiju prije rezoniranja; …

Kako god, od ontologija se u budućnosti očekuje puno, u prvom redu da budu nositelji nove generacije internetskih usluga, Semantičkog weba. Stoga i Oracle nudi podršku za rad s ontologijama. U radu su opisane tehnologije koje se koriste za semantičku pohranu podataka; predstavljeni su različiti modeli podataka (RDF, RDFS, OWL) i njihove mogućnosti za opisivanje poslovnih domena; skicirane su semantičke mogućnosti Oracle baze koje uključuju i kombiniranje semantičkih podataka s podacima zapisanim u relacijskim tablicama.

Kroz poslovni primjer iz bankarske industrije prezentirano je kreiranje klasa, kao i nasljeđivanje, instanciranje klasa, definiranje poslovnih pravila te rezoniranje.

LITERATURA

[1] Oracle Database Semantic Technologies Developer's Guide 11g Release 1 (11.1), [Online], Available:http://docs.oracle.com/cd/B28359_01/appdev.111/b28397/sdo_rdf_concepts.htm#CIHECGII

[2] Oracle database semantic technologies overview, [Online], Available:http://docs.oracle.com/cd/E11882_01/appdev.112/e25609/sdo_rdf_concepts.htm

[3] Introduction to Oracle Objects, [Online], Available:http://docs.oracle.com/cd/B19306_01/appdev.102/b14260/adobjint.htm#i457391

[4] RDF Primer, [Online], Available: http://www.w3.org/TR/rdf-primer/

[5] OWL 2 Web Ontology Language Primer (Second Edition), [Online], Available: http://www.w3.org/TR/owl2-primer/

[6] Reasoning in ontologies, [Online], Available: http://www.obitko.com/tutorials/ontologies-semantic-web/reasoning.html

[7] Implementing an Inference Engine for RDFS/OWL Constructs and User-Defined Rules in Oracle, [Online], Available: http://www.oracle.com/technetwork/database/options/semantic-tech/icde-2008-inf-engine-130436.pdf

[8] The Semantic Web and Agent Technologies Lab, [Online], Available: http://swat.cse.lehigh.edu/index.html

[9] Dr. Waralak V. Siricharoen, "Ontology Modeling and Object Modeling in Software Engineering", in International Journal of Software Engineering and its Applications, Vol. 3. No. 1., January, 2011

[10] Chokri Ben Necib, Johann Christoph Freytag, "Ontology Based Query Processing in Database Management Systems", in Proceedings of CoopIS/DOA/ODBASE, pp.839~857, 2003.

[11] Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart, "Ontologies - Querying Data through Ontologies", 2011 [Online]. Available: http://webdam.inria.fr/Jorge/files/slquery-onto.pdf

[12] M. Prcela, “Predstavljanje ontologija na Web-u”, Institut Ruđer Bošković, 2008

[13] Vedran Podobnik, “Višeagentski sustav za pružanje telekomunikacijskih usluga zasnovan na profilima korisnika”, Faculty of Electrical Engineering and Computing, University of Zagreb, PhD thesis, Zagreb, 2010