Моника Шопова, спец. Софтуерно инженерство, ф.н. 61602

monika.shopova@gmail.com

dive in SPARQL

Query language for getting information from RDF graphs

Съдържание

[1.Въведение в семантичен уеб 1](#_Toc409783884)

[1.1 Какво точно е “Семантичен уеб”? 1](#_Toc409783885)

[1.2URLs, URI адреси, и пространства от имена 2](#_Toc409783886)

[2. Resource Description Framework (RDF) 3](#_Toc409783887)

[2.1 Съхраняване на RDF във файлове 3](#_Toc409783888)

[2.2 Съхраняване на RDF в бази 4](#_Toc409783889)

[3. Въведение в SPARQL 4](#_Toc409783890)

[4. RDF Dataset 6](#_Toc409783891)

[4.1 Примери за RDF Datasets 6](#_Toc409783892)

[5. Видове заявки 7](#_Toc409783893)

[5.1 SELECT 7](#_Toc409783894)

[5.2 CONSTRUCT 8](#_Toc409783895)

[5.3 ASK 9](#_Toc409783896)

[5.4 DESCRIBE 10](#_Toc409783897)

[6. Типове данни 10](#_Toc409783898)

[7. Функции 10](#_Toc409783899)

[7.1 bound 11](#_Toc409783900)

[7.2 IF 11](#_Toc409783901)

[7.3 NOT EXIST И EXISTS 11](#_Toc409783902)

[7.4 logical-or 11](#_Toc409783903)

[7.5 logical-and 11](#_Toc409783904)

# 1.Въведение в семантичен уеб

## 1.1 Какво точно е “Семантичен уеб”?

Може да определим семантичната мрежа като набор от стандарти и най-добри практики за споделяне на данни и семантиката на тези данни чрез Интернет за ползване от приложения.

Semantic Web е надстройка над съществуващата WWW, която има за цел да направи информацията в мрежата, по-понятна за компютрит

За създаване на разбираем от компютрите език, се използва форматът [RDF](http://bg.wikipedia.org/wiki/Resource_Description_Framework) (Resource Description Framework), който е на основа на синтаксиса на [XML](http://bg.wikipedia.org/wiki/XML) и използва идентификатори [URI](http://bg.wikipedia.org/w/index.php?title=URI&action=edit&redlink=1) за обозначение на ресурсите.

Семантичната мрежа е изградена според W3C стандарти: моделът на данни RDF, езика за заявки SPARQL, и RDF Schema и OWL стандарти за съхранение на речници и онтологии. Един продукт или проект може да се справи с семантика, но ако не използва тези стандарти, не може да се свърже и да е част от семантичната мрежа ( така както една хипертекст система от 1985 може да води към страница на World Wide Web, без да използва HTML или HTTP стандарти) .

Идеята на "семантика" често се определя като "значението на думите." Принципите на свързаните данни и свързаните стандарти правят по-лесно споделянето на данни. Използването на URI адреси може да осигури малко семантика чрез предоставяне на контекста на термин. Например, дори и да не знаем за какво "sh98003588 # концепция" се отнася, можем да видим от URI http://id.loc.gov/authorities/sh98003588#concept, че тя идва от библиотеката на Американския Конгрес. Съхраняването на пълно значението на думите, така че компютрите да могат да "разбират" тези значения, може да изисква прекалено много от сегашните компютри, но W3C Web Ontology Language (известен също като OWL) вече ни позволява да съхраняваме ценни парчета със семантично значение, така че можем да получим повече от нашите данни.

## 1.2URLs, URI адреси, и пространства от имена

Когато Бърнърс-Лий е изобретил интернет, заедно с написването на първия уеб сървъра и браузъра, той развива спецификации за три неща, така че всички сървъри и браузъри могат да работят заедно:

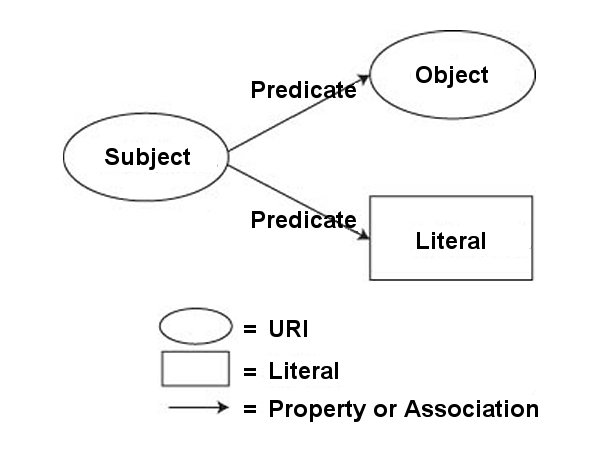
  • Един от начините да представляват структурата на документа, така че браузъра ще знае кои части на документа са параграфи, кои са хедъри, кои са линкове, и така нататък. Тази спецификация е Hypertext Markup Language или HTML.

• Начин за клиентски програми, като например уеб браузъри и сървъри, да комуникират един с друг. Протокол за трансфер на хипертекст, или HTTP, се състои от няколко кратки команди и трицифрени кодове, които по същество позволяват на клиентска програма, като например уеб браузър, да казва неща като: "Хей www.learningsparql.com сървър, изпрати ми файла index.html от директорията с ресурси! "Те също така предоставят на сървъра каже "ОК, заповядай! " или "За съжаление, аз не знам за този ресурс.“ .

  • Компактен начин за клиента да определи кои ресурс го иска - например, името на файла, директорията, в която се съхранява, и сървъра, който има тази файлова система. Можете да я наречете уеб адрес, или бихте могли да го наречете указател на ресурс. Berners Lee нарича сървър-директория–име на ресурса комбинацията, която клиентът изпраща с помощта на специален интернет протокол (например, http://www.learningsparql.com/ resources / index.html) Uniform Resource Locator, или URL.

# 2. Resource Description Framework (RDF)

• Това е модел на данните в която основната единица на информация е известна като triple (наредена тройка).



subject (resource identifier) predicate (property name) object (property value)

richard homeTel (229) 276-5135

cindy email cindym@gmail.com

  • Наредената тройка се състои от субект, предикат, и обект. Може също така да си ги представим като идентификатор на ресурс, име на атрибут или свойство, както и стойността на атрибута или свойството.

  • За да премахнем всички двусмислия от информацията, посочена от дадена тройка, субекта и предиката от тройката трябва да бъдат URI адреси. (Може да използваме префикси на мястото на URI адреси.)

## 2.1 Съхраняване на RDF във файлове

Техническият термин за запазване на RDF като низ от байтове, които могат да бъдат записани на диск е сериализация. Ние използваме този термин вместо "файлове", има операционни системи, които не използват термина "файл" за име на колекция от запаметени данни, но на практика, всички RDF serializations до този момент са текстови файлове, които се използват различни синтаксиса за представяне на тройки. Форматът на сериализация, който се използва най-често, се нарича Turtle.

## 2.2 Съхраняване на RDF в бази

Ако е необходимо да съхранявате много голям брой тройки, съхраняването им в Turtle или RDF / XML в един голям текстов файл може да не е най-добрият вариант, тъй като една система, която индексира данните и решава кои данни да се зареди в паметта ( това е Database Management

System - система за управление на бази данни) може да бъде по-ефективна. Има начини за съхраняване на RDF в релационна база данни за управление, като MySQL или Oracle, но най-добрият начин е програма за управление на бази данни, оптимизирана за RDF тройки. Ние наричаме това triplestore, и има търговски triplestores с отворен код.

# 3. Въведение в SPARQL

Какво е SPARQL? Името е акроним на SPARQL протокол и RDF Query Language, който е описан от набор от спецификации от W3C.

Както можете да кажете от "RQL", SPARQL е проектиран да изпълнява заявка над RDF данни, но не е ограничен до заявки данни, съхранявани в един от форматите на RDF. Има търговски приложения, с отворен код, с които могат да се обработват релационни данни, XML, JSON, електронни таблици и други формати като RDF, така че можете да изпращате SPARQL заявки към данните в тези формати, или към комбинации от тези формати, което е един от най-мощните аспекти на комбинацията SPARQL / RDF. "Протокол" частта от името SPARQL се отнася до правилата за това как програмата клиент и обработващия SPARQL заявки сървър обменят SPARQL заявки и резултати. Тези правила са посочени в отделен документ от документа за спецификация на заявки и са най-вече проблем за SPARQL процесорни разработчици.

Примерни RDF наредени тройки:

subject (resource identifier) predicate (property name) object (property value)

richard homeTel (229) 276-5135

cindy email [cindym@gmail.com](mailto:cindym@gmail.com)

Следния файл показва горните наредени тройки, описани в RDF Turtle формат. Файлът съдържа данни тип „адресна книга“, използвайки твърдения от сорта на “richard’s homeTel value is (229) 276-5135” и “cindy’s email value is [cindym@gmail.com](mailto:cindym@gmail.com).” В RDF може да има няколко стойности за един атрибут, както във файла Craig има два email адреса:

@prefix ab: <http://learningsparql.com/ns/addressbook#> .

ab:richard ab:homeTel "(229) 276-5135" .

ab:richard ab:email "richard49@hotmail.com" .

ab:cindy ab:homeTel "(245) 646-5488" .

ab:cindy ab:email "cindym@gmail.com" .

ab:craig ab:homeTel "(194) 966-1505" .

ab:craig ab:email "craigellis@yahoo.com" .

ab:craig ab:email "c.ellis@usairwaysgroup.com" .

Като изречение, написано на английски, Turtle (и SPARQL) тройките обикновено завършват с точка. Разстоянията, които виждате, преди точките по-горе, не са необходими, но са често срещана практика да станат данните по-лесно четими.

Една SPARQL заявка обикновено казва: "Искам тези парчета от информация от подгрупата на данни, които отговарят на тези условия." Описваме условията с тройни модели, които са подобни на RDF тройки, но може да включват променливи, за да добавят гъвкавост в начина, по които те работят с данните.

SELECT ?craigEmail

WHERE

{

ab:craig ab:email ?craigEmail .

}

Тази заявка пита за всички ab:email стойности, асоциирани с ресура ab:craig. На чист български, заявката иска всички email-и на Craig.

• Разнообразие от SPARQL процесори са на разположение, за да обработват заявки и към локални, и към отдалечени данни. SPARQL processor е програма, с която се изпълнява заявка SPARQL към набор от данни и да ти казва резултата. За заявки спрямо файл с данни на твърдия диск, има безплатна, Java-базирана програма ARQ . ARQ е част от Apache Jena Framework.

• Не е необходимо да се използват префикси в заявката, но те могат да направят заявка по-компактна и по-лесно да се четима от такава, която използва пълните URI адреси. Когато все пак използвате пълния URI адрес, хубаво е да се загради в ъглови скоби, за да покаже на процесора, че това е URI.

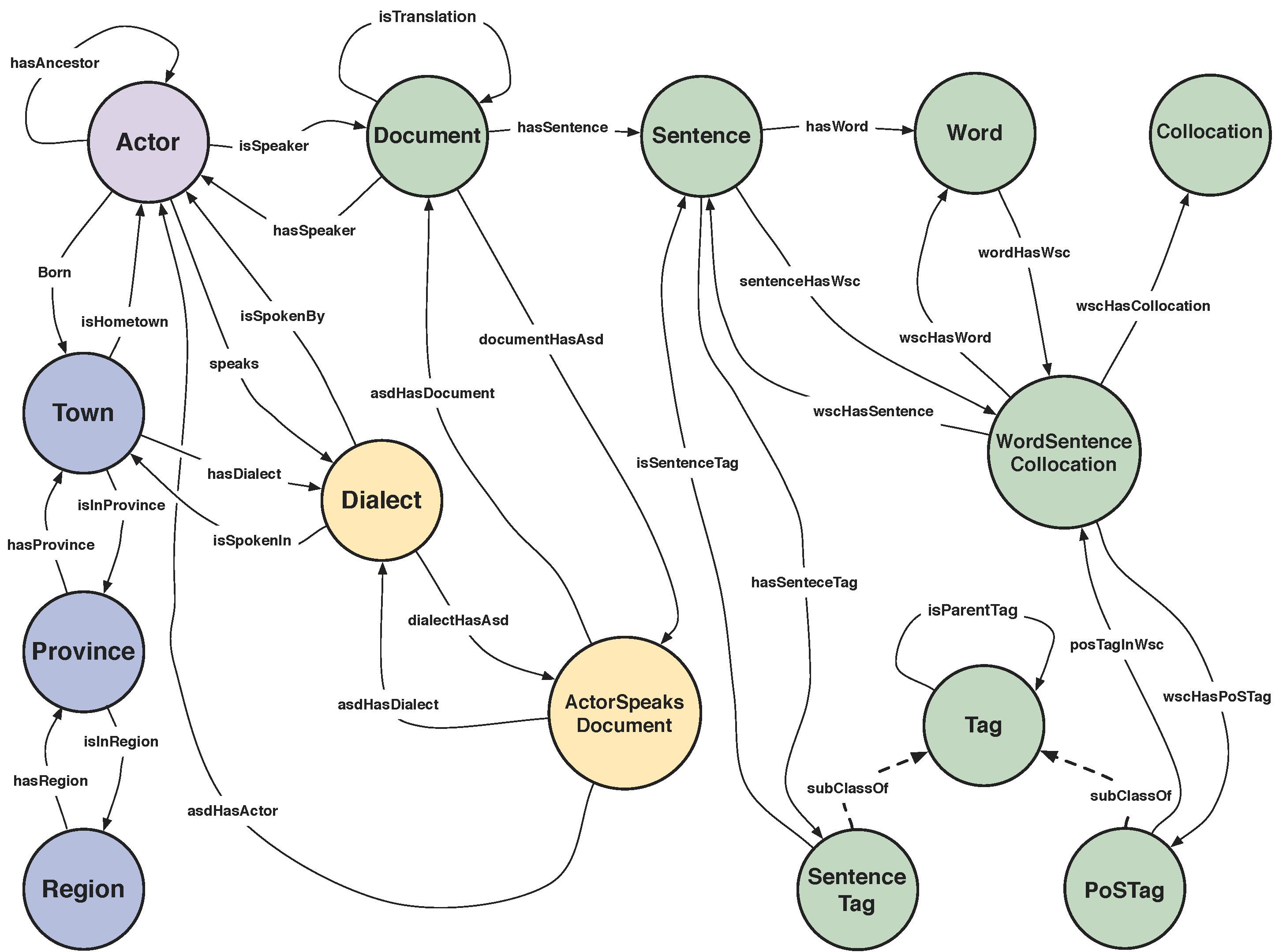
• Празните места не влияят на синтаксиса на SPARQL.

• Подобно на XML, RDF позволява да определите префикс, който да представлява пространство от имена URI, но за разлика от XML, RDF позволява да използвате пълните URI адреси от списък с имената вместо представки. След като декларирането, че представката v: се отнася за пространството от имена http://www.w3.org/2006/vcard/, на RDF набор от данни може да се каже, че Бърнърс-Лий има v:title "Director", но може да се каже, че той има <http://www.w3.org/2006/vcard/title> "Director", като се използва цялата пространство от имена URI вместо префикса.

# 4. RDF Dataset

RDF Dataset (mодел на данните на RDF) изразява информация като графи, състоящи се от наредени тройки от субект, предикат обект. Много RDF data stores съдържат по няколко RDF графи и записват информация за всеки граф, което позволява на приложението да направи заявки, които включват информация от повече от един граф.

SPARQL заявка се изпълнява над RDF Dataset, което представлява колекция от графи. Един RDF Dataset включва един граф, граф по подразбиране, който няма име, и нула или повече графи с имена, където всеки именуван граф се идентифицира с IRI.



## 4.1 Примери за RDF Datasets

# **Default graph**

@prefix dc: <http://purl.org/dc/elements/1.1/> .

<http://example.org/bob> dc:publisher "Bob" .

<http://example.org/alice> dc:publisher "Alice" .

# **Named graph: http://example.org/bob**

@prefix foaf: <http://xmlns.com/foaf/0.1/> .

\_:a foaf:name "Bob" .

\_:a foaf:mbox <mailto:bob@oldcorp.example.org> .

5. Видове заявки

## 5.1 SELECT

Както и в SQL, най-често използвания глагол в SPARQL е SELECT. Тя ви позволява да вземете данни от колекция независимо дали искате единен телефонен номер или списък с имена, фамилни имена и телефонни номера на служителите, наети след 1 януари сортирани по фамилно име. SPARQL процесори като ARQ обикновено показват резултат от SELECT заявка като таблица на редове и колони, с колона за всяка избрана име на променлива, и SPARQL APIs зареждат стойностите в подходяща структура от данни за езика за програмиране, който е в основата на това API. В SPARQL, SELECT е известен като форма на заявка.

Пример:

Данни:

@prefix dc: <http://purl.org/dc/elements/1.1/> .

@prefix : <http://example.org/book/> .

@prefix ns: <http://example.org/ns#> .

:book1 dc:title "SPARQL Tutorial" .

:book1 ns:price 42 .

:book1 ns:discount 0.2 .

:book2 dc:title "The Semantic Web" .

:book2 ns:price 23 .

:book2 ns:discount 0.25 .

Заявка:

PREFIX dc: <http://purl.org/dc/elements/1.1/>

PREFIX ns: <http://example.org/ns#>

SELECT ?title (?p\*(1-?discount) AS ?price)

{ ?x ns:price ?p .

?x dc:title ?title .

?x ns:discount ?discount

}

Резултати:

|  |  |
| --- | --- |
| **title** | **price** |
| "The Semantic Web" | 17.25 |
| "SPARQL Tutorial" | 33.6 |

## 5.2 CONSTRUCT

CONSTRUCT връща наредени тройки. Може да вземете тройки директно от източника, без да ги променяте, или може да вземете стойности и да ги използвате, за да се създадат нови тройки. Това ви позволява да копирате, създавате и конвертирате RDF данни, и е по-лесно да се идентифицират данни, които не отговарят на специфичните бизнес правила.

#### Достъп до графи в RDF Dataset

Използвайки CONSTRUCT, е възможно да се извлекат части или цели графи от определен RDF dataset (набор от данни). В първия пример връща граф (ако е в набора от данни) с IRI label

http: //example.org/aGraph; В противен случай, той връща празен граф.

CONSTRUCT { ?s ?p ?o } WHERE { GRAPH <http://example.org/aGraph> { ?s ?p ?o } . }

Достъпът до граф може да бъде обвързан с друга информация. Например, ако графът по подразбиране съдържа метаданни за назованите графи в набора от данни, със заявка, подобна на следната, може да се извлече един граф на базата на информация за името на графа:

PREFIX dc: <http://purl.org/dc/elements/1.1/>

PREFIX app: <http://example.org/ns#>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

CONSTRUCT { ?s ?p ?o } WHERE

{

GRAPH ?g { ?s ?p ?o } .

?g dc:publisher <http://www.w3.org/> .

?g dc:date ?date .

FILTER ( app:customDate(?date) > "2005-02-28T00:00:00Z"^^xsd:dateTime ) .

}

където app:customDate идентифицира допълнителна функция, която представя date format като  xsd:dateTime .

## 5.3 ASK

Заявката ASK пита процесор дали даден граф модел описва набор от тройки от определен набор от данни или не, и процесорът връща булева вярно или невярно. Това е удобно за изразяване на бизнес правила относно условията, които трябва или не трябва да важат за данните ви.

@prefix foaf: <http://xmlns.com/foaf/0.1/> .

\_:a foaf:name "Alice" .

\_:a foaf:homepage <http://work.example.org/alice/> .

\_:b foaf:name "Bob" .

\_:b foaf:mbox <mailto:bob@work.example> .

PREFIX foaf: <http://xmlns.com/foaf/0.1/>

ASK { ?x foaf:name "Alice" }

true

## 5.4 DESCRIBE

DESCRIBE пита за наредени тройки, които описват даден ресурс. Спецификацията SPARQL оставя на процесора за заявки да реши кои наредени тройки да изпрати обратно като описание на име ресурс. Това е довело до непоследователни реализации на DESCRIBE заявки, така че тази заявка не е много популярнa, но си струва да се запознаем с нея.

DESCRIBE <http://example.org/>

# 6. Типове данни

Различните езици за програмиране, езици за маркиране, както и езици за заявки предлагат различни набори от типове данни, от които да избирате. Според спецификацията на SPARQL, основните типове данни са следните :

• xsd:integer

• xsd:decimal

• xsd:float

• xsd:double

• xsd:string

• xsd:boolean

• xsd:dateTime

# 7. Функции

## 7.1 bound

xsd:boolean **BOUND** (variable var)

Връща истина, ако var е обвързан със стойност. Иначе връща false. Променливи със стойността NaN или INF се считат за обвързани.

## 7.2 IF

rdfTerm IF (*expression1*, *expression2*, *expression3*)

Функцията IF () приема три аргумента. Ако първият е с резултат true, функцията връща стойността на втория аргумент; В противен случай, тя се връща на третия.

## 7.3 NOT EXIST И EXISTS

Има филтър оператор EXIST, който взима един graph pattern. EXISTS връща вярно / невярно зависимост от това дали моделът отговаря на набор от данни. NOT EXISTS формата всъщност представлява отрицание на EXISTS: intofn: not(EXISTS {...}).

xsd:boolean NOT EXISTS { pattern }

Връща false ако pattern съвпада. Иначе връща true.

NOT EXISTS { pattern } е еквивалентно на fn:not(EXISTS { pattern }).

xsd:boolean EXISTS { pattern }

Връща true ако pattern съвпада. Иначе връща false.

## 7.4 logical-or

xsd:boolean xsd:boolean left **||** xsd:boolean right

Връща логическо „или“ от ляво и от дясно.

## 7.5 logical-and

xsd:boolean xsd:boolean left **&&** xsd:boolean right

Връща логическо „и“ от ляво и от дясно.