SPARQL УПИТНИ ЈЕЗИК

Јелена Јовановић

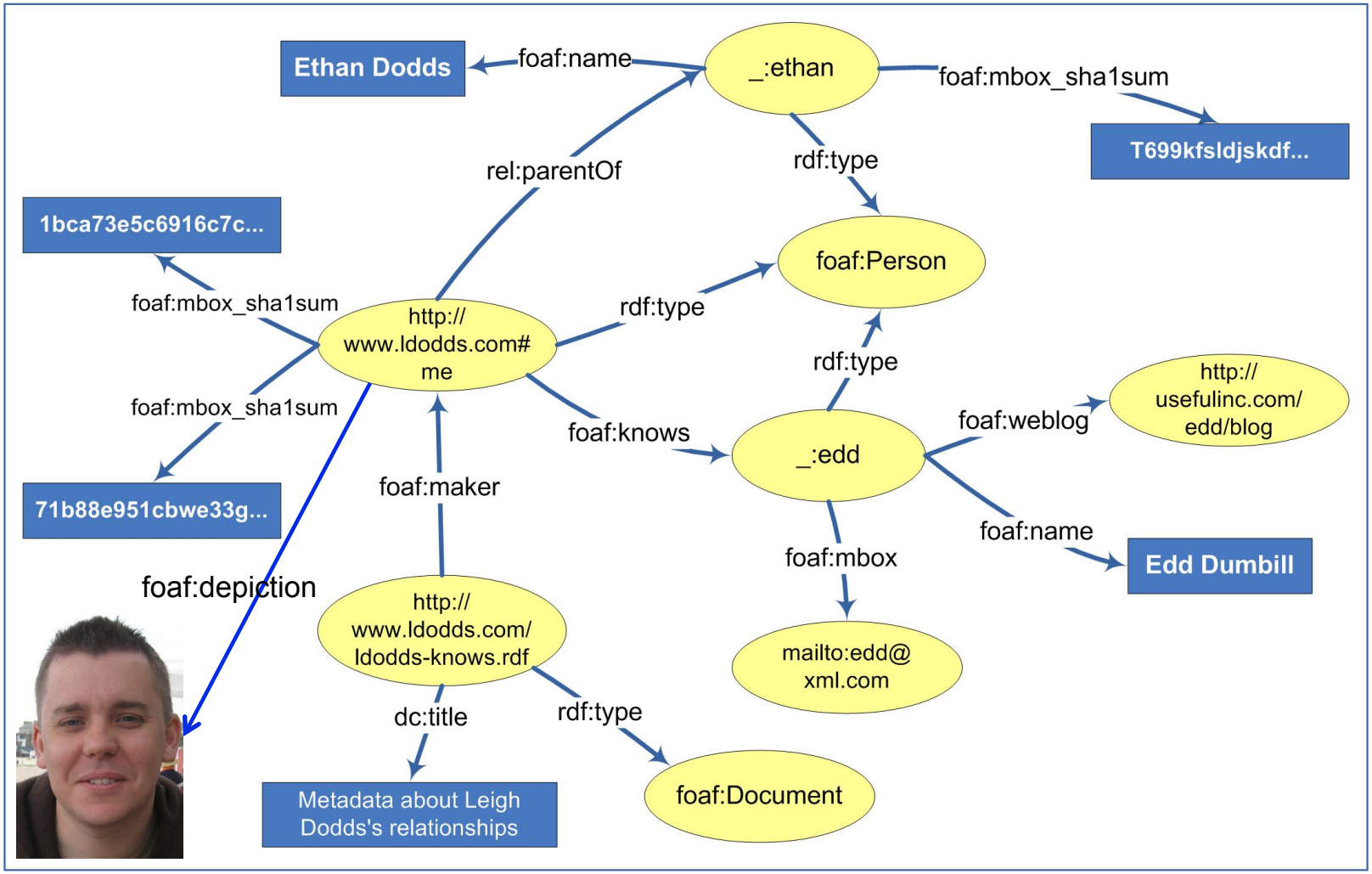
Email: jeljov@gmail.com
Web: http://jelenajovanovic.net

SPARQL упитни језик

- W3C стандард за упите над RDF графовима
- Користи се за упите не само над подацима оригинално датим у RDF формату, већ било којим подацима који се могу мапирати у RDF
- Мапирање се може реализовати коришћењем
  - (W3C) стандарда попут R2RML који омогућује трансформацију релационих података у RDF
  - различитих алата попут оних излистаних на:

    http://www.w3.org/wiki/ConverterToRdf

Почнимо са једним примером



Графички приказ једног малог сегмента RDF графа:
http://www.ldodds.com/ldodds-knows.rdf

Задатак 1: пронаћи имена свих особа које се помињу у датом RDF графу

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?name
FROM <http://www.ldodds.com/ldodds-knows.rdf>
WHERE
{
  ?x rdf:type foaf:Person.
  ?x foaf:name ?name.
}
```

Трипл патерн

Граф патерн

Основна структура SPARQL упита

- PREFIX
  - SPARQL еквивалент декларацији XML namespace-а
- SELECT
  - Као и у SQL упиту, користи се за дефинисање података које упит треба да врати
- FROM
  - Идентификује податке над којима ће се упит извршити
  - Типично се задаје програмски, у време извршења упита
- WHERE
  - Дефинише део RDF графа на који се упит односи

Напомене о SPARQL синтакси

- Варијабле се означавају префиксима "? " или "$“
  - Потпуно је све једно који ће се од ових префикса користити
- Тзв. ‘празни чворови’ (blank nodes) се представљају:
  - у форми лабеле, нпр., "_:abc", или
  - у скраћеној форми: "[]"
- Тачка (.) одваја трипл патерне
- Тачка-зарез (;) одваја трипл патерне са заједничким субјектом

О граф патернима

- Граф патерн…
  - …је колекција трипл патерна
  - …идентификује облик (RDF) графа над којим треба извршити упит
- У оквиру једног граф патерна

  једна иста варијабла мора имати исту вредност,

  без обзира на ком месту се појављује

О граф патернима

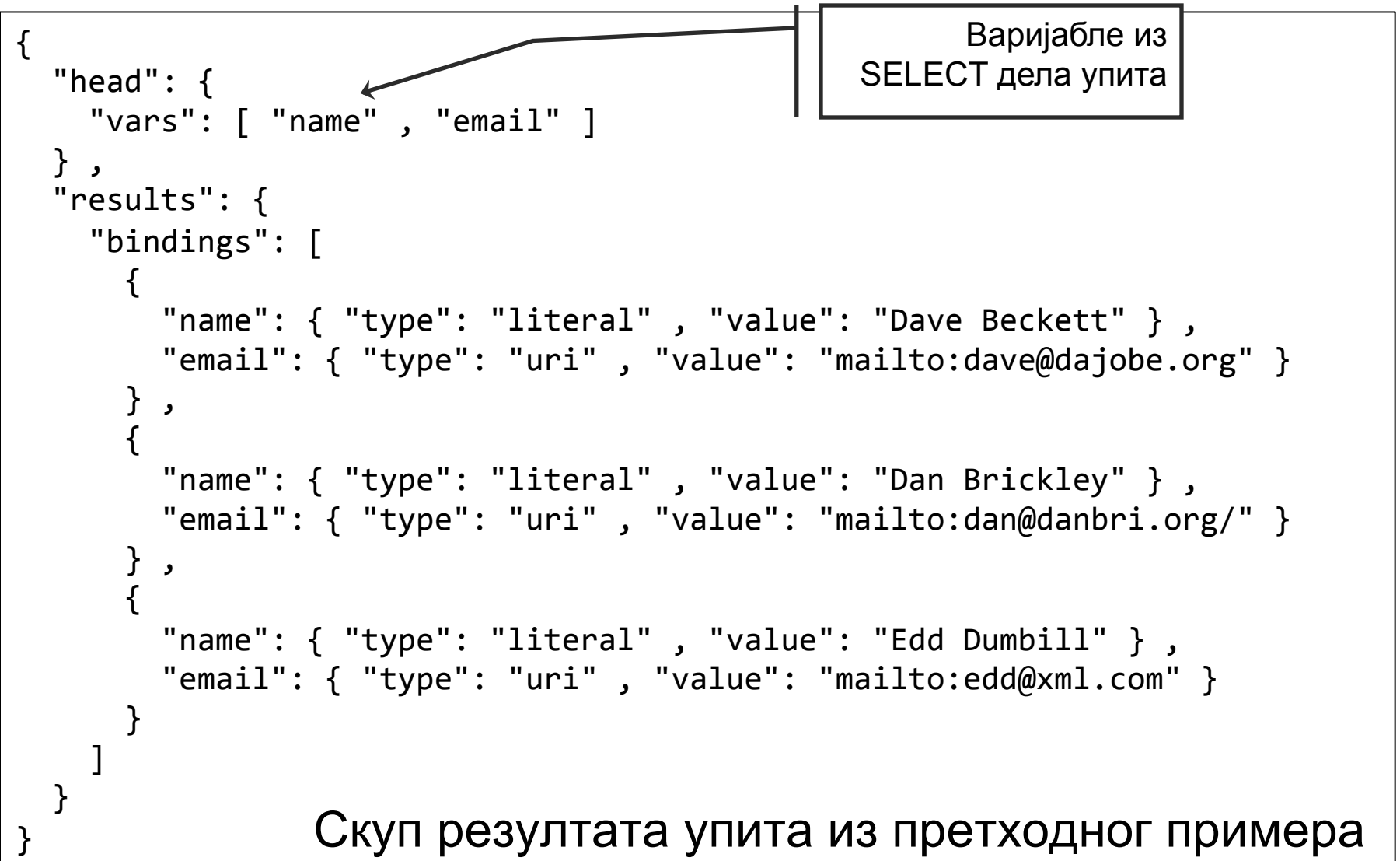
- У SPARQL упиту се не може селектовати нека варијабла (тј. наћи у SELECT делу) уколико се она не појављује у граф патерну (тј., у WHERE делу упита)

- Разлог:
  - процесор SPARQL упита НЕМА шему података на основу које би могао одредити типове и својства ресурса ;
  - шему одређује граф патерн тј. WHERE део упита

Задатак 2: пронаћи имена и email адресе особа које аутор документа познаје

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?name ?email
FROM <http://www.ldodds.com/ldodds-knows.rdf>
WHERE
{
   ?doc rdf:type foaf:Document ;
            foaf:maker ?author .
   ?author foaf:knows ?someone .
   ?someone foaf:name ?name ; foaf:mbox ?email .
 }
```

Резултати SELECT упита




```
{
  "head": {
    "vars": [ "name" , "email" ]
  } ,
  "results": {
    "bindings": [
      {
        "name": { "type": "literal" , "value": "Dave Beckett" } ,
        "email": { "type": "uri" , "value": "mailto:dave@dajobe.org" }
      } ,
      {
        "name": { "type": "literal" , "value": "Dan Brickley" } ,
        "email": { "type": "uri" , "value": "mailto:dan@danbri.org/" }
      } ,
      {
        "name": { "type": "literal" , "value": "Edd Dumbill" } ,
        "email": { "type": "uri" , "value": "mailto:edd@xml.com" }
      }
    ]
  }
}
```

Скуп резултата упита из претходног примера

Опционо мечирање

- RDF подаци су најчешће *полу-структурирани*
  - То значи да два ресурса истог типа могу бити описана различитим скупом атрибута
  - На пример,
    - FOAF опис особе може садржати само њену e-mail адресу;
    - алтернативно, може садржати пуно име, надимак, URLs фотографија, и сл.
- Механизам *опционог мечирања* SPARQL-а омогућује да се ради са оваквом хетерогеношћу података

Задатак 3: Пронаћи све особе које аутор документа познаје, као и њихове блогове *уколико су познати*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?person ?blog
FROM <http://www.ldodds.com/ldodds-knows.rdf>
WHERE
{
   ?doc rdf:type foaf:Document; foaf:maker ?author .
   ?author foaf:knows ?person.
   OPTIONAL { ?person foaf:weblog ?blog. }
 }
```

Блок OPTIONAL

- Блок OPTIONAL може садржати граф патерн произвољне сложености, не само један трипл патерн (као у претходном примеру)
- Комплетан граф патерн садржан у OPTIONAL блоку мора се мечирати са подацима како би био део резултата упита

Блок OPTIONAL

- Уколико упит садржи више OPTIONAL блокова
  - Они стоје независно један од другог
  - Сваки од блокова може бити изузет или присутан у решењу (независно од осталих блокова)
- OPTIONAL блокови могу бити и угњеждени
  - унутрашњи OPTIONAL блок се разматра само уколико се патерни спољног OPTIONAL блока могу мечирати (подацима из задатог извора)

Задатак 4: Пронаћи све особе које аутор документа познаје, као и њихове блогове и e-mail адресе, *уколико су познати*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?person ?email ?blog
FROM <http://www.ldodds.com/ldodds-knows.rdf>
WHERE
{
   ?doc rdf:type foaf:Document; foaf:maker ?author .
   ?author foaf:knows ?person.
   OPTIONAL { ?person foaf:mbox_sha1sum ?email. }
   OPTIONAL { ?person foaf:weblog ?blog . }
 }
```

Алтернативно мечирање

- Хајде да претпоставимо…
  - *foaf:knows* и *rel:hasMet* својства омогућују представљање доста сличних релација међу људима
  - Заинтересовани смо за све особе које аутор документа или познаје (*foaf:knows*) или их је некад упознао (*rel:hasMet*)
- У оваквим ситуацијама, користи се *алтернативно мечирање* како би се добила вредност било ког од расположивих својстава

Задатак 5: Пронаћи имена свих особа које аутор документа или познаје или је (некад) упознао

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rel: <http://purl.org/vocab/relationship/>
SELECT ?name
FROM <http://www.ldodds.com/ldodds-knows.rdf>
WHERE
{
   ?doc foaf:maker ?author .
   { ?author foaf:knows [ foaf:name ?name] . }
            UNION
    {?author rel:hasMet [ foaf:name ?name] . }
 }
```

UNION

- Насупрот OPTIONAL граф патернима, у случају коришћења UNION блока, *макар једна* од алтернатива мора бити мечирана;
- Уколико су обе гране UNION блока мечиране, два решења ће бити генерисана.

DISTINCT

- У скупу резултата претходног упита нека имена се појављују два пута
- Додавањем DISTINCT кључне речи иза SELECT, искључује се вишеструко појављивање исте вредности из скупа резултата
  - Као што је то случај и у SQL-у

Задатак 5а: Пронаћи имена свих особа које аутор документа или познаје или је (некад) упознао (без понављања имена)

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rel: <http://purl.org/vocab/relationship/>
SELECT DISTINCT ?name
FROM <http://www.ldodds.com/ldodds-knows.rdf>
WHERE
{
   ?doc foaf:maker ?author .
    { ?author foaf:knows [ foaf:name ?name] . }
            UNION
    {?author rel:hasMet [ foaf:name ?name] . }
 }
```

ORDER BY наредба

- Указује да би резултати требало да буду сортирани према вредности задатог својства
- Може садржати једну или више варијабли, у ком случају се сортирање врши према свим наведеним варијаблама
- Подразумевани смер сортирања је растући
  - Ово се може и експлицитно дефинисати коришћењем кључних речи DESC (опадајући) и ASC (растући)

Задатак 5б: Пронаћи имена свих особа које аутор документа или познаје или је (некад) упознао; сортирати имена у опадајућем редоследу

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rel: <http://purl.org/vocab/relationship/>
SELECT DISTINCT ?name
FROM <http://www.ldodds.com/ldodds-knows.rdf>
WHERE
{
   ?doc foaf:maker ?author .
    { ?author foaf:knows [ foaf:name ?name] . }
            UNION
    {?author rel:hasMet [ foaf:name ?name] . }
 }
ORDER BY DESC (?name)
```

SPARQL филтери

- SPARQL филтери сужавају скуп решења на основу задатих ограничења
- Изрази којима се дефинишу ограничења могу бити различите врсте, али се морају израчунати у boolean вредност (true или false)
- Наредни примери илуструју неке од могућих облика филтрирања резултата упита

Задатак 6: Пронаћи имена свих људи чији је датум рођења непознат

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX bio: <http://purl.org/vocab/bio/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?name
FROM <http://www.ldodds.com/ldodds-knows.rdf>
WHERE
{
  ?person foaf:name ?name .
   FILTER NOT EXISTS {
            ?person bio:event ?ev .
            ?ev rdf:type bio:Birth ;
                  bio:date ?birthdate. }
}
```

Напомена: Функција NOT EXISTS уведена је у верзији 1.1 SPARQL језика; уколико упит не ради, значи да користите SPARQL engine базиран на старој верзији SPARQL-а

Задатак 7а: Пронаћи имена свих чланова Dodds породице

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
FROM <http://www.ldodds.com/ldodds-knows.rdf>
WHERE
{
  ?person foaf:name ?name
  FILTER regex(?name, "dodds", "i")
}
```

Филтрирање коришћењем регуларних израза
Слично као SQL "LIKE"

Алтернатива:
FILTER strEnds(lcase(?name), “dodds”)

Задатак 7б: Пронаћи имена свих особа које имају Gmail email адресу

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
FROM <http://www.ldodds.com/ldodds-knows.rdf>
WHERE
{
  ?person foaf:name ?name ; foaf:mbox ?mbox
  FILTER regex( str(?mbox), "@gmail\\.com$” )
}
```

Добар туторијал за регуларне изразе: http://regex.bastardsbook.com/

Задатак 8: Пронаћи све рецензије са оценом вишом од 6 чији је аутор особа под именом Jim (филтрирање засновано на вредности елемената)

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rev: <http://www.purl.org/stuff/rev#>

SELECT ?review
FROM  <http://www.cs.umd.edu/~hendler/2003/foaf.rdf>
WHERE {
      ?someone rdf:type foaf:Person;
            foaf:name ?name FILTER regex(?name, "Jim", "i").
      ?someone foaf:made ?review .
      ?review rev:rating ?rating
               FILTER (xsd:decimal(?rating) >= "6"^^xsd:decimal) .
   }
```

SPARQL type casting

Груписање и сумирање података

- GROUP BY омогућује груписање резултата упита по једној или више задатих варијабли или израза
- HAVING омогућује селекцију тј. филтрирање резултата на нивоу групе
- За сумирање резултата расположиве су функције SUM, COUNT, AVG, MIN, MAX и сл. које се примењују над групама података

Задатак 9: Пронаћи произвођаче који производе више од 10 различитих уређаја, и приказати број различитих уређаја које производе

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT ?manufacturer (COUNT(?product) AS ?count)
WHERE {
   ?product rdf:type <http://dbpedia.org/ontology/Device> ;
                dbpedia-owl:manufacturer ?manufacturer .
}
GROUP BY ?manufacturer
HAVING (COUNT(?product) > 10)
```

Друге врсте SPARQL упита

- Поред SELECT упита,
  SPARQL подржава још 3 врсте упита:
  - ASK
  - DESCRIBE
  - CONSTRUCT

Друге врсте SPARQL упита

- ASK упит
  - Намењен провери да ли неки упит уопште има решење
  - Не враћа никакву информацију о самом решењу упита, већ само да ли оно постоји
  - Пример: да ли су Natalie Portman и Scarlett Johansson играле у истом филму

```
PREFIX db: <http://dbpedia.org/ontology/>
ASK  {
    ?movie
            db:starring <http://dbpedia.org/resource/Natalie_Portman> ;
            db:starring <http://dbpedia.org/resource/Scarlett_Johansson> .
}
```

Друге врсте SPARQL упита

- Резултат ASK упита:
  - Могући резултати: true/false
  - JSON формат резултата ASK упита:

```
{
   "head" : { } ,
   "boolean" : true
}
```

Друге врсте SPARQL упита

- DESCRIBE упит
  - **Враћа граф** који садржи све расположиве триплете о ресурсу који је мечиран у оквиру граф патерна (тј. у WHERE делу упита)
  - Пример:

```
PREFIX db: <http://dbpedia.org/ontology/>
DESCRIBE ?movie
WHERE {
    ?movie
        db:starring <http://dbpedia.org/resource/Natalie_Portman> ;
        db:starring <http://dbpedia.org/resource/Scarlett_Johansson> .
}
```

Враћа граф који садржи све расположиве триплете о филму/ филмовима у којима су играле обе глумице.

Друге врсте SPARQL упита

- CONSTRUCT упит
  - Користи се за
    креирање нових RDF графова на основу
    постојећих тј. за трансформацију RDF графова
  - Овај упит је за RDF граф
    исто што и XSLT за XML податке

Задатак 10: Мапирати податке о месту и датуму рођења музичара из DBpedia вокабулара у Schema.org вокабулар

```
PREFIX db-ont: <http://dbpedia.org/ontology/>
PREFIX schema: <http://schema.org/>
CONSTRUCT {
        ?someone  a  schema:Person ;
                schema:birthPlace ?birthplace ;
                schema:birthDate ?birthdate ;
                schema:jobTitle "Musician".
} WHERE {
   ?someone  a  db-ont:MusicalArtist ;
        db-ont:birthDate ?birthdate ;
        db-ont:birthPlace ?birthplace .
 }
```

скраћени облик за rdf:type

Задатак 11: Успоставити нову релацију међу ентитетима

```
PREFIX schema: <http://schema.org/>
PREFIX rel: <http://purl.org/vocab/relationship/>
CONSTRUCT {
        ?child rel:hasAunt ?aunt.
} WHERE {
        ?child  schema:parent  ?parent .
        ?parent schema:parent ?grandparent .
        ?aunt   schema:parent  ?grandparent ;
                schema:gender  ?gender
        FILTER (?parent != ?aunt && regex(?gender, “female”, “i”)) .
}
```

Упит креира граф у коме се између два ентитета (особе) успоставља релација *hasAunt* уколико су задовољени услови дати у WHERE делу упита

Упити над више дистрибуираних извора

- Сви упити које смо видели до сада, извршавали су се над подацима који долазе из једног извора (тј. RDF графа)
- Међутим, упити се могу извршавати и на више извора података
  - Тад говоримо о федеративним упитима (*federated queries*)
  - SPARQL 1.1 уводи кључну реч SERVICE за дефинисање додатних извора података

Задатак 12: пронаћи све познанике Leigh Dodds-a који имају исто презиме као неко од познатих научника

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX db: <http://dbpedia.org/ontology/>
SELECT ?person
FROM <http://www.ldodds.com/ldodds-knows.rdf>
WHERE {
    <http://www.ldodds.com#me> foaf:knows ?person .
    ?person foaf:surname ?surname .
   SERVICE <http://dbpedia.org/sparql> {
          ?someone  a  db:Scientist ;
                    foaf:surname ?surname .
    }
}
```

Јединствени идентификатор (IRI) за Leigh Dodds-a, како је дат у коришћеном извору података (FROM део упита)

Научите SPARQL кроз примере

- Search RDF data with SPARQL
  - http://www-128.ibm.com/developerworks/xml/library/j-sparql/
- SPARQL by Example
  - http://www.cambridgesemantics.com/semantic-university/sparql-by-example
- A detailed SPARQL tutorial
  - http://www.w3.org/2004/Talks/17Dec-sparql/
- SPARQL screencast
  - http://linkeddata.deri.ie/node/58
- Bring existing data to the Semantic Web
  - http://www-128.ibm.com/developerworks/xml/library/x-semweb.html

Научите SPARQL кроз примере

- RDF as self-describing data
  - http://goo.gl/Gdr5LG
- SPARQL at the movies
  - http://www.snee.com/bobdc.blog/2008/11/sparql-at-the-movies.html
- Bart (Simpson) blackboard queries
  - http://goo.gl/aM9mcd ; http://goo.gl/z9qOlH
- Example SPARQL queries over 10+ different RDF datasets
  - http://openuplabs.tso.co.uk/datasets
- SPARQL queries over Europeana repository
  - http://europeana.ontotext.com/sparql

Алати за учење SPARQL-а

- YASGUI – Yet Another SPARQL GUI
  - http://yasgui.laurensrietveld.nl/
- Flint SPARQL Editor
  - http://openuplabs.tso.co.uk/demos/sparqleditor
- SPARQLer - an online SPARQL query tool
  - http://www.sparql.org/sparql.html
- ARQ, a SPARQL processor for Jena framework
  - http://jena.sourceforge.net/ARQ/