ОСНОВЕ **SPARQL** УПИТНОГ ЈЕЗИКА

SPARQL – W3C стандард

- SPARQL обухвата:
  - Спецификацију упитног језика
  - Спецификацију језика за модификацију RDF графа
  - Спецификацију резултата упита
    - дефинише XML, JSON, CSV, и TSV формате за серијализацију резултата упита
  - Спецификацију протокола за приступ подацима
    - за упите над удаљеним RDF базама, или *било ком репозиторијуму* који се може мапирати у RDF модел
  - Спецификацију федеративних упита
    - упити који се извршавају над више извора RDF података
  - …

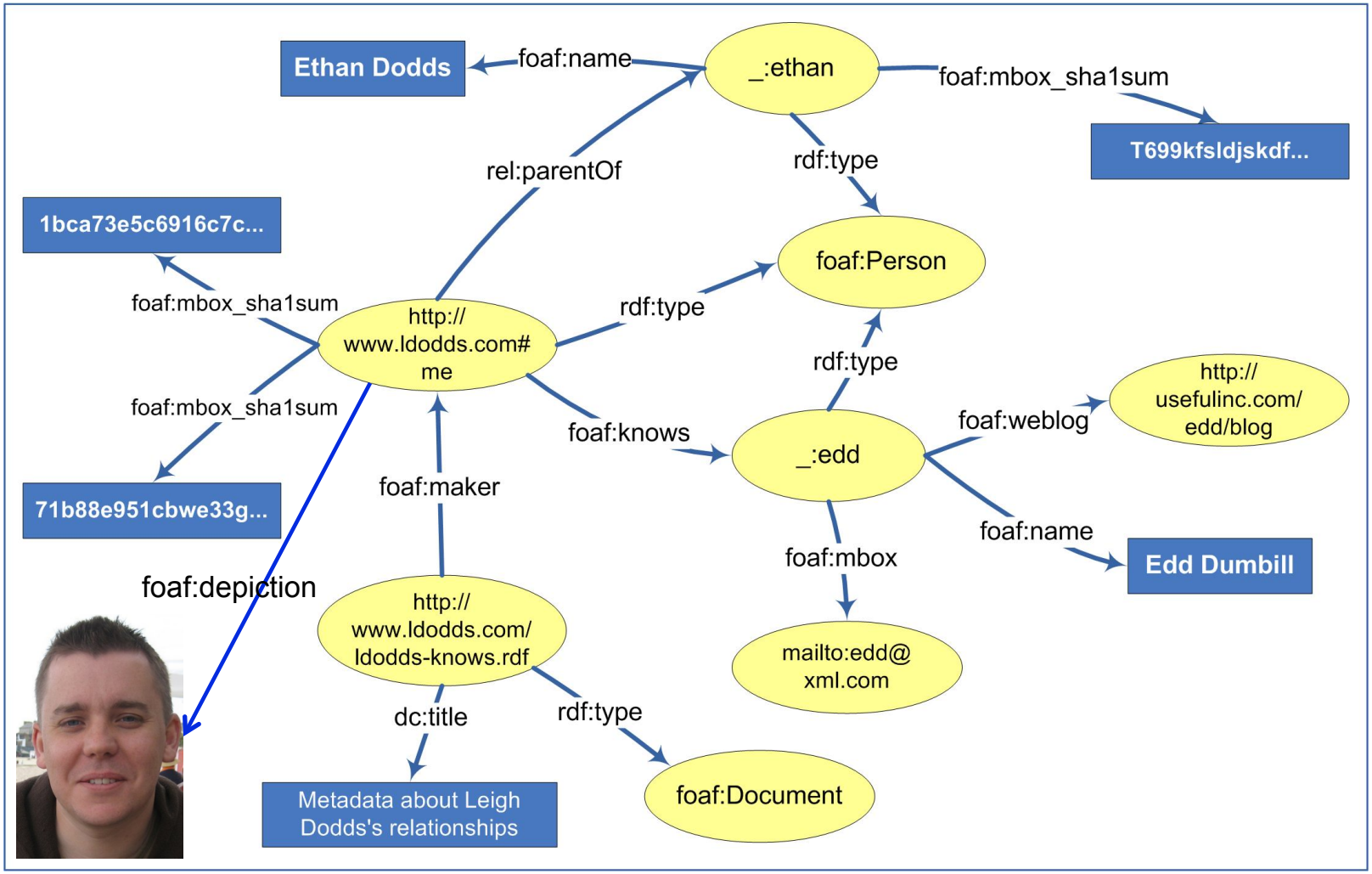
SPARQL – W3C стандард

- SPARQL може да ради са било којим извором података који се може мапирати у RDF
- Мапирање се може реализовати коришћењем
  - (W3C) стандарда попут R2RML (омогућује трансформацију релационих података у RDF)
  - различитих алата попут оних излистаних на:

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

SPARQL упитни језик

Почнимо са једним примером



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

Задатак 1: пронаћи имена свих особа које се помињу у документу

```
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

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

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 .
}
```

Граф патерн

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).

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

Резултати SELECT упита

- Скуп резултата упита из претходног примера је следећег облика:

Варијабле из SELECT дела упита

```
<sparql
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xs="http://www.w3.org/2001/XMLSchema#"
   xmlns="http://www.w3.org/2005/sparql-results#" >
  <head>
   <variable name="name"/>
   <variable name="email"/>
  </head>
  <results ordered="false" distinct="false">
   <result>
     <binding name="name">
       <literal>Edd Dunbill</literal>
     </binding>
     <binding name="email">
       <uri>mailto:edd@xml.com</uri>
     </binding>
   </result>
  </results>
</sparql>
```

Опционо мечирање

- RDF подаци су најчешће *полу-структурирани*
  - То значи да два ресурса истог типа могу имати различит скуп својстава
  - 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 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)

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 .
  OPTIONAL {
            ?person bio:event ?ev .
            ?ev rdf:type bio:Birth ; bio:date ?birthdate .
  }
  FILTER (!bound(?birthdate))
}
```

Функција **bound** враћа вредност true уколико је задатој варијабли придружена вредност; у супротном враћа false.

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

SPARQL 1.1

```
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. }
}
```

У новој верзији SPARQL-а, уместо функције **bound** може се користити NOT EXISTS

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"

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$") )
}
```

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 који 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)
```

Упит се може испробати коришћењем YASGUI online SPARQL едитора

Друге врсте SPARQL упита

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

Twinkle алат не подржава упите овог типа; уместо Twinkle-а, за вежбу се може користити, на пример, YASGUI SPARQL Editor или Virtuoso SPARQL editor

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 Nataly 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
  - XML формат резултата ASK упита:

```
<sparql xmlns="http://www.w3.org/2001/sw/DataAccess/rf1/result2">
  <head/>
  <results>
        <boolean>true</boolean>
  </results>
</sparql>
```

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 Bio вокабулар

```
PREFIX dbpedia-ont: <http://dbpedia.org/ontology/>
PREFIX bio: <http://purl.org/vocab/bio/0.1/>
PREFIX dcterms: <http://purl.org/dc/terms/>
CONSTRUCT {
        ?someone  bio:event [
                rdf:type bio:Birth ;
                bio:place ?birthplace ;
                dcterms:date ?birthdate ].
 } WHERE {
    ?someone rdf:type dbpedia-ont:MusicalArtist ;
        dbpedia-ont:birthDate ?birthdate ;
        dbpedia-ont:birthPlace ?birthplace .
 }
```

Упити над више дистрибуираних извора

- Сви упити које смо видели до сада, извршавали су се над подацима који долаза 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 11: пронаћи све познанике Leigh Dodds-а који имају исто презиме као пионири рачунарства

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX yago: <http://dbpedia.org/class/yago/>
SELECT ?person
FROM <http://www.ldodds.com/ldodds-knows.rdf>
WHERE {
    <http://www.ldodds.com#me> foaf:knows ?person .
    ?person foaf:familyName ?surname .
    SERVICE <http://dbpedia.org/sparql> {
            ?someone rdf:type yago:ComputerPioneers ;
                        foaf:surname ?surname .
    }
}
```

Сазнајте више о SPARQL-у

- SPARQL Query Language for RDF - specification
  - http://www.w3.org/TR/rdf-sparql-query/
- SPARQL 1.1 Query Language – specification
  - http://www.w3.org/TR/sparql11-query/
- 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/
- Bring existing data to the Semantic Web
  - http://www-128.ibm.com/developerworks/xml/library/x-semweb.html

Научите SPARQL кроз примере

- SPARQL screencast
  - http://linkeddata.deri.ie/node/58
- 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
- Primeri SPARQL upita nad >10 različitih RDF dataset-ova
  - http://openuplabs.tso.co.uk/datasets
- SPARQL upiti nad Europeana repozitorijumom
  - http://europeana.ontotext.com/sparql

Неки згодни алати за учење SPARQL-а

- Twinkle
  - http://www.ldodds.com/projects/twinkle/
- Flint SPARQL Editor
  - http://openuplabs.tso.co.uk/demos/sparqleditor
- SPARQLer - an online SPARQL query tool
  - http://www.sparql.org/sparql.html
- SparQLed – SPARQL editor with support for SPARQL 1.1
  - http://sindicetech.com/sindice-suite/sparqled/
- ARQ, a SPARQL processor for Jena framework
  - http://jena.sourceforge.net/ARQ/