SPARQL
–
SPARQL Protocol and RDF Query Language

W3C Proposed Recommendation 12 November 2007

UNIVERSITÄT KOBLENZ · LANDAU

Anfragesprache

Anforderungen

- Handhabung von RDF
- Sekundär: RDFS
- Später: OWL

„Standard“:

- Sparql for RDF

Anfragearten:

- Präzise Anfragen
  - „Was ist die Vorlesungsnummer der Vorlesung `Ethik‘?“
- Konjunktive Anfragen
  - „Wer unterrichtet einen Kurs mit einer Personalnummer, die gleich ist mit der Personalnummer die die Vorlesung `Ethik‘ gibt?“
- Ähnlichkeitsanfragen (MMDB SS07)

SPARQL

- Proposed recommendation

Kernfähigkeiten

- Extraktion von Information aus RDF Graphen
  - URIs
  - Blank nodes
  - Getypte und ungetypte Literale
- Extraktion von RDF Subgraphen
- Konstruktion neuer RDF Graphen basierend auf den Informationen in den angefragten Graphen
- Remote access protocol: SPROT

Beispielanfragen

Data:

```
<http://example.org/book/book1>
        <http://purl.org/dc/elements/1.1/title>
"SPARQL Tutorial"
```

Query:

```
SELECT ?title
WHERE
{
<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .
}
```

Triple pattern

Query Result:

title
"SPARQL Tutorial"

Prefixes

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE
{
<http://example.org/book/book1> dc:title ?title
}
```

```
PREFIX dc: http://purl.org/dc/elements/1.1/
PREFIX : <http://example.org/book/>
SELECT $title
WHERE
{
:book1 dc:title $title
}
```

```
BASE <http://example.org/book/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT $title
WHERE
{
<book1> dc:title ?title
}
```

```
BASE <http://example.org/book/>
PREFIX dcore: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE
{
<book1> dcore:title ?title
}
```


Basic Graph Pattern Matching

Data:

```
@prefix foaf:
<http://xmlns.com/foaf/0.1/> .
 _:a foaf:name "Johnny Lee Outlaw" .
_:a foaf:mbox
<mailto:outlaw@example.com> .
_:b foaf:name "A. N. Other" .
_:b foaf:mbox
<mailto:other@example.com> .
```

Query:

```
PREFIX foaf:
<http://xmlns.com/foaf/0.1/>
SELECT ?mbox
WHERE
{
?x foaf:name "Johnny Lee Outlaw" .
?x foaf:mbox ?mbox
}
```

Query Result:

mbox
<mailto:outlaw@example.com>

Optional Pattern Matching

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#> .
_:a rdf:type foaf:Person .
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@example.com> .
_:a foaf:mbox <mailto:alice@work.example> .
_:b rdf:type foaf:Person .
_:b foaf:name "Bob" .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
{
    ?x foaf:name ?name .
    OPTIONAL { ?x foaf:mbox ?mbox }
}
```

Query Result:

name	mbox
„Alice“	<mailto:alice@example.com>
„Alice“	<mailto:alice@work.example>
„Bob“	

Konstruktion eines Ausgabegraphen

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:givenname "Alice" .
_:a foaf:family_name "Hacker" .
_:b foaf:firstname "Bob" .
_:b foaf:surname "Hacker" .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vcard:
<http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT
{
?x vcard:N _:v .
_:v vcard:givenName ?gname .
 _:v vcard:familyName ?fname
}
WHERE
{
                { ?x foaf:firstname ?gname }
UNION
                { ?x foaf:givenname ?gname } .
                { ?x foaf:surname ?fname }
UNION
                { ?x foaf:family_name ?fname } .
}
```

Result:

```
@prefix vcard:
          <http://www.w3.org/2001/vcard-rdf/3.0#> .
_:v1 vcard:N        _:x .
 _:x vcard:givenName  "Alice" .
 _:x vcard:familyName "Hacker" .
 _:v2 vcard:N        _:z .
 _:z vcard:givenName  "Bob" .
_:z vcard:familyName "Hacker" .
```

Filter

<is web>

```
SELECT ?y ?z
WHERE
{  ?x ex:gelesenVon ?y . ?y hatAlter ?a .
   ?x ex:gehoertVon ?z .  ?z hatAlter ?b .}
FILTER (?a > ?b)
```

```
ex:5041 ex:gelesenVon ex:Sokrates
ex:5041 ex:gehoertVon ex:Theophrastos
ex:5041 ex:gelesenVon ex:Sokrates
ex:5041 ex:gehoertVon ex:Carnap
ex:5001 ex:gelesenVon ex:Kant
ex:5001 ex:gehoertVon ex:Fichte
ex:Sokrates hatAlter 2477^^xsd:int
ex:Carnap hatAlter 117^^xsd:int
ex:Theophrastos hatAlter 2478^^xsd:int
```


Komplexeres Szenario

UNIVERSITÄT KOBLENZ · LANDAU

Beispiel: Named Graphs

<is web>

EXAMPLE 1.

```
:DBLP {:NGPaper dc:creator :Alice.
       :NGPaper dc:creator :Bob. }

:bobFOAF {
  :bobFOAF foaf:primaryTopic   :Bob.
  :Bob     foaf:name           "Bob"^^xsd:String.
  :Bob     foaf:currentProject :K-Space.
  :K-Space foaf:fundedBy       :EU.
  :Bob     foaf:phone          "+49-261-287-2868".}

:MikesProject {
  :SemWebProj foaf:name "Networked Graphs".}
```

Vergleiche „**Networked Graphs: A Declarative Mechanism for SPARQL Rules, SPARQL Views and RDF Data Integration on the Web“ von** Simon Schenk, Steffen Staab

Beispielanfrage mit Optional

<is web>

EXAMPLE 2.

```
CONSTRUCT {?member foaf:currentProject ?project.
           ?member foaf:phone ?phone}
FROM NAMED :bobFOAF  FROM NAMED :chrisFOAF  FROM NAMED...
WHERE {
 GRAPH ?foafFile {
  ?foafFile foaf:primaryTopic ?member
  OPTIONAL { ?member foaf:phone ?phone } } }
```

Beispielanfrage mit Negation

<is web>

EXAMPLE 3.

```
CONSTRUCT {:SemWebProject :acknowledges ?author}
FROM NAMED :DBLP        FROM NAMED :mikesProject
WHERE {
 GRAPH :DBLP {
  ?paper dc:creator ?author.
  ?paper dc:creator ?member  }.
 GRAPH :mikesProject {
  ?member foaf:currentProject :SemWebProject
  OPTIONAL { ?x foaf:currentProject :SemWebProject
             FILTER (?x = ?author) }
  FILTER (!BOUND(?x)) } }
```


Semantik von SPARQL

UNIVERSITÄT KOBLENZ · LANDAU

RDF Terme

<is web>

Definition: RDF Term

Sei I die Menge aller IRIs.
Sei L die Menge von RDF Literalen.
Sei B die Menge der Blank Nodes in RDF Graphen

Die Menge von RDF Termen, T, ist
T:=I [L [B.

RDF Datenmenge

<is web>

Definition: RDF Datenmenge

Eine RDF Datenmenge ist eine Menge:
{ G, (<u_1>, G_1), (<u_2>, G_2), . . . (<u_n>, G_n) }
wobei G und jedes G_i Graphen sind und jedes <u_i> eine IRI ist. Jedes <u_i> ist einzigartig.

G wird Default Graph genannt. (<u_i>, G_i) werden Named Graphs genannt.

Definition: Aktive Graph

Der **aktive Graph** ist der Graph aus dem Datenmenge, der für basic graph pattern matching verwendet wird.

Triple Pattern

Definition: Anfragevariable

Eine Anfragevariable ist ein Mitglied aus der Menge V, wobei V unendlich groß ist und disjunkt von der Menge der RDF Terme, T.

Definition: Triple Pattern

Ein Triple Pattern ist ein Element aus der Menge

(T [V) £ (I [V) £ (T [V)

Beispiel: Triple Pattern

?x ex:gelesenVon ?y .

Basic Graph Pattern

Definition: Basic Graph Pattern

Ein Basic Graph Pattern ist eine Menge von Triple Patterns

NB: Das leere Graph Pattern ist ein Basic Graph Pattern – die leere Menge.

Beispiel:

{ ?x ex:gelesenVon ?y .
?x ex:gehoertVon ?z .}

ex:5041 ex:gelesenVon ex:Sokrates
ex:5041 ex:gehoertVon ex:Theophrastos
ex:5041 ex:gelesenVon ex:Sokrates
ex:5041 ex:gehoertVon ex:Carnap
ex:5001 ex:gelesenVon ex:Kant
ex:5001 ex:gehoertVon ex:Fichte

Lösung (Solution Mapping)

Eine Lösung ist eine Abbildung von einer Menge von Variablen in eine Menge von RDF Termen.

Definition: Lösung

Eine **Lösung**, μ, ist eine partielle Funktion μ: V ! T.

Der Definitionsbereich von μ, dom(μ), ist die Teilmenge von V, auf der μ definiert ist.

Beispiel: { ?x ex:gelesenVon ?y . ?x ex:gehoertVon ?z .}

μ:

?x	?y	?z
Ethik	Sokrates	Theophrastos

μ(`?x‘)=`Ethik‘, μ(`?y‘)=`Sokrates‘,…

Lösung (Solution Mapping)

Definition: Lösungsmenge

Eine Lösungsmenge ist eine Menge von Lösungen

{μ_1, μ_2, μ_3, …}.

ex:5041 ex:gelesenVon ex:Sokrates
ex:5041 ex:gehoertVon ex:Theophrastos
ex:5041 ex:gelesenVon ex:Sokrates
ex:5041 ex:gehoertVon ex:Carnap
ex:5001 ex:gelesenVon ex:Kant
ex:5001 ex:gehoertVon ex:Fichte

	?x	?y	?z
μ_1:	Ethik	Sokrates	Theophrastos
μ_2:	Ethik	Sokrates	Carnap
μ_3:	Grundzüge	Kant	Fichte

μ_3(`?z‘)=`Fichte‘

SPARQL Anfrage (Syntax)

Definition: SPARQL Query

Eine SPARQL abstrakte Anfrage ist ein Tupel (E, DS, R) wobei:

E ein SPARQL Algebra-Ausdruck ist,

DS eine RDF Datenmenge ist, und

R ein Anfrageformular ist (query form).

Beispiel:

CONSTRUCT ?x ?y ?z. } R
FROM ex:myGraph } DS
WHERE ?x ?y ?z . } E

SPARQL Anfrage (Syntax)

Definition: SPARQL Query

Eine SPARQL abstrakte Anfrage ist ein Tupel (E, DS, R) wobei:

E ein SPARQL Algebra-Ausdruck ist,

DS eine RDF Datenmenge ist, und

R ein Anfrageformular ist (query form).

Beispiel:

CONSTRUCT ?x ?y ?z.
FROM ex:myGraph FROM ex:yourGraph
FROM NAMED ex:hisGraph FROM NAMED ex:herGraph
WHERE ?x ?y ?z .

Rekursive Abarbeitung des Algebra Ausdrucks

{ ?vorlesung ex:hatSws ?sws . FILTER (?sws < 3)
OPTIONAL { ?vorlesung ex:buchempfehlung ?buch . }
{ ?vorlesung ex:gelesenVon ?person . } UNION
{ ?vorlesung ex:gehörtVon ?person . }
}

Konvertierungen: Von SPARQL Algebra zu Relationaler Algebra

AND	wird zu	⋈
UNION	wird zu	[
OPTIONAL	wird zu	⟕
FILTER	wird zu	^

Rekursive Abarbeitung des Algebra Ausdrucks

{ BGP(?vorlesung ex:hatSws ?sws .)
FILTER (?sws < 3)
OPTIONAL { BGP(?vorlesung ex:buchempfehlung ?buch .)}
{ BGP(?vorlesung ex:gelesenVon ?person .) } UNION
{ BGP(?vorlesung ex:gehörtVon ?person .) }
}

Konvertierungen: Von SPARQL Algebra zu Relational Algebra

AND	wird zu	⋈
UNION	wird zu	[
OPTIONAL	wird zu	⟕
FILTER	wird zu	^

{ (BGP(?vorlesung ex:hatSws ?sws .)
FILTER (?sws < 3))
⟕ { BGP(?vorlesung ex:buchempfehlung ?buch .)}

(BGP(?vorlesung ex:gelesenVon ?person .) [
BGP(?vorlesung ex:gehörtVon ?person .))
}

Konvertierungen: Von SPARQL Algebra zu Relational Algebra

AND	wird zu	⋈
UNION	wird zu	[
OPTIONAL	wird zu	⟕
FILTER	wird zu	^

((BGP(?vorlesung ex:hatSws ?sws .)
FILTER (?sws < 3))
⟕ BGP(?vorlesung ex:buchempfehlung ?buch .)) ⋈

(BGP(?vorlesung ex:gelesenVon ?person .) [
BGP(?vorlesung ex:gehörtVon ?person .))

Konvertierungen: Von SPARQL Algebra zu Relational Algebra

AND	wird zu	⋈
UNION	wird zu	[
OPTIONAL	wird zu	⟕
FILTER	wird zu	^

FILTER((?sws < 3),
(BGP(?vorlesung ex:hatSws ?sws .)
⟕ BGP(?vorlesung ex:buchempfehlung ?buch .)) ⋈
(BGP(?vorlesung ex:gelesenVon ?person .) [
BGP(?vorlesung ex:gehörtVon ?person .)))

Konvertierungen: Von SPARQL Algebra zu Relational Algebra

AND	wird zu	⋈
UNION	wird zu	[
OPTIONAL	wird zu	⟕
FILTER	wird zu	^

FILTER((?sws < 3),
(BGP(?vorlesung ex:hatSws ?sws .)
⟕ BGP(?vorlesung ex:buchempfehlung ?buch .)) ⋈
(BGP(?vorlesung ex:gelesenVon ?person .) [
BGP(?vorlesung ex:gehörtVon ?person .)))

μ_3:

?person	?vorlesung
ex:Sokrates	ex:5041

μ_4:

?person	?vorlesung
ex:Theophrastos	ex:5041

μ_2:

?vorlesung	?buch
ex:5041	ex:SokBuch

μ_1:

?vorlesung	?sws
ex:5041	2

ex:5041 ex:gelesenVon ex:Sokrates .
ex:5041 ex:gehoertVon ex:Theophrastos .
ex:5041 ex:buchempfehlung ex:SokBuch .
ex:5041 ex:hatSWS 2 .

ex:5001 ex:gelesenVon ex:Kant
ex:5001 ex:gehoertVon ex:Fichte

FILTER((?sws < 3),
(BGP(?vorlesung ex:hatSws ?sws .)
⟕ BGP(?vorlesung ex:buchempfehlung ?buch .)) ⋈
(BGP(?vorlesung ex:gelesenVon ?person .) [
BGP(?vorlesung ex:gehörtVon ?person .)))

μ_3:

?person	?vorlesung
ex:Sokrates	ex:5041

μ_4:

?person	?vorlesung
ex:Theophrastos	ex:5041

μ_2:

?vorlesung	?buch
ex:5041	ex:SokBuch

μ_1:

?vorlesung	?sws
ex:5041	2

FILTER((?sws<3),
$(\mu_1 \text{ ⟕ } \mu_2) \bowtie$
$(\mu_3 [\mu_4))$

- Kant fehlt im Beispiel

FILTER((?sws<3),
$(\mu_1 \text{ ⟕ } \mu_2) \bowtie$
$(\mu_3 [\mu_4))$

$\mu_3[\mu_4$:

?person	?vorlesung
ex:Sokrates	ex:5041
ex:Theophrastos	ex:5041

μ_3:

?person	?vorlesung
ex:Sokrates	ex:5041

μ_4:

?person	?vorlesung
ex:Theophrastos	ex:5041

μ_1:

?vorlesung	?sws
ex:5041	2

μ_2:

?vorlesung	?buch
ex:5041	ex:SokBuch

$\mu_1 \text{ ⟕ } \mu_2$:

?vorlesung	?sws	?buch
ex:5041	2	ex:SokBuch

FILTER((?sws<3),
$(\mu_1 \text{ ⟕ } \mu_2) \bowtie$
$(\mu_3 [\mu_4))$

?vorlesung	?sws	?buch	?person
ex:5041	2	ex:SokBuch	ex:Sokrates
ex:5041	2	ex:SokBuch	ex:Theophrastos

$\mu_3[\mu_4$:

?person	?vorlesung
ex:Sokrates	ex:5041
ex:Theophrastos	ex:5041

$\mu_1 \text{ ⟕ } \mu_2$:

?vorlesung	?sws	?buch
ex:5041	2	ex:SokBuch

Solution Sequence Modifiers

Definition: Solution Sequence Modifier

A solution sequence modifier is one of:

Order By modifier: put the solutions in order

Projection modifier: choose certain variables

Distinct modifier: ensure solutions in the sequence are unique

Reduced modifier: permit any non-unique solutions to be eliminated

Offset modifier: control where the solutions start from in the overall sequence of solutions

Limit modifier: restrict the number of solutions

Networked Graphs and Views

UNIVERSITÄT KOBLENZ · LANDAU

Networked Graphs – 1 (Views)

EXAMPLE 5.

```
:bobFOAF {
 :bobFOAF foaf:primaryTopic    :Bob.
 :Bob     foaf:name            "Bob"^^xsd:String.
 :Bob     foaf:currentProject  :K-Space.
 :K-Space foaf:fundedBy        :EU.
 :Bob     foaf:phone           "+49-261-287-2868".
 :bobFOAF g:definedBy
   "CONSTRUCT {:Bob foaf:knows ?person}
    FROM :mikesProject
    WHERE { ?person foaf:currentProject :SemWebProject
            FILTER (?person != :Bob)}"^^g:query.
}
```


Networked Graphs – 2 (Views)

EXAMPLE 5.

```
:bobFOAF {
 :bobFOAF foaf:primaryTopic    :Bob.
 :Bob     foaf:name            "Bob"^^xsd:String.
 :Bob     foaf:currentProject  :K-Space.
 :K-Space foaf:fundedBy        :EU.
 :Bob     foaf:phone           "+49-261-287-2868".
 :bobFOAF g:definedBy
   "CONSTRUCT {:Bob foaf:knows ?person}
    FROM :mikesProject
    WHERE { ?person foaf:currentProject :SemW
            FILTER (?person != :Bob)}"^^g:que
}
```

```
:mikesProject {
 :SemWebProj foaf:name "Networked Graphs".
 :mikesProject g:definedBy
   "CONSTRUCT {?member foaf:currentProject ?project.
               ?member foaf:phone ?phone}
    FROM NAMED :bobFOAF    FROM NAMED :chrisFOAF ...
    WHERE { GRAPH ?foafFile {
      ?foafFile foaf:primaryTopic ?member
      OPTIONAL {?member foaf:phone ?phone}}}"^^g:query.
 :mikesProject g:definedBy
   "CONSTRUCT {?paper dc:creator ?author}..."^^g:query.
 :mikesProject g:definedBy
   "CONSTRUCT {:SemWebProject :acknowledges ?author}
    FROM NAMED :DBLP       FROM NAMED :mikesProject
    WHERE {
      GRAPH :DBLP { ?paper dc:creator ?author.
                    ?paper dc:creator ?member}.
      GRAPH :mikesProject {
        ?member foaf:currentProject :SemWebProject
        OPTIONAL { ?x foaf:currentProject :SemWebProject
                   FILTER (?x = ?author)}
        FILTER (!BOUND(?x))} }"^^g:query.
}
```

Interdependenz

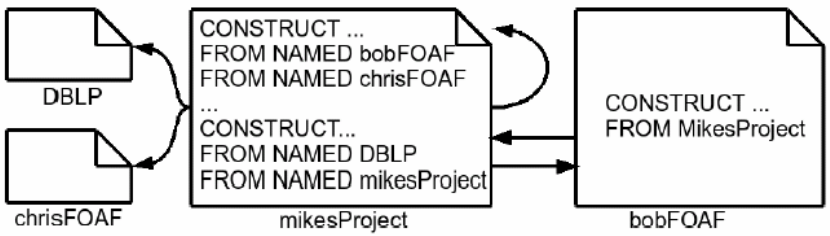

Figure 1: An Interdependence Set

Lösung im: Advanced Data Modeling! – SS08

Filter

Einfache Filter

- Testen auf Typen
- Vergleiche (auf jeweiligen Datentypen, z.B. integer<)
- Reguläre Ausdrücke

Komplexe Filter

- Bool'sche Kombinationen einfacher Filter
- Externer Funktionsaufruf (Funktion benamt durch IRI)

Ausblick

Weitere Ansätze

- OWL QL for OWL [Fikes]
- Ähnlichkeitsanfragen: [Anyanwu]
- Relevance feedback: [Stojanovic]
- More recent work by [Parsia et al07], [Kubias et al, 07]

- Kemafor Anyanwu, Angela Maduko, Amit P. Sheth: SemRank: ranking complex relationship search results on the semantic web. WWW 2005: 117-127
- Nenad Stojanovic: On Analysing Query Ambiguity for Query Refinement: The Librarian Agent Approach. ER 2003: 490-505
- Fikes, R.; Hayes, P.; & Horrocks, I. OWL-QL - A Language for Deductive Query Answering on the Semantic Web. Knowledge Systems Laboratory, Stanford University, Stanford, CA, 2003. ftp://ftp.ksl.stanford.edu/pub/KSL_Reports/KSL-03-14.pdf.gz
- Parsia et al. In: Proc. OWL-ED 2007 workshop
- Kubias, Staab, Pan. In: Proc. OWL-ED 2007 workshop