SIEMENS

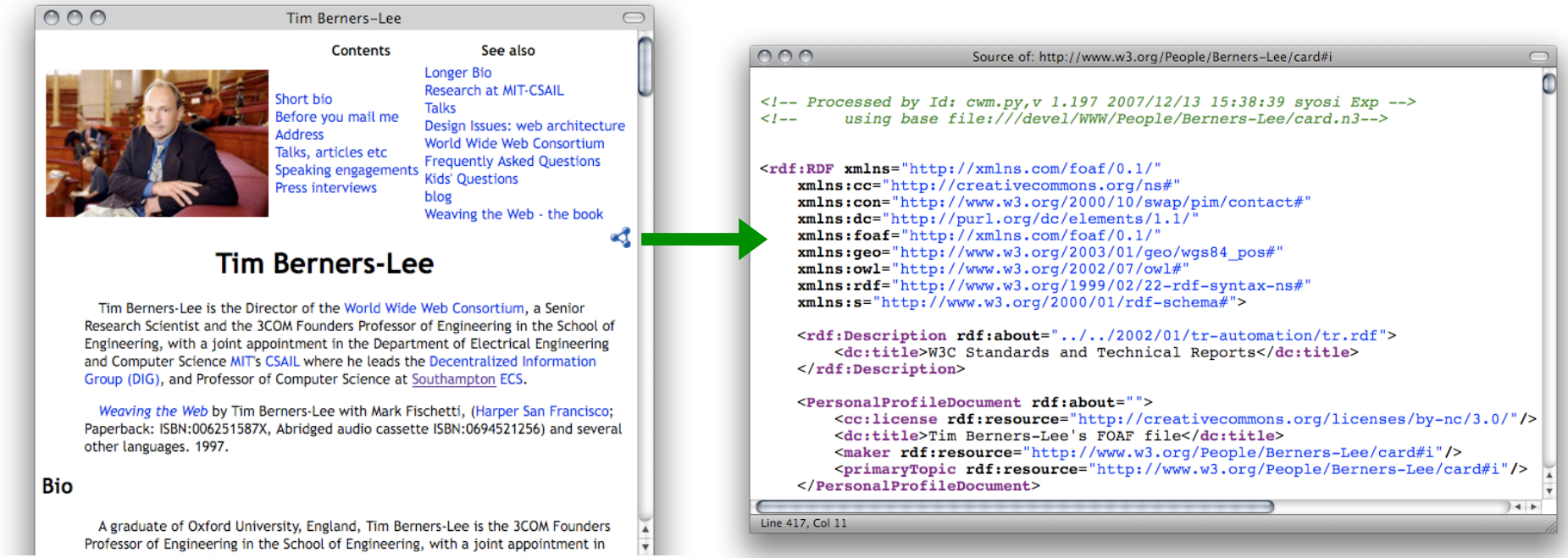
Unit 4:
SPARQL1.0 in Detail &
SPARQL's formal semantics

SIEMENS

Recall from Lecture 1/Assignment 1

Lots of RDF data already published, e.g. about **Tim Berners-Lee**

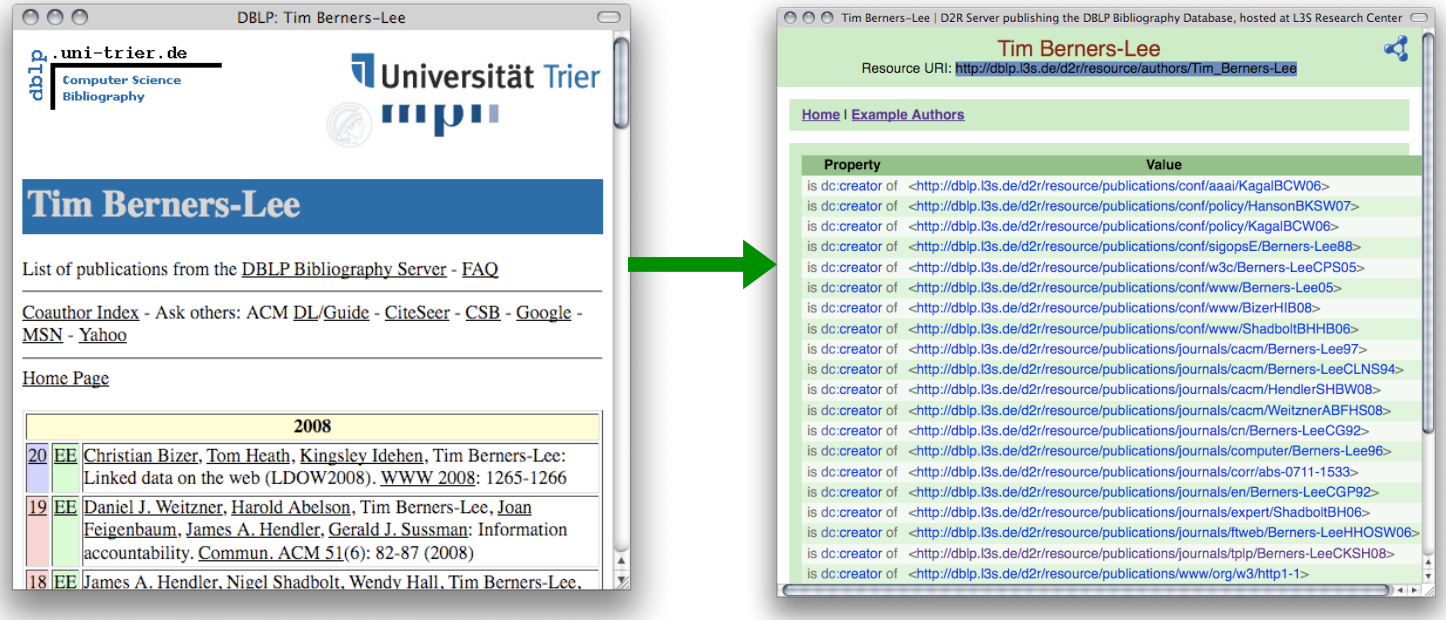
FOAF/RDF linked from a home page: personal data (foaf:name, foaf:phone, etc.), relationships foaf:knows, rdfs:seeAlso)



Recall from Lecture 1/Assignment 1

SIEMENS

Also from exports … if they're up and running (sorry about that)



Gives unique URIs to authors, documents, etc. on DBLP! E.g.,

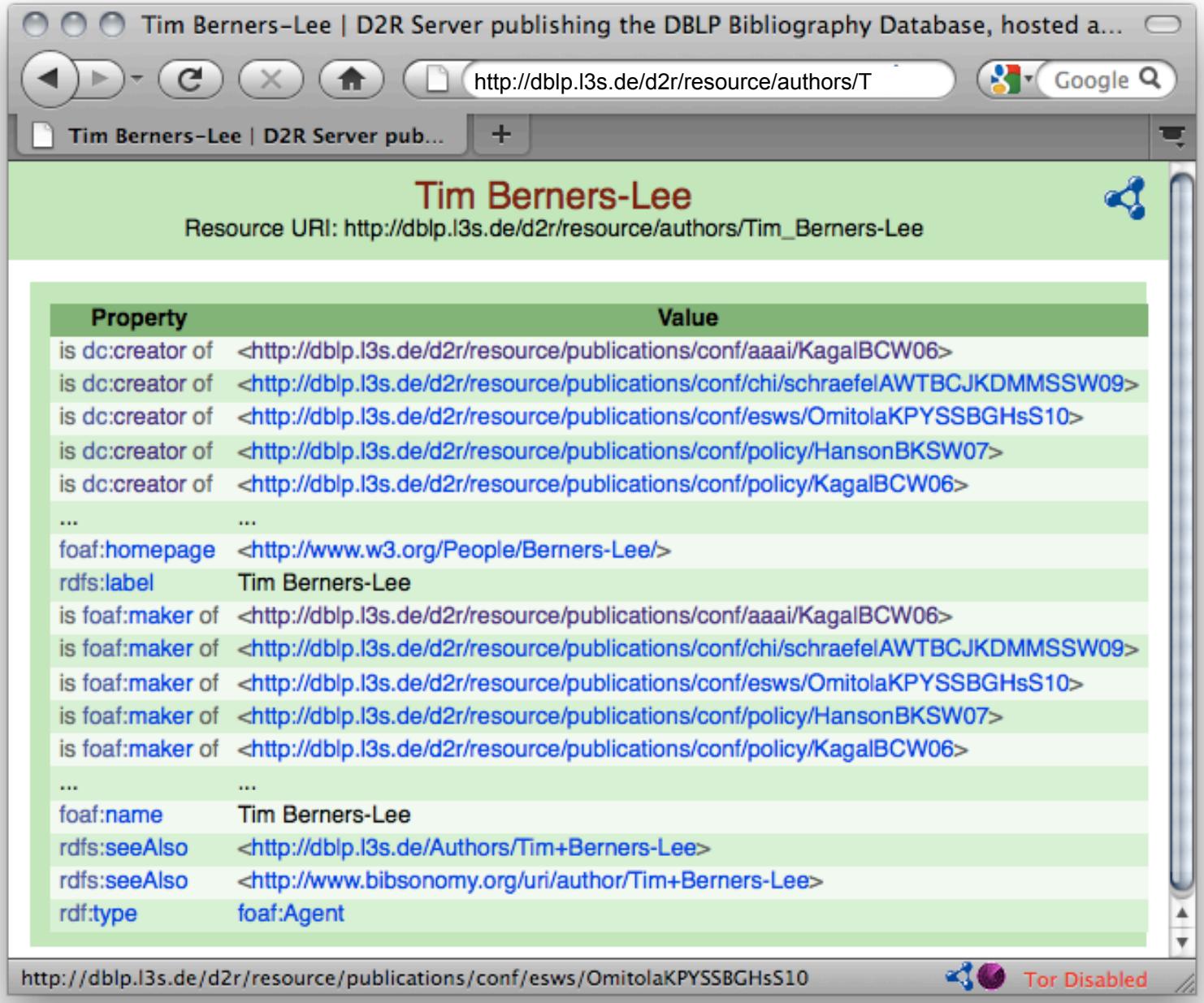
http://dblp.l3s.de/d2r/resource/authors/Tim_Berners-Lee,

http://dblp.l3s.de/d2r/resource/publications/journals/tplp/Berners-LeeCKSH08

Provides RDF version of all DBLP data and even a SPARQL query interface!

RDF Data at http://dblp.l3s.de/d2r/resource/authors/Tim_Berners-Lee

SIEMENS



SIEMENS

We know how this data looks in Turtle syntax…

- DBLP Data in RDF: **Triples Turtle Syntax**:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix swrc: <http://swrc.ontoware.org/ontology#> .

<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> rdf:type swrc:Article.
<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> dcterms:issued "2008"^^xsd:gYear .
<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> foaf:maker <http://dblp.l3s…/Tim_Berners-Lee> .
<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> foaf:maker <http://dblp.l3s…/Dan_Connolly> .
<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> foaf:maker <http://dblp.l3s…/Jim_Hendler> .
<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> foaf:maker <http://dblp.l3s…/Lalana_Kagal> .
<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> foaf:maker <http://dblp.l3s…/Yosi_Scharf> .
  …
<http://dblp.l3s…/conf/aaai/KagalBCW06> rdf:type swrc:inProceedings .
<http://dblp.l3s…/conf/aaai/KagalBCW06> foaf:maker <http://dblp.l3s…/Tim_Berners-Lee> .
  …
<http://dblp.l3s…/Tim_Berners-Lee> foaf:homepage <http://www.w3.org/People/Berners-Lee/> .
<http://dblp.l3s…/Tim_Berners-Lee> foaf:name "Tim Berners-Lee" .
```

SIEMENS

We know the Turtle shortcuts (we'll need these again today)…

- DBLP Data in RDF: **Triples Turtle Syntax**:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix swrc: <http://swrc.ontoware.org/ontology#> .

<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> rdf:type swrc:Article ;
                                                   dcterms:issued "2008"^^xsd:gYear ;
                                                   foaf:maker <http://dblp.l3s…/Tim_Berners-Lee> ,
                                                              <http://dblp.l3s…/Dan_Connolly> ,
                                                              <http://dblp.l3s…/Jim_Hendler> ,
                                                              <http://dblp.l3s…/Lalana_Kagal> ,
                                                              <http://dblp.l3s…/Yosi_Scharf> .
  …
<http://dblp.l3s…/conf/aaai/KagalBCW06> rdf:type swrc:inProceedings ;
                                        foaf:maker <http://dblp.l3s…/Tim_Berners-Lee> .
  …
<http://dblp.l3s…/Tim_Berners-Lee> foaf:homepage <http://www.w3.org/People/Berners-Lee/> ;
                                   foaf:name "Tim Berners-Lee" .
```

SIEMENS

SPARQL 1.0 in Detail:

Let's focus on SELECT queries first…
in more detail than in lecture 1:

- Basic Graph Patterns
- UNION patterns
- OPTIONAL Patterns
- FILTERs
- GRAPH graph patterns & Datasets
- ORDER BY / LIMIT / OFFSET

Basic Graph Patterns

A *triple pattern* is an RDF triple, optionally containing variables (prefixed by '?') in either s/p/o position.
A *basic graph pattern (BGP)* is a set of triple patterns.

Informally, SPARQL tries to match these patterns against a given RDF graph and returns as solution a ***multi-set of sets of variable bindings*** for the variables in the pattern.

SIEMENS

Example: Basic graph pattern matching

- Let's assume this graph:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix swrc: <http://swrc.ontoware.org/ontology#> .

<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> rdf:type swrc:Article ;
                                                   dcterms:issued "2008"^^xsd:gYear ;
                                                   foaf:maker <http://dblp.l3s…/Tim_Berners-Lee> ,
                                                              <http://dblp.l3s…/Dan_Connolly> ,
                                                              <http://dblp.l3s…/Jim_Hendler> ,
                                                              <http://dblp.l3s…/Lalana_Kagal> ,
                                                              <http://dblp.l3s…/Yosi_Scharf> .

<http://dblp.l3s…/Tim_Berners-Lee> foaf:homepage <http://www.w3.org/People/Berners-Lee/> ;
                                   foaf:name "Tim Berners-Lee" .

<http://dblp.l3s…/Dan_Connolly> foaf:name "Dan Connolly" .
<http://dblp.l3s…/Jim_Hendler>  foaf:name "James Hendler" .
<http://dblp.l3s…/Lalana_Kagal>  foaf:name "Lalana Kagal" .
<http://dblp.l3s…/Yosi_Scharf> foaf:name "Yosi Scharf" .
```

SIEMENS

Example: Basic graph pattern matching

```
{ ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
      ?D foaf:maker ?CoAuth .
      ?CoAuth foaf:name ?N  }
```

```
<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> rdf:type swrc:Article ;
                                                   dcterms:issued "2008"^^xsd:gYear ;
                                                   foaf:maker <http://dblp.l3s…/Tim_Berners-Lee> ,
                                                              <http://dblp.l3s…/Dan_Connolly> ,
                                                              <http://dblp.l3s…/Jim_Hendler> ,
                                                              <http://dblp.l3s…/Lalana_Kagal> ,
                                                              <http://dblp.l3s…/Yosi_Scharf> .

<http://dblp.l3s…/Tim_Berners-Lee> foaf:homepage <http://www.w3.org/People/Berners-Lee/> ;
                                   foaf:name "Tim Berners-Lee" .

<http://dblp.l3s…/Dan_Connolly> foaf:name “Dan Connolly” .
<http://dblp.l3s…/Jim_Hendler>  foaf:name “James Hendler” .
<http://dblp.l3s…/Lalana_Kagal>  foaf:name “Lalana Kagal” .
<http://dblp.l3s…/Yosi_Scharf> foaf:name “Yosi Scharf” .
```

SIEMENS

Example: Basic graph pattern matching

```
{ ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
      ?D foaf:maker ?CoAuth .
      ?CoAuth foaf:name ?N  }
```

```
<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> rdf:type swrc:Article ;
                                                   dcterms:issued "2008"^^xsd:gYear ;
                                                   foaf:maker <http://dblp.l3s…/Tim_Berners-Lee> ,
                                                              <http://dblp.l3s…/Dan_Connolly> ,
                                                              <http://dblp.l3s…/Jim_Hendler> ,
                                                              <http://dblp.l3s…/Lalana_Kagal> ,
                                                              <http://dblp.l3s…/Yosi_Scharf> .

<http://dblp.l3s…/Tim_Berners-Lee> foaf:homepage <http://www.w3.org/People/Berners-Lee/> ;
                                   foaf:name "Tim Berners-Lee" .

<http://dblp.l3s…/Dan_Connolly> foaf:name “Dan Connolly” .
<http://dblp.l3s…/Jim_Hendler>  foaf:name “James Hendler” .
<http://dblp.l3s…/Lalana_Kagal>  foaf:name “Lalana Kagal” .
<http://dblp.l3s…/Yosi_Scharf> foaf:name “Yosi Scharf” .
```

SIEMENS

Example: Basic graph pattern matching

```
{ ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
      ?D foaf:maker ?CoAuth .
      ?CoAuth foaf:name ?N  }
```

```
<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> rdf:type swrc:Article ;
                                                   dcterms:issued "2008"^^xsd:gYear ;
                                                   foaf:maker <http://dblp.l3s…/Tim_Berners-Lee> ,
                                                              <http://dblp.l3s…/Dan_Connolly> ,
                                                              <http://dblp.l3s…/Jim_Hendler> ,
                                                              <http://dblp.l3s…/Lalana_Kagal> ,
                                                              <http://dblp.l3s…/Yosi_Scharf> .

<http://dblp.l3s…/Tim_Berners-Lee> foaf:homepage <http://www.w3.org/People/Berners-Lee/> ;
                                   foaf:name "Tim Berners-Lee" .

<http://dblp.l3s…/Dan_Connolly> foaf:name "Dan Connolly" .
<http://dblp.l3s…/Jim_Hendler>  foaf:name "James Hendler" .
<http://dblp.l3s…/Lalana_Kagal>  foaf:name "Lalana Kagal" .
<http://dblp.l3s…/Yosi_Scharf> foaf:name "Yosi Scharf" .
```

SIEMENS

Example: Basic graph pattern matching

```
{ ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
      ?D foaf:maker ?CoAuth .
      ?CoAuth foaf:name ?N  }
```

?D	?CoAuth	?N
Berners-LeeCKSH0	:Tim_Berners-Lee	"Tim Berners-Lee"
Berners-LeeCKSH0	:Dan_Connolly	"Dan Connolly"
Berners-LeeCKSH0	:Jim_Hendler	"James Hendler"
Berners-LeeCKSH0	:Lalana_Kagal	"Lalana Kagal"
Berners-LeeCKSH0	:Yosi_Scharf	"Yosi Scharf"

SIEMENS

Example: Basic graph pattern matching

```
{ ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
      ?D foaf:maker ?CoAuth .
      ?CoAuth foaf:name ?N  }
```

(multi-)set of sets of variable bindings:

Solution =

{ {?D → Berners-LeeCKSH0, ?CoAuth → :Tim_Berners-Lee, ?N → "Tim Berners-Lee" } ,
{?D → Berners-LeeCKSH0, ?CoAuth → :Dan_Connolly , ?N → "Dan Connolly" } ,
{?D → Berners-LeeCKSH0, ?CoAuth → :Jim_Hendler , ?N → "James Hendler" } ,
{?D → Berners-LeeCKSH0, ?CoAuth → :Lalana_Kagal , ?N → "Lalana Kagal" } ,
{?D → Berners-LeeCKSH0, ?CoAuth → :Yosi_Scharf , ?N → "Yosi Scharf" } }

SIEMENS

Example: Basic graph pattern matching

```
 {<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> foaf:maker
<http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.  }
```

(multi-)set of sets of variable bindings: *What if the pattern has no variables?*

Solution =
{∅}

Note: **{}** != **{ ∅ }**

SIEMENS

Example: Basic graph pattern matching

```
 {<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> foaf:maker
<http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.  }
```

(multi-)set of sets of variable bindings: *What if the pattern has no variables?*

Solution =

{∅}

Matches, with one empty binding.

Note: **{}** != **{ ∅ }**

SIEMENS

Example: Basic graph pattern matching

```
 {<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> foaf:maker
<http://dblp.l3s.de/…/authors/Thomas Eiter>. }
```

(multi-)set of variable bindings: *What if there is no matching triples?*

Solution =
{ }

Note: **{}** != **{** ∅ **}**

SIEMENS

Example: Basic graph pattern matching

```
 {<http://dblp.l3s…/journals/tplp/Berners-LeeCKSH08> foaf:maker
<http://dblp.l3s.de/…/authors/Thomas Eiter>. }
```

(multi-)set of variable bindings: *What if there is no matching triples?*

Solution =

{ }

Doesn't match anything.

Note: **{}** != **{** ∅ **}**

SIEMENS

SELECT (DISTINCT) & ASK

"Result forms" SELECT & ASK are defined *on top* of pattern matching:

- SELECT … Projection
- ASK … True/False, i.e. (returns **Solution != {}**)

SELECT (DISTINCT) & ASK

SIEMENS

"Result forms" SELECT & ASK are defined on top of pattern matching:

- SELECT … Projection

```
SELECT *
WHERE { ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
        ?D foaf:maker ?CoAuth .
        ?CoAuth foaf:name ?N  }
```

Solution =

{ {?D → Berners-LeeCKSH0, ?CoAuth → :Tim_Berners-Lee, ?N → "Tim Berners-Lee" } ,
{?D → Berners-LeeCKSH0, ?CoAuth → :Dan_Connolly , ?N → "Dan Connolly" } ,
{?D → Berners-LeeCKSH0, ?CoAuth → :Jim_Hendler , ?N → "James Hendler" } ,
{?D → Berners-LeeCKSH0, ?CoAuth → :Lalana_Kagal , ?N → "Lalana Kagal" } ,
{?D → Berners-LeeCKSH0, ?CoAuth → :Yosi_Scharf , ?N → "Yosi Scharf" } }

SELECT (DISTINCT) & ASK

SIEMENS

"Result forms" SELECT & ASK are defined on top of pattern matching:

- SELECT … **Projection**

```
SELECT ?D
WHERE { ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
        ?D foaf:maker ?CoAuth .
        ?CoAuth foaf:name ?N  }
```

Solution =

{ {?D → Berners-LeeCKSH0 } ,
{?D → Berners-LeeCKSH0} ,
{?D → Berners-LeeCKSH0} ,
{?D → Berners-LeeCKSH0} ,
{?D → Berners-LeeCKSH0 } }

***multi-**set (!) of sets of variable bindings,*

i.e. possibly duplicates

SELECT (DISTINCT) & ASK

SIEMENS

"Result forms" SELECT & ASK are defined on top of pattern matching:

- SELECT **DISTINCT** … Projection+eliminate duplicates

```
SELECT DISTINCT ?D
WHERE { ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
        ?D foaf:maker ?CoAuth .
        ?CoAuth foaf:name ?N  }
```

Solution =

{{?D → Berners-LeeCKSH0 } }

SELECT (DISTINCT) & ASK

SIEMENS

"Result forms" SELECT & ASK are defined on top of pattern matching:

- SELECT … Projection to non-occurring variables may yield empty binding set

```
SELECT ?X
WHERE { ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
        ?D foaf:maker ?CoAuth .
        ?CoAuth foaf:name ?N  }
```

Solution =

{ ∅,
∅,
∅,
∅,
∅ }

SELECT (DISTINCT) & ASK

SIEMENS

"Result forms" SELECT & ASK are defined on top of pattern matching:

- **ASK** … True/False, i.e. (returns **Solution != {}**)

```
ASK
WHERE { ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
        ?D foaf:maker ?CoAuth .
        ?CoAuth foaf:name ?N  }
```

TRUE

SIEMENS

UNIONs of conjunctive queries…

Example:

*"Give me all names of co-authors **or** friends of Tim Berners-Lee"*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N
WHERE {
        { [ foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>,
                    [ foaf:name ?N ] ] . }


      }
```

?N
"Lalana Kagal"
"Tim Berners-Lee"
"Dan Connolly"
"Jim Hendler"
…

SIEMENS

UNIONs of conjunctive queries…

Example:

*"Give me all names of co-authors **or** friends of Tim Berners-Lee"*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N
WHERE {

        { <http://www.w3.org/People/Berners-Lee/card#i> foaf:knows ?F .
          ?F foaf:name ?N }
      }
```

?N
"Lalana Kagal"
"Tim Berners-Lee"
"Dan Connolly"
"Jim Hendler"
…

?N
"Michael Hausenblas"
"Jim Hendler"
"Charles McCathieNevile"
…

UNIONs of conjunctive queries…

SIEMENS

Example:

*"Give me all names of co-authors **or** friends of Tim Berners-Lee"*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N
WHERE {
        { [ foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>,
                       [ foaf:name ?N ] ] . }
        UNION
        { <http://www.w3.org/People/Berners-Lee/card#i> foaf:knows ?F .
          ?F foaf:name ?N }
      }
```

?N
"Lalana Kagal"
"Tim Berners-Lee"
"Dan Connolly"
"Jim Hendler"
…

U

?N
"Michael Hausenblas"
"Jim Hendler"
"Charles McCathieNevile"
…

=

?N
"Lalana Kagal"
"Tim Berners-Lee"
"Dan Connolly"
"Jim Hendler"
…
"Michael Hausenblas"
"Jim Hendler"
"Charles McCathieNevile"
…

UNIONs of conjunctive queries…

SIEMENS

Example:

*"Give me all names of co-authors **or** friends of Tim Berners-Lee"*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N
WHERE {
        { [ foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners_Lee>,
                       [ foaf:name ?N ] ] . }
        UNION
        { <http://www.w3.org/People/Berners-Lee/card#i> foaf:knows ?F .
          ?F foaf:name ?N }
      }
```

Note: Again: Duplicates possible by UNION, **bag/multiset** semantics!

?N
"Lalana Kagal"
"Tim Berners-Lee"
"Dan Connolly"
"Jim Hendler"
…

U

?N
"Michael Hausenblas"
"Jim Hendler"
"Charles McCathieNevile"
…

=

?N
"Lalana Kagal"
"Tim Berners-Lee"
"Dan Connolly"
"Jim Hendler"
…
"Michael Hausenblas"
"Jim Hendler"
"Charles McCathieNevile"
…

UNIONs of conjunctive queries…

SIEMENS

Example:

*"Give me all names of co-authors **or** friends of Tim Berners-Lee"*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?CoAuthN ?FrN
WHERE {
        { [ foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-L…>,
                       [ foaf:name ?CoAuthN ] ] . }
        UNION
        { <http://www.w3.org/People/Berners-Lee/card#i> foaf:knows ?F .
          ?F foaf:name ?FrN }
      }
```

Note: variables can be unbound in a result!

?CoAuthN	?FrN
"Lalana Kagal"	
"Tim Berners-Lee"	
"Dan Connolly"	
"Jim Hendler"	
…	
	"Michael Hausenblas"
	"Jim Hendler"
	"Charles McCathieNevile"
	…

OPTIONAL query parts

SIEMENS

Optional parts in queries (Left Outer Join)

Example:

*"Give me all names of co-authors of Tim Berners-Lee and **optionally** their homepage"*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N ?H
WHERE {
       ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
       ?D foaf:maker ?CoAuth .
        ?CoAuth foaf:name ?N .
       OPTIONAL { ?CoAuth foaf:homepage ?H }
      }
```

Link

OPTIONAL query parts

SIEMENS

Optional parts in queries (Left Outer Join)

Example:

*"Give me all names of co-authors of Tim Berners-Lee and **optionally** their homepage"*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N ?H
WHERE {
       ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berne…-Lee>.
       ?D foaf:maker ?CoAuth .
        ?CoAuth foaf:name ?N .
       OPTIONAL { ?CoAuth foaf:homepage ?H }
      }
```

Another example where variables can be unbound in results!

N	H
"Lalana Kagal"	-
"Tim Berners-Lee"	<http://www.w3.org/People/Berners-Lee/>
"Dan Connolly"	-
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>
"m. c. schraefel"	<http://www.ecs.soton.ac.uk/~mc/>
"Paul André"	-
"Ryen White"	<http://www.dcs.gla.ac.uk/~whiter/>
"Desney S. Tan"	<http://research.microsoft.com/%7Edesney/>
"Tim Berners-Lee"	<http://www.w3.org/People/Berners-Lee/>
"Sunny Consolvo"	-

Link

FILTERING out query results

SIEMENS

FILTERs allow to specify FILTER conditions on patterns

Example:

*"Give me all names of co-authors of Tim Berners-Lee and **whose homepage starts with http://www.w3…**"*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N ?H
WHERE {
       ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
       ?D foaf:maker ?CoAuth .
       ?CoAuth foaf:name ?N .
       ?CoAuth foaf:homepage ?H .
       FILTER( regex( str(?H) , "^http://www.w3" ) )
      }
```

FILTERING out query results

SIEMENS

FILTERs allow to specify FILTER conditions on patterns

Example:

*"Give me all names of co-authors of Tim Berners-Lee and **whose homepage starts with http://www.w3…**"*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N ?H
WHERE {
       ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
       ?D foaf:maker ?CoAuth .
       ?CoAuth foaf:name ?N .
       ?CoAuth foaf:homepage ?H .
       FILTER( regex( str(?H) , "^http://www.w3" ) )
      }
```

N	H
"Tim Berners-Lee"	<http://www.w3.org/People/Berners-Lee/>
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>
"Tim Berners-Lee"	<http://www.w3.org/People/Berners-Lee/>
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>
"Tim Berners-Lee"	<http://www.w3.org/People/Berners-Lee/>
"Tim Berners-Lee"	<http://www.w3.org/People/Berners-Lee/>

FILTERING out query results

SIEMENS

FILTERs allow to specify FILTER conditions on patterns, and can be combined with &&, ||, !

Example:

*"Give me all names of co-authors of Tim Berners-Lee and **whose homepage starts with http://www.w3…" AND different from Tim B.-L. himself"***

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N ?H
WHERE {
       ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
       ?D foaf:maker ?CoAuth .
       ?CoAuth foaf:name ?N .
       ?CoAuth foaf:homepage ?H .
       FILTER( regex( str(?H) , "^http://www.w3" ) &&
       ?CoAuth != <http://dblp.l3s.de/…/authors/Tim_Berners-Lee> )
      }
```

N	H
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>

FILTERING out query results

SIEMENS

FILTERs allow to specify FILTER conditions on pattern

- Can use an extensible library of built-in functions
  - **checking**: bound(), isIRI(), isBlank(), regex() …
  - **Conversion/extraction:** str(), datatype(), lang() …
- Can be complex: && , ||, !
- ATTENTION: Evaluated in a 3-valued logic: **true, false, error**

A	B	A || B	A && B
T	T	T	T
T	F	T	F
F	T	T	F
F	F	F	F
T	E	T	E
E	T	T	E
F	E	E	F
E	F	E	F
E	E	E	E

A	!A
T	F
F	T
E	E

FILTERING out query results

SIEMENS

FILTERs allow to specify FILTER conditions on pattern

- Can use an extensible library of built-in functions
  - **checking**: bound(), isIRI(), isBlank(), regex() …
  - **Conversion/extraction:** str(), datatype(), lang() …
- Can be complex: && , ||, !
- ATTENTION: Evaluated in a 3-valued logic: **true, false, error**

A	B	A || B	A && B
T	T	T	T
T	F	T	F
F	T	T	F
F	F	F	F
T	E	T	E
E	T	T	E
F	E	E	F
E	F	E	F
E	E	E	E

Example FILTERs and OPTIONAL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N ?H
WHERE {
       ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
       ?D foaf:maker ?CoAuth . ?CoAuth foaf:name ?N .
       OPTIONAL { ?CoAuth foaf:homepage ?H . }
       FILTER( ! regex( str(?H) , "^http://www.w3" ) &&
       ?CoAuth != <http://dblp.l3s.de/…/authors/Tim_Berners-Lee> )
      }
```

A	!A
T	F
F	T
E	E

FILTERING out query results

SIEMENS

A	B	A || B	A && B
T	T	T	T
T	F	T	F
F	T	T	F
F	F	F	F
T	E	T	E
E	T	T	E
F	E	E	F
E	F	E	F
E	E	E	E

FILTERs allow to specify FILTER conditions on pattern

- Can use an extensible library of built-in functions
  - **checking**: bound(), isIRI(), isBlank(), regex() …
  - **Conversion/extraction:** str(), datatype(), lang() …
- Can be complex: && , ||, !
- ATTENTION: Evaluated in a 3-valued logic: **true, false, error**

Example FILTERs and OPTIONAL:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N ?H
WHERE {
        ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>.
        ?D foaf:maker ?CoAuth . ?CoAuth foaf:name ?N .
        OPTIONAL { ?CoAuth foaf:homepage ?H . }
        FILTER( ! regex( str(?H) , "^http://www.w3" ) &&
        ?CoAuth != <http://dblp.l3s.de/…/authors/Tim_Berners-Lee> )
      }
```

A	!A
T	F
F	T
E	E

Will result in E for unbound ?H
➔Whole FILTER expr
always E for unbound ?H
You don't get those without homepage

N	H
"m. c. schraefel"	<http://www.ecs.soton.ac.uk/~mc/>
"Ryen White"	<http://www.dcs.gla.ac.uk/~whiter/>
"Desney S. Tan"	<http://research.microsoft.com/%7Edesney/>
"Isaac S. Kohane"	<http://chip.org/~zak/>
"Gary Marsden"	<http://people.cs.uct.ac.za/~gaz/>

SIEMENS

FILTERING out query results

- **ATTENTION**: FILTERs can **NOT** assign/create new values…

```
PREFIX ex: <http://example.org/>
SELECT ?Item ?NewP
WHERE { ?Item ex:price ?Pr FILTER (?NewP = ?Pr + 10 ) }
```

SIEMENS

FILTERING out query results

- **ATTENTION**: FILTERs can **NOT** assign/create new values…

```
PREFIX ex: <http://example.org/>
SELECT ?Item ?NewP
WHERE { ?Item ex:price ?Pr FILTER (?NewP = ?Pr + 10 ) }
```

New “Non-safe” variable in FILTERs are considered unbound. The Filter will just always result in E
➔ Result always empty

- Obviously, common query languages like SQL can do this…

```
SELECT Item, Price+10 AS NewPrice  FROM Table
```

… FILTER in SPARQL is like WHERE in SQL,
but SPARQL1.0 doesn’t have asignment (AS)

SIEMENS

Datasets & Querying named GRAPHs 1/2

*Find me people who have been involved with **at least three** ISWC or ESWC conference events.*

(from SPARQL endpoint at data.semanticweb.org)

```
SELECT ?person WHERE {
       GRAPH ?g1 { ?person a foaf:Person }
       GRAPH ?g2 { ?person a foaf:Person }
       GRAPH ?g3 { ?person a foaf:Person }
       FILTER(?g1 != ?g2 && ?g1 != ?g3 && ?g2 != ?g3) . }
```

The GRAPH ?g construct allows a pattern to match against one of the named graphs in the **RDF dataset**.
The URI of the matching graph is bound to ?g (or whatever variable was actually used).
The FILTER assures that we're finding a person who occurs in three *distinct* graphs.

SIEMENS

Datasets & Querying named GRAPHs 2/2

An RDF dataset is a set $D = \{G,(G_1,N_{1)}, \ldots (G_n,N_n)\}$ consisting of

- *The **default graph** G*
- ***Named graphs**, i.e. pairs of Graphs and Names (G_n,N_n)*

D = {G} such that G = $G_{\texttt{polleres.net}} \cup G_{\texttt{gibbi.com}}$

SIEMENS

Datasets & Querying named GRAPHs 2/2

An RDF dataset is a set $D = \{G,(G_1,N_1), \ldots (G_n,N_n)\}$ *consisting of*

- *The* ***default graph*** G
- ***Named graphs****, i.e. pairs of Graphs and Names* (G_n,N_n)

The RDF dataset of a query can be

- *Implicitly given (like in the query on the last slide)*
- *Explicitly specified by* ***FROM, FROM NAMED*** *clauses*

(explicit specification of the dataset may be forbidden in case the dataset is fixed implicitly)

```
SELECT ?person
FROM <http://polleres.net/foaf.rdf>
FROM <http://gibbi.com/foaf.rdf>
WHERE {
        ?person a foaf:Person
      }
```

D = {G} such that G = $G_{\text{polleres.net}} \uplus G_{\text{gibbi.com}}$

SIEMENS

Datasets & Querying named GRAPHs 2/2

An RDF dataset is a set $D = \{G,(G_1,N_1), \ldots (G_n,N_n)\}$ consisting of

- *The **default graph** G*
- ***Named graphs**, i.e. pairs of Graphs and Names (G_n,N_n)*

The RDF dataset of a query can be

- *Implicitly given (like in the query on the last slide)*
- *Explicitly specified by **FROM, FROM NAMED** clauses*

(explicit specification of the dataset may be forbidden in case the dataset is fixed implicitly)

```
SELECT ?person
FROM <http://polleres.net/foaf.rdf>
FROM <http://gibbi.com/foaf.rdf>
WHERE {
        ?person a foaf:Person
      }
```

If given explicitly via FROM/FROM NAMED clauses the default graph of the Dataset is the RDF merge of the graphs given in FROM clauses

$D = \{G\}$ such that $G =$ $G_{\text{polleres.net}} \uplus G_{\text{gibbi.com}}$

SIEMENS

Datasets & Querying named GRAPHs 2/2

An RDF dataset is a set $D = \{G,(G_1,N_1), \ldots (G_n,N_n)\}$ consisting of

- *The **default graph** G*
- ***Named graphs**, i.e. pairs of Graphs and Names (G_n,N_n)*

The RDF dataset of a query can be

- *Implicitly given (like in the query on the last slide)*
- *Explicitly specified by **FROM, FROM NAMED** clauses*

(explicit specification of the dataset may be forbidden in case the dataset is fixed implicitly)

```
SELECT ?G ?person
FROM NAMED <http://polleres.net/foaf.rdf>
FROM NAMED <http://gibbi.com/foaf.rdf>
WHERE {
        GRAPH ?G  { ?person a foaf:Person }
      }
```

$D = \{G,$ ($G_{\text{polleres.net}}$,http://polleres.net/foaf.rdf), ($G_{\text{gibi.com}}$,http://gibbi.com/foaf.rdf) $\}$
such that $G = \{\}$

SIEMENS

Datasets & Querying named GRAPHs 2/2

An RDF dataset is a set $D = \{D,(G_1,N_1), \ldots (G_n,N_n)\}$ consisting of

- *The **default graph** G*
- ***Named graphs**, i.e. pairs of Graphs and Names (G_n,N_n)*

The RDF dataset of a query can be

- *Implicitly given (like in the query on the last slide)*
- *Explicitly specified by **FROM, FROM NAMED** clauses*

(explicit specification of the dataset may be forbidden in case the dataset is fixed implicitly)

```
SELECT ?G ?person
FROM NAMED <http://polleres.net/foaf.rdf>
FROM NAMED <http://gibbi.com/foaf.rdf>
WHERE {
        { ?person a foaf:Person }
      }
```

$D = \{G,$ ($G_{\text{polleres.net}}$,http://polleres.net/foaf.rdf), ($G_{\text{gibi.com}}$,http://gibbi.com/foaf.rdf) $\}$
such that $G = \{\}$

SIEMENS

Datasets & Querying named GRAPHs 2/2

An RDF dataset is a set $D = \{D,(G_1,N_1), \ldots (G_n,N_n)\}$ consisting of

- *The **default graph** G*
- ***Named graphs**, i.e. pairs of Graphs and Names (G_n,N_n)*

The RDF dataset of a query can be

- *Implicitly given (like in the query on the last slide)*
- *Explicitly specified by **FROM, FROM NAMED** clauses*

(explicit specification of the dataset may be forbidden in case the dataset is fixed implicitly)

```
SELECT ?G ?person
FROM NAMED <http://polleres.net/foaf.rdf>
FROM NAMED <http://gibbi.com/foaf.rdf>
WHERE {
        { ?person a foaf:Person }
      }
```

Note: No results, since Default Graph is empty.

D = {*G,* ($G_{polleres.net}$,http://polleres.net/foaf.rdf), ($G_{gibi.com}$,http://gibbi.com/foaf.rdf) }
such that G = {}

SIEMENS

Datasets & Querying named GRAPHs 2/2

An RDF dataset is a set $D = \{G,(G_1,N_1), \ldots (G_n,N_n)\}$ consisting of

- *The **default graph** G*
- ***Named graphs**, i.e. pairs of Graphs and Names (G_n,N_n)*

The RDF dataset of a query can be

- *Implicitly given (like in the query on the last slide)*
- *Explicitly specified by **FROM, FROM NAMED** clauses*

(explicit specification of the dataset may be forbidden in case the dataset is fixed implicitly)

```
SELECT ?G ?person
FROM NAMED <http://polleres.net/foaf.rdf>
FROM NAMED <http://gibbi.com/foaf.rdf>
WHERE {
 GRAPH <http://www.w3.org/timbl/Card> { ?person a foaf:Person }
       }
```

$D = \{G,$ ($G_{\text{polleres.net}}$,http://polleres.net/foaf.rdf), ($G_{\text{gibi.com}}$,http://gibbi.com/foaf.rdf) $\}$
such that $G = \{\}$

SIEMENS

Datasets & Querying named GRAPHs 2/2

An RDF dataset is a set $D = \{G,(G_1,N_1), \ldots (G_n,N_n)\}$ *consisting of*

- *The* ***default graph*** G
- ***Named graphs****, i.e. pairs of Graphs and Names* (G_n,N_n)

The RDF dataset of a query can be

- *Implicitly given (like in the query on the last slide)*
- *Explicitly specified by* ***FROM, FROM NAMED*** *clauses*

(explicit specification of the dataset may be forbidden in case the dataset is fixed implicitly)

```
SELECT ?G ?person
FROM NAMED <http://polleres.net/foaf.rdf>
FROM NAMED <http://gibbi.com/foaf.rdf>
WHERE {
 GRAPH <http://www.w3.org/timbl/Card> { ?person a foaf:Person }
       }
```

Note: No results, since there is no graph named http://w3.org/timbl/Card in the dataset

$D = \{G,$ ($G_{\text{polleres.net}}$,http://polleres.net/foaf.rdf), ($G_{\text{gibi.com}}$,http://gibbi.com/foaf.rdf) $\}$
such that $G = \{\}$

Slicing and Dicing results

SIEMENS

Solution Modifiers

- ORDER BY
- LIMIT/OFFSET

Graph G:

```
:p1 a foaf:Person.
:p2 a foaf:Person.
…
:p100 a foaf:Person.
```

Example:

```
SELECT ?P
FROM <G>
WHERE {  ?P foaf:Person }
   ORDER BY ?P
   LIMIT 5
```

?P
:p1
:p2
:p3
:p4
:p5

Slicing and Dicing results

SIEMENS

Graph G:

```
:p1 a foaf:Person.
:p2 a foaf:Person.
…
:p100 a foaf:Person.
```

Solution Modifiers

- ORDER BY
- LIMIT/OFFSET

Example:

```
SELECT ?P
FROM <G>
WHERE { ?P foaf:Person }
   ORDER BY ?P
   LIMIT 5
   OFFSET 5
```

?P
:p6
:p7
:p8
:p9
:p10

Slicing and Dicing results

SIEMENS

Graph G:

```
:p1 a foaf:Person.
:p2 a foaf:Person.
…
:p100 a foaf:Person.
```

Solution Modifiers

- ORDER BY
- LIMIT/OFFSET

Example:

```
SELECT ?P
FROM <G>
WHERE { ?P foaf:Person }

   LIMIT 5
   OFFSET 5
```

?P
:p8
:p2
:p99
:p20
:p42

Slicing and Dicing results

SIEMENS

Solution Modifiers

- ORDER BY
- LIMIT/OFFSET

Example:

```
SELECT ?P
FROM <G>
WHERE { ?P foaf:Person }

   LIMIT 5
   OFFSET 5
```

Graph G:

```
:p1 a foaf:Person.
:p2 a foaf:Person.
…
:p100 a foaf:Person.
```

ATTENTION! Order is Important when LIMIT and OFFSET are used to achieve deterministic results!

?P
:p8
:p2
:p99
:p20
:p42

SIEMENS

More complex query examples 1/2

"IF-THEN-ELSE"

- *"Give me the names of persons, if it exists, otherwise the nicknames, if it exists, otherwise the labels"*

```
SELECT ?X ?N
WHERE{ ?X rdf:type foaf:Person
         OPTIONAL { ?X foaf:name ?N }
         OPTIONAL { ?X foaf:nickname ?N }
         OPTIONAL { ?X rdfs:label ?N } }
```

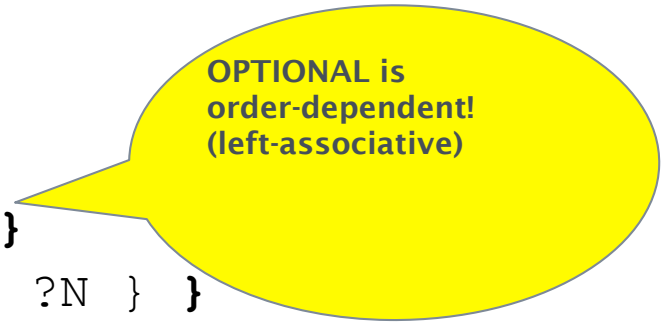
SIEMENS

More complex query examples 1/2

"IF-THEN-ELSE"

- *"Give me the names of persons, if it exists, otherwise the nicknames, if it exists, otherwise the labels"*

```
SELECT ?X ?N
WHERE{ { { ?X rdf:type foaf:Person
           OPTIONAL { ?X foaf:name ?N } }
             OPTIONAL { ?X foaf:nickname ?N } }
                OPTIONAL { ?X rdfs:label ?N } }
```



SIEMENS

More complex query examples 1/2

"IF-THEN-ELSE"

- *"Give me the names of persons, if it exists, otherwise the nicknames, if it exists, otherwise the labels"*

```
SELECT ?X ?N
WHERE{ { { ?X rdf:type foaf:Person
           OPTIONAL { ?X foaf:name ?N } }
             OPTIONAL { ?X foaf:nickname ?N } }
                OPTIONAL { ?X rdfs:label ?N } }
```

"Conditional OPTIONAL"

- *"Give me the names, and the homepage **only of those whose name starts with 'D'** "*

```
SELECT  ?N ?H
WHERE{ ?X foaf:name ?N
           OPTIONAL { ?X foaf:homepage ?H
                      FILTER ( regex( str(?N), "^D" ) ) }
     }
```

SIEMENS

More complex query examples 1/2

"IF-THEN-ELSE"

- *"Give me the names of persons, if it exists, otherwise the nicknames, if it exists, otherwise the labels"*

```
SELECT ?X ?N
WHERE{ { { ?X rdf:type foaf:Person
           OPTIONAL { ?X foaf:name ?N } }
             OPTIONAL { ?X foaf:nickname ?N } }
               OPTIONAL { ?X rdfs:label ?N } }
```

OPTIONAL is order-dependent! OPTIONAL is NOT modular/compositional! (?N is not considered unsafe in this FILTER)*

"Conditional OPTIONAL"

- *"Give me the names, and the homepage **only of those whose name sta s with 'D'** "*

```
SELECT  ?N ?H
WHERE{ ?X foaf:name ?N
         OPTIONAL { ?X foaf:homepage ?H
                    FILTER ( regex( str(?N), "^D" ) ) }
     }
```

Non-compositionality raised some eyebrows… [Angles&Gutierrez, 2008] showed that compositional semantics can be achieved by rewriting.

SIEMENS

More complex query examples 2/2

Negation is doable in SPARQL1.0:

- *"Give me all Persons without a homepage"*
- **Option 1:** by combination of OPTIONAL and FILTER(!bound(…))

```
SELECT ?X
WHERE{ ?X rdf:type foaf:Person
         OPTIONAL { ?X foaf:homepage ?H }
       FILTER( !bound( ?H ) ) }
```

SIEMENS

More complex query examples 2/2

Negation is doable in SPARQL1.0:

- *"Give me all Persons without a homepage"*
- **Option 1:** by combination of OPTIONAL and FILTER(!bound(…))

```
SELECT ?X
WHERE{ ?X rdf:type foaf:Person
         OPTIONAL { ?X foaf:homepage ?H }
       FILTER( !bound( ?H ) ) }
```

- **Option 2:** by another weird combination of OPTIONAL with GRAPH queries…

```
SELECT ?X
WHERE{ ?X rdf:type foaf:Person
         OPTIONAL { ?X foaf:homepage ?H }
       GRAPH boundcheck.ttl {?H :is :unbound } }
```

where the aux. graph `boundcheck.ttl` *contains the single triple* `[] :is :unboud.`

SIEMENS

More complex query examples 2/2

Negation is doable in SPARQL1.0:

- *"Give me all Persons without a homepage"*
- **Option 1:** by combination of OPTIONAL and FILTER(!bound(…))

```
SELECT ?X
WHERE{ ?X rdf:type foaf:Person
         OPTIONAL { ?X foaf:homepage ?H }
       FILTER( !bound( ?H ) ) }
```

- **Option 2:** by another weird combination of OPTIONAL with GRAPH queries…

```
SELECT ?X
WHERE{ ?X rdf:type foaf:Person
         OPTIONAL { ?X foaf:homepage ?H }
       GRAPH boundcheck.ttl {?H :is :unbound } }
```

where the aux. graph `boundcheck.ttl` *contains the single triple* `[] :is :unboud.`

BTW: It's more convenient to express set difference in SPARQL1.1 (but there are still a lot of SPARQL endpoints running in SPARQL1.0)

SPARQL's formal semantics

SIEMENS

SPARQL1.0 Formal Semantics

SIEMENS

Graph patterns

- BGPs
- *P1* . *P2*
- *P* `FILTER` *R*
- *P1* `UNION` *P2*
- *P1* `OPTIONAL` *P2*
- `GRAPH` *G* { *P* }

Semantics

- *eval(P, D(G))* …

P is a graph pattern

recursively defined for all graph patterns in Section 12.5 of

http://www.w3.org/TR/rdf-sparql-query/

D is a dataset, G is the “active graph”

Just as for RDF, the spec. semantics is a bit hard to read …

Formal Semantics á la [Perez et al. 2006]

Easier to explain… let's steal from that here and explain the diffs:

Definition 1:

The evaluation of the BGP P over a graph G, denoted by eval(P,G), is the set of all mappings μ: *Var* → *V(G)* such that:

$dom(\mu)$ is exactly the set of variables occurring in P

$\mu(P) \subseteq G$

Example Graph:

```
:tim        foaf:knows    :jim .
:jim        foaf:knows    :tim .
:tim        foaf:knows    :lalana .
```

Example Pattern:

```
P = { ?X  foaf:knows    ?Y }.
```

eval(P,G) = { $\mu1 = \{ ?x \rightarrow$:tim , $?y \rightarrow$:jim $\}$, $\mu2 = \{ ?x \rightarrow$:jim , $?y \rightarrow$:tim $\}$,
$\mu3 = \{ ?x \rightarrow$:tim , $?y \rightarrow$:lalana } }

SIEMENS

Compatible mappings:

Definition 2:
mappings $\mu 1$, $\mu 2$ are **compatible** iff they agree in their shared variables.

not compatible:

$\mu 1 = \{ ?x \rightarrow$ `:tim` , $?y \rightarrow$ `:jim` $\}$ $\mu 2 = \{ ?x \rightarrow$ `:tim` , $?y \rightarrow$ `:lalana` $\}$

compatible:

$\mu 1 = \{ ?x \rightarrow$ `:tim` $\}$ $\mu 2 = \{ ?x \rightarrow$ `:tim` , $?y \rightarrow$ `:lalana` $\}$

compatible?

$\mu 1 = \{ ?x \rightarrow$ `:tim` $\}$ $\mu 2 = \{ ?y \rightarrow$ `:lalana` $\}$

SIEMENS

Algebra á la [Perez et al. 2006]

Definition 2:
mappings $\mu 1$, $\mu 2$ are **compatible** iff they agree in their shared variables.

Let M1, M2 be sets of mappings

Definition 3:

Join:
$M1 \bowtie M2 = \{ \mu 1 \cup \mu 2 \mid \mu 1 \in M1, \mu 2 \in M2, \text{and } \mu 1, \mu 2 \text{ are compatible} \}$

Union:
$M1 \cup M2 = \{ \mu \mid \mu \in M1 \text{ or } \mu \in M2 \}$

Diff:
$M1 \setminus M2 = \{ \mu \in M1 \mid \text{forall } \mu' \in M2, \mu \text{ and } \mu' \text{ are not compatible} \}$

LeftJoin:
$M1 \mathbin{⟕} M2 = (M1 \bowtie M2) \cup (M1 \setminus M2)$

Filter:
$M|_R = \{ \mu \mid \mu \in M \text{ and } \mu(R) = \text{true} \}$

SIEMENS

Algebra á la [Perez et al. 2006]

Definition 2:
mappings $\mu 1$, $\mu 2$ are **compatible** iff they agree in their shared variables.

Let M1, M2 be sets of mappings

Definition 3:

P1 . *P2*

Join:
$M1 \bowtie M2 = \{ \mu 1 \cup \mu 2 \mid \mu 1 \in M1, \mu 2 \in M2, \text{ and } \mu 1, \mu 2 \text{ are compatible} \}$

P1 UNION *P2*

Union:
$M1 \cup M2 = \{ \mu \mid \mu \in M1 \text{ or } \mu \in M2 \}$

Diff:
$M1 \setminus M2 = \{ \mu \in M1 \mid \text{forall } \mu' \in M2, \mu \text{ and } \mu' \text{ are not compatible} \}$

P1 OPTIONAL *P2*

LeftJoin:
$M1 \bowtie M2 = (M1 \bowtie M2) \cup (M1 \setminus M2)$

P FILTER *R*

Filter:
$M|_R = \{ \mu \mid \mu \in M \text{ and } \mu(R) = \text{true} \}$

SIEMENS

Semantics full as per [Perez et al.2006]

eval(BGP,G) … *see* **Definition 1**

eval(P1 . P2,G) = *eval(P1, G)* $\bowtie$ *eval(P2, G)*

eval(P1 `UNION` *P2,G)* = *eval(P1, G)* $\cup$ *eval(P2, G)*

eval(P1 `OPTIONAL` *P2, G)* = *eval(P1, G)* $⟕$ *eval(P2, G)*

eval(P `FILTER` *R,G)* = *eval(P, G)* $|_R$

SIEMENS

Semantics full as per [Perez et al.2006]

eval(BGP,G) … *see* **Definition 1**

eval(P1 . P2,G) = *eval(P1, G)* $\bowtie$ *eval(P2, G)*

eval(P1 `UNION` *P2,G)* = *eval(P1, G)* $\cup$ *eval(P2, G)*

eval(P1 `OPTIONAL` *P2, G)* = *eval(P1, G)* $⟕$ *eval(P2, G)*

eval(P `FILTER` *R,G)* = *eval(P, G)* $|_R$

As for `GRAPH` patterns, the extension from *eval(P,G)* to *eval(P,D(G))* is straightforward:

Let $D = \{G,(G_1,N_1), \ldots (G_n,N_n)\}$

- for top query patterns evaluation is equal to eval(P, G),
- eval(GRAPH N_i P, D(G)) = eval(P, G_i) if (G_i,N_i) in D, and {} otherwise
- eval(GRAPH *?Var* P, D(G)) = $\bigcup_{\text{for all } (G_i,N_i) \text{ in } D}$ eval(P, G_i) ∪ { {?Var → N_i } }

ISSUE 1) Recall from before:

SPARQL allows duplicates !

SIEMENS

Unions of conjunctive queries

Example:

*"Give me all names of co-authors **or** friends of Tim Berners-Lee"*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N
WHERE {
        { [ foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Le >,
                       [ foaf:name ?N ] ] . }
        UNION
        { <http://www.w3.org/People/Berners-Lee/card#i>
          ?F foaf:name ?N }
      }
```

Note: Duplicates possible, **bag/multiset** semantics!

?N
"Lalana Kagal"
"Tim Berners-Lee"
"Dan Connolly"
"Jim Hendler"
…
"Michael Hausenblas"
"Jim Hendler"
"Charles McCathieNevile"
…

ISSUE 2) Similar troubles with duplicates from projection & What about blank nodes in patterns?

SIEMENS

Example from before, but now with projection:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?X
FROM <graph>
WHERE { ?X foaf:knows ?Y }
```

graph:

```
:tim        foaf:knows    :jim .
:jim        foaf:knows    :tim .
:tim        foaf:knows    :lalana .
```

?X
:tim
:jim
:tim

ISSUE 2) Similar troubles with duplicates from projection & What about blank nodes in patterns?

SIEMENS

Example from before, but now with projection:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?X
FROM <graph>
WHERE { ?X foaf:knows _:Y }
```

Blank nodes "behave" like (non-distinguished) variables in BGPs

graph:

```
:tim        foaf:knows    :jim .
:jim        foaf:knows    :tim .
:tim        foaf:knows    :lalana .
```

?X
:tim
:jim
:tim

ISSUE 2) Similar troubles with duplicates from projection & What about blank nodes in patterns?

SIEMENS

Example from before, but now with projection:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?X
FROM <graph>
WHERE { ?X foaf:knows [] }
```

Blank nodes "behave" like (non-distinguished) variables in BGPs

graph:

```
:tim        foaf:knows    :jim .
:jim        foaf:knows    :tim .
:tim        foaf:knows    :lalana .
```

?X
:tim
:jim
:tim

ISSUE 3) Recall from before:
FILTERS can make OPTIONAL non-compositional!

SIEMENS

- "Conditional OPTIONAL"
  - *"Give me the names and only of those whose name starts with 'D' the homepage"*

```
SELECT  ?N ?H
WHERE{  ?X foaf:name ?N
            OPTIONAL { ?X foaf:homepage ?H
                       FILTER ( regex( str(?N), "^D" ) ) }
        }
```

OPTIONAL is NOT modular/compositional! (?N is not considered unsafe in this FILTER)*

N	H
"Lalana Kagal"	-
"Tim Berners-Lee"	-
"Dan Connolly"	-
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>
"m. c. schraefel"	-
"Paul André"	-
"Ryen White"	-
"Desney S. Tan"	<http://research.microsoft.com/%7Edesney/>
"Tim Berners-Lee"	-
"Sunny Consolvo"	-

Adapting [Perez et al. 2006]

ISSUE 1) Algebra operations need to be adapted to multiset/bag semantics:

ISSUE 3) non-compositionality of FILTERs in OPTIONAL

Definition 3:

Join:
$M1 \bowtie M2 = \{ \mu 1 \cup \mu 2 \mid \mu 1 \in M1, \mu 2 \in M2, \text{ and } \mu 1, \mu 2 \text{ are compatible}\}$

Union:
$M1 \cup M2 = \{ \mu \mid \mu \in M1 \text{ or } \mu \in M2\}$

Diff:
$M1 \setminus M2 = \{\mu \in M1 \mid \text{forall } \mu' \in M2, \mu \text{ and } \mu' \text{ are not compatible } \}$

LeftJoin:
$M1 \mathbin{⟕} M2 = (M1 \bowtie M2) \cup (M1 \setminus M2)$

Filter:
$M|_R = \{ \mu \mid \mu \in M \text{ and } \mu(R) = \text{true}\}$

SIEMENS

Adapting [Perez et al. 2006]

ISSUE 1) Algebra operations need to be adapted to multiset/bag semantics:

Definition 3:

Join:

$$M1 \bowtie M2 = \{\!\{ \mu 1 \cup \mu 2 \mid \mu 1 \in M1, \mu 2 \in M2, \text{ and } \mu 1, \mu 2 \text{ are compatible} \}\!\}$$

Union:

$$M1 \cup M2 = \{\!\{ \mu \mid \mu \in M1 \text{ or } \mu \in M2 \}\!\}$$

Diff:

$$M1 \setminus M2 = \{\!\{ \mu \in M1 \mid \text{forall } \mu' \in M2, \mu \text{ and } \mu' \text{ are not compatible} \}\!\}$$

LeftJoin:

$$M1 \bowtie\!\!\!\!- M2 = (M1 \bowtie M2) \cup (M1 \setminus M2)$$

Filter:

$$M|_R = \{ \mu \mid \mu \in M \text{ and } \mu(R) = \text{true} \}$$

ISSUE 3) non-compositionality of FILTERs in OPTIONAL

SIEMENS

Adapting [Perez et al. 2006]

ISSUE 1) Algebra operations need to be adapted to multiset/bag semantics:

ISSUE 3) non-compositionality of FILTERs in OPTIONAL

Definition 3:

Join:

$M1 \bowtie M2 = \{\!\{ \mu 1 \cup \mu 2 \mid \mu 1 \in M1, \mu 2 \in M2, \text{ and } \mu 1, \mu 2 \text{ are compatible} \}\!\}$

Union:

$M1 \cup M2 = \{\!\{ \mu \mid \mu \in M1 \text{ or } \mu \in M2 \}\!\}$

Diff:

$M1 \setminus M2 = \{\!\{ \mu \in M1 \mid \text{forall } \mu' \in M2, \mu \text{ and } \mu' \text{ are not compatible} \}\!\}$

LeftJoin:

~~$M1 \bowtie M2 = (M1 \bowtie M2) \cup (M1 \setminus M2)$~~

Filter:

$M|_R = \{ \mu \mid \mu \in M \text{ and } \mu(R) = \text{true} \}$

SIEMENS

Semantics as per SPARQL1.0 spec:

eval(BGP,G) … *see* **Definition 1**

eval(P1 . P2,G) = *eval(P1, G)* $\bowtie$ *eval(P2, G)*

eval(P1 `UNION` *P2,G)* = *eval(P1, G)* ∪ *eval(P2, G)*

eval(P `FILTER` *R,G)* = *eval(P, G)* $|_R$

eval(P1 `OPTIONAL` {*P2* `FILTER` *R*} *, G) consists of all* μ such that*:*

1. μ = μ1 ∪ μ2, such that
   μ1 ∈ eval(P1,G) and μ2 ∈ eval(P2,G) are compatible and μ(R) = true, or
2. μ ∈ eval(P1,G) and
   there is no compatible μ2 ∈ eval(P2,G) for μ, or
3. μ ∈ eval(P1,G) and
   for any compatible μ2 ∈ eval(P2,G), μ ∪ μ2 does not satisfy R.

SIEMENS

Semantics as per SPARQL1.0 spec:

eval(BGP,G) … *see* **Definition 1**

eval(P1 . P2,G) = *eval(P1, G)* $\bowtie$ *eval(P2, G)*

eval(P1 `UNION` *P2,G)* = *eval(P1, G)* ∪ *eval(P2, G)*

eval(P `FILTER` *R,G)* = *eval(P, G)* $|_R$

eval(P1 `OPTIONAL` {*P2* `FILTER` *R*} *, G) consists of all* μ such that*:*

1. μ = μ1 ∪ μ2, such that
   μ1 ∈ eval(P1,G) and μ2 ∈ eval(P2,G) are compatible and μ(R) = true, or
2. μ ∈ eval(P1,G) and
   there is no compatible μ2 ∈ eval(P2,G) for μ, or
3. μ ∈ eval(P1,G) and
   for any compatible μ2 ∈ eval(P2,G), μ ∪ μ2 does not satisfy R.

Addresses **ISSUE 3)** non-compositionality of FILTERs in OPTIONAL

SIEMENS

What about Issue 2?

Definition 1:

The evaluation of the BGP P over a graph G, denoted by eval(P,G), is the set of all mappings μ: *Var* → *V(G)* such that:

$dom(\mu)$ is exactly the set of variables occurring in P

$\mu(P) \subseteq G$

Example Graph:

```
:tim      foaf:knows    :jim .
:jim      foaf:knows    :tim .
:tim      foaf:knows    :lalana .
```

Example Pattern:

```
P = { ?X  foaf:knows    ?Y }.
```

eval(P,G) = { $\mu1 = \{ ?x \rightarrow$:tim , $?y \rightarrow$:jim $\}$, $\mu2 = \{ ?x \rightarrow$:jim , $?y \rightarrow$:tim $\}$,
$\mu3 = \{ ?x \rightarrow$:tim , $?y \rightarrow$:lalana }

SIEMENS

What about Issue 2?

Definition 1:

The evaluation of the BGP P over a graph G, denoted by eval(P,G), is the set of all mappings μ: *Var* → *V(G)* such that:

$dom(\mu)$ is exactly the set of variables occurring in P

$\mu(P) \subseteq G$

Example Graph:

```
:tim        foaf:knows    :jim .
:jim        foaf:knows    :tim .
:tim        foaf:knows    :lalana .
```

Example Pattern:

```
P = { ?X  foaf:knows    _:Y }.
```

```
eval(P,G) = {}
```

SIEMENS

What about Issue 2?

Definition 1:

The evaluation of the BGP P over a graph G, denoted by eval(P,G), is the **multiset** of all mappings μ: *Var* → *V(G)* such that:

dom(μ) is exactly the set of variables occurring in P

$\mu(P) \models G$

Example Graph:

```
:tim      foaf:knows    :jim .
:jim      foaf:knows    :tim .
:tim      foaf:knows    :lalana .
```

Example Pattern:

```
P = { ?X  foaf:knows   _:Y }.
```

eval(P,G) = {{ $\mu 1 = \{ ?x \rightarrow$:tim $\}$, $\mu 2 = \{ ?x \rightarrow$:jim$\}$,
$\mu 3 = \{ ?x \rightarrow$:tim $\}$ }}

SIEMENS

What about Issue 2?

Addresses **Issue 2)** We mean here the multiset wrt. to the number of existing instance mappings wrt. **Simple Entailment**

Definition 1:

The evaluation of the BGP P over a graph G, denoted by eval(P,G), is the **multiset** of all mappings μ: $Var \rightarrow V(G)$ such that:

$dom(\mu)$ is exactly the set of variables occurring in P

$\mu(P) \models G$

Example Graph:

```
:tim      foaf:knows   :jim .
:jim      foaf:knows   :tim .
:tim      foaf:knows   :lalana .
```

Example Pattern:

```
P = { ?X  foaf:knows   _:Y }.
```

eval(P,G) = {{ $\mu 1 = \{ ?x \rightarrow$:tim $\}$, $\mu 2 = \{ ?x \rightarrow$:jim$\}$,
$\mu 3 = \{ ?x \rightarrow$:tim $\}$ }}

SIEMENS

Some academic works around SPARQL1.0

SPARQL semantics

- [Perez et al. 2006] (pre-dates the spec) [Perez et al. 2009]

SPARQL equivalences

- also in [Perez et al. 2006],[Perez et al. 2009]
- More in [Schmidt et al. 2010]

SPARQL expressivity

- Reducible to Datalog with negation [Polleres 2007]
- Other way around also works [Angles &Gutierrez 2008]

SIEMENS

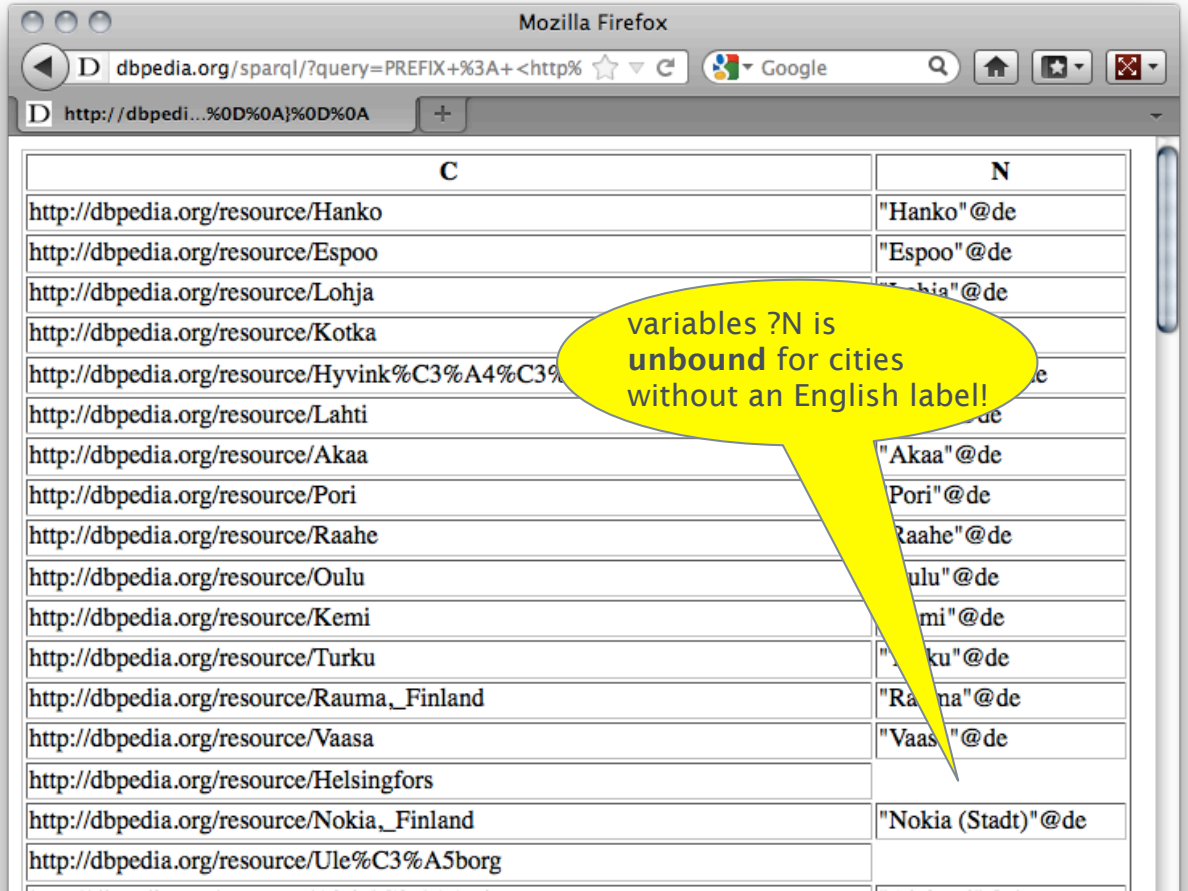
Another example from DBPedia (time allowed…)

Conjunction (.) , disjunction (UNION), **optional (OPTIONAL)** patterns and filters (FILTER)

SIEMENS

Cites in Finland with optionally their German (@de) name

```
SELECT ?C ?N
WHERE
{
 ?C  dcterms:subject category:Cities_and_towns_in_Finland .
 OPTIONAL { ?C rdfs:label ?N . FILTER( LANG(?N) = "de" ) }
}
```



C	N
http://dbpedia.org/resource/Hanko	"Hanko"@de
http://dbpedia.org/resource/Espoo	"Espoo"@de
http://dbpedia.org/resource/Lohja	
http://dbpedia.org/resource/Kotka	
http://dbpedia.org/resource/Hyvink%C3%A4%C3%	
http://dbpedia.org/resource/Lahti	
http://dbpedia.org/resource/Akaa	"Akaa"@de
http://dbpedia.org/resource/Pori	Pori"@de
http://dbpedia.org/resource/Raahe	Raahe"@de
http://dbpedia.org/resource/Oulu	ulu"@de
http://dbpedia.org/resource/Kemi	mi"@de
http://dbpedia.org/resource/Turku	ku"@de
http://dbpedia.org/resource/Rauma,_Finland	"Ra na"@de
http://dbpedia.org/resource/Vaasa	"Vaas "@de
http://dbpedia.org/resource/Helsingfors	
http://dbpedia.org/resource/Nokia,_Finland	"Nokia (Stadt)"@de
http://dbpedia.org/resource/Ule%C3%A5borg	

BTW, why does “Helsingfors” not have a German label?

SIEMENS

Helsingfors is the Swedisch name of of Helsinki, and only exists in dbpedia as a redirect to Helsinki:

property	hasValue	isValueOf
owl:sameAs	-	<http://www4.wiwiss.fu-berlin.de/flickrwrappr/photos/Helsingfors>
dbpedia:ontology/deathPlace	-	:Ossian_Schauman
dbpedia:ontology/birthPlace	-	:Lydia_Chukovskaya
dbpedia:ontology/birthPlace	-	:Olav_Ri%C3%A9go
foaf:primaryTopic	-	<http://en.wikipedia.org/wiki/Helsingfors>
rdfs:label	"Helsingfors"@en	-
dbpedia:ontology/wikiPageRedirects	:Helsinki	-
<http://purl.org/dc/terms/subject>	:Category:Cities_and_towns_in_Finland	-
foaf:page	<http://en.wikipedia.org/wiki/Helsingfors>	-

We can use the IF-THEN-ELSE feature of **OPTIONAL** to fix this query, can't we?

SIEMENS

Cites Finland with optionally (if they have one) their German (@de) name … and otherwise try to find whether there is a redirect to a resource with a German name

```
SELECT ?C ?N
WHERE
{
 ?C  dcterms:subject category:Cities_and_towns_in_Finland .
 OPTIONAL { ?C rdfs:label ?N . FILTER( LANG(?N) = "de" ) }
 OPTIONAL { ?C dbont:wikiPageRedirects  [rdfs:label ?N] . FILTER( LANG(?N) = ”de" ) }
}
```

Link

Unfortunately doesn't work as intended on DBpedia SPARQL endpoint, cf.

https://twitter.com/#!/AxelPolleres/status/189257251154960384

Assignments

Assignment 2 to be submitted by Wednesday

Assignment 3 will follow on Friday

SIEMENS

References

SPARQL1.0 Specification: http://www.w3.org/TR/rdf-sparql-query/ (2008)

[Perez et al. 2006] Jorge Pérez, Marcelo Arenas, Claudio Gutierrez: Semantics and Complexity of SPARQL. International Semantic Web Conference 2006: 30-43

[Perez et al. 2009] Jorge Pérez, Marcelo Arenas, Claudio Gutierrez: Semantics and complexity of SPARQL. ACM Trans. Database Syst. 34(3): (2009)

[Schmidt et al. 2010] Michael Schmidt, Michael Meier, Georg Lausen: Foundations of SPARQL query optimization. ICDT 2010: 4-33

[Polleres 2007] Axel Polleres: From SPARQL to rules (and back). WWW 2007: 787-796

[Angles &Gutierrez 2008] Renzo Angles, Claudio Gutierrez: The Expressive Power of SPARQL. International Semantic Web Conference 2008: 114-129

[Perez et al. 2008] Jorge Pérez, Marcelo Arenas, Claudio Gutierrez: nSPARQL: A Navigational Language for RDF. International Semantic Web Conference 2008: 66-81