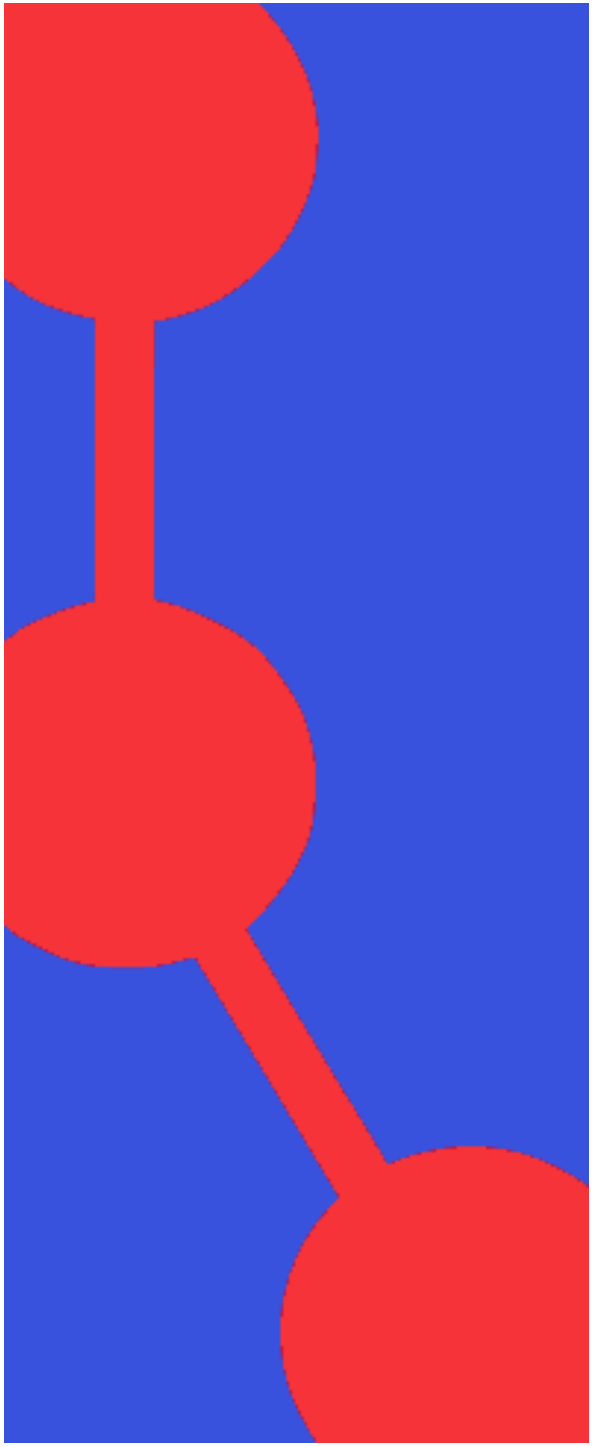


SPARQL1.1: An introduction

AxelPolleres
Digital Enterprise Research Institute, National University of Ireland, Galway

Edna Ruckhaus
(ruckhaus@delicias.dia.fi.upm.es, ruckhaus@ldc.usb.ve)
Facultad de Informática
Universidad Politécnica de Madrid

Dpto. De Computación y T.I.
Universidad Simón Bolívar, Venezuela

What is SPARQL?

- Loads of **structured data** out there
- You want to do **structured queries** on top of it …

Query Language for RDF

- SQL “look-and-feel” for the Semantic Web
- Means to query the Web of Data
- Means to map between vocabularies
- Means to access RDF stores

SPARQL1.0 (standard since 2008):

- **Query Language**
- Protocol
- Result Format

SPARQL1.1 (in progress):

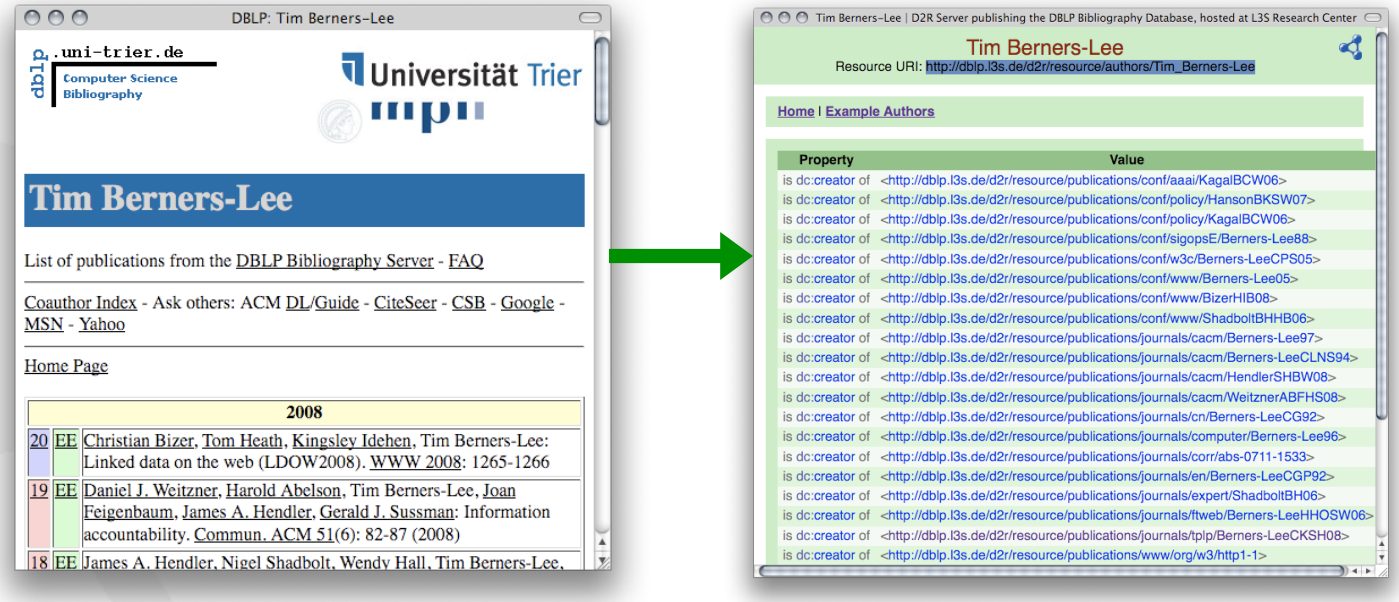
- **SPARQL 1.1 query language (additional features: aggregate functions, subqueries, negation, project expressions, property paths, basic federated queries)**
- **SPARQL 1.1 Entailment regimes**
- SPARQL 1.1 Update: A full data manipulation language

RDF Data online: Example 2/2

(i) directly by the publishers

(ii) by exporters, e.g. D2R and friends, RDFa exporters, etc.

e.g. L3S' RDF export of the DBLP citation index, using FUB's D2R (http://dblp.l3s.de/d2r/)



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!

SPARQL Query Language

Focused on pattern matching on the RDF graph

"Elections and voters that have not voted in those elections"

```
PREFIX vote:<tag:govshare.info,2005:rdf/vote/>
        SELECT ?Eleccion, ?Persona
            FROM <votes.
    WHERE  {?Rep vote:voter ?Persona .
            ?Rep vote:option "NoVote".
          ?Eleccion vote:hasBallot ?Rep}
```

Namespaces

Projected attributes

RDF datasets

Basic Graph Pattern
Set of triple patterns

How can I query that data? SPARQL

Basic graph pattern matching ~ **Conjunctive** queries

Example:

"Give me all names of co-authors 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 ] ] . }
```

- **Blank nodes in Queries play a *similar* role as (non-distinguished) variables.**
- **Turtle style shortcuts are allowed (*a bit extreme here, admittedly*)**

Link

More complex patterns in SPARQL 1.0

- UNION
- OPTIONAL
- FILTER
- Querying named GRAPHs
- Solution Modifiers (ordering, slicing/dicing results)
- … plus some non-trivial combinations of these

UNIONs of conjunctive queries…

Avoid Duplicates: keyword DISTINCT

Example:

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

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?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"
…

UNIONs of conjunctive queries…

Avoid Duplicates: keyword `DISTINCT`

Example:

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

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?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…

Avoid Duplicates: keyword DISTINCT

Example:

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

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

Note: again Duplicates possible!

?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"
"Charles McCathieNevile"
…

UNIONs of conjunctive queries…

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 ?CoAuthN ?FrN
WHERE {
        { [ foaf:maker <http://dblp.l3s.de/…/authors/Tim_Bern  s-Lee>,
                       [ foaf:name ?CoAuthN ] ] . }
        UNION
        { <http://www.w3.org/People/Berners-Lee/card#i> foaf:kn ows ?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

Optional parts in queries (Left Outer Join)

Example:

"Give me all names of co-authors of Tim Berners-L

*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 }
      }
```

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"	-

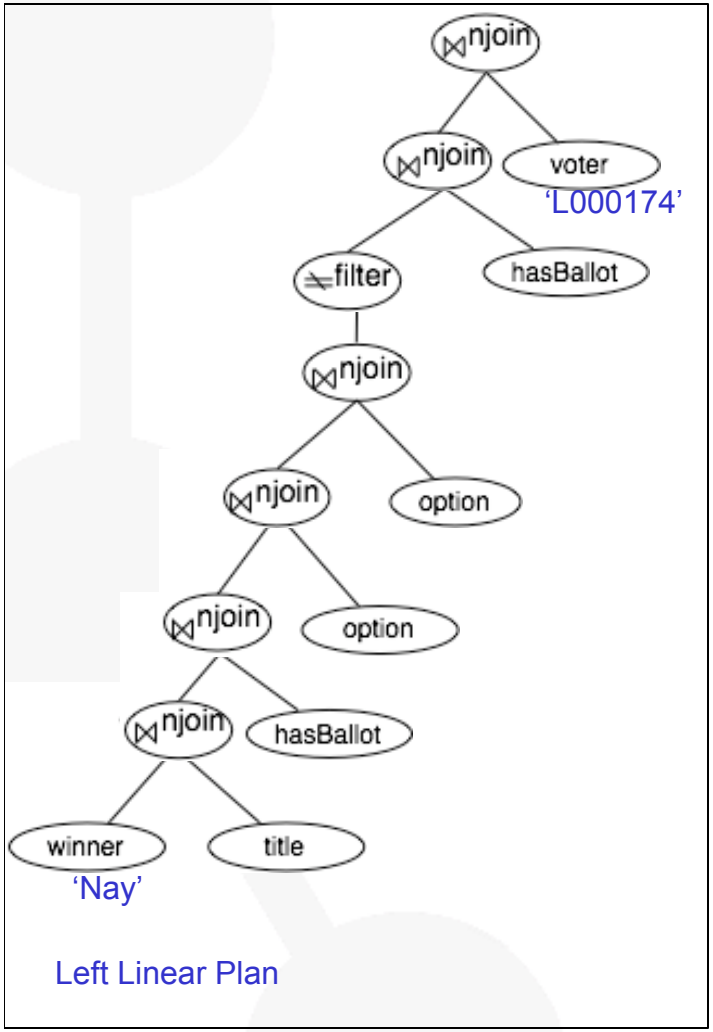
Some SPARQL Axioms

- . (Join) is commutative and associative
- OPTIONAL is associative
  - “Well-designed patterns”
- UNION is commutative and associative
- …..

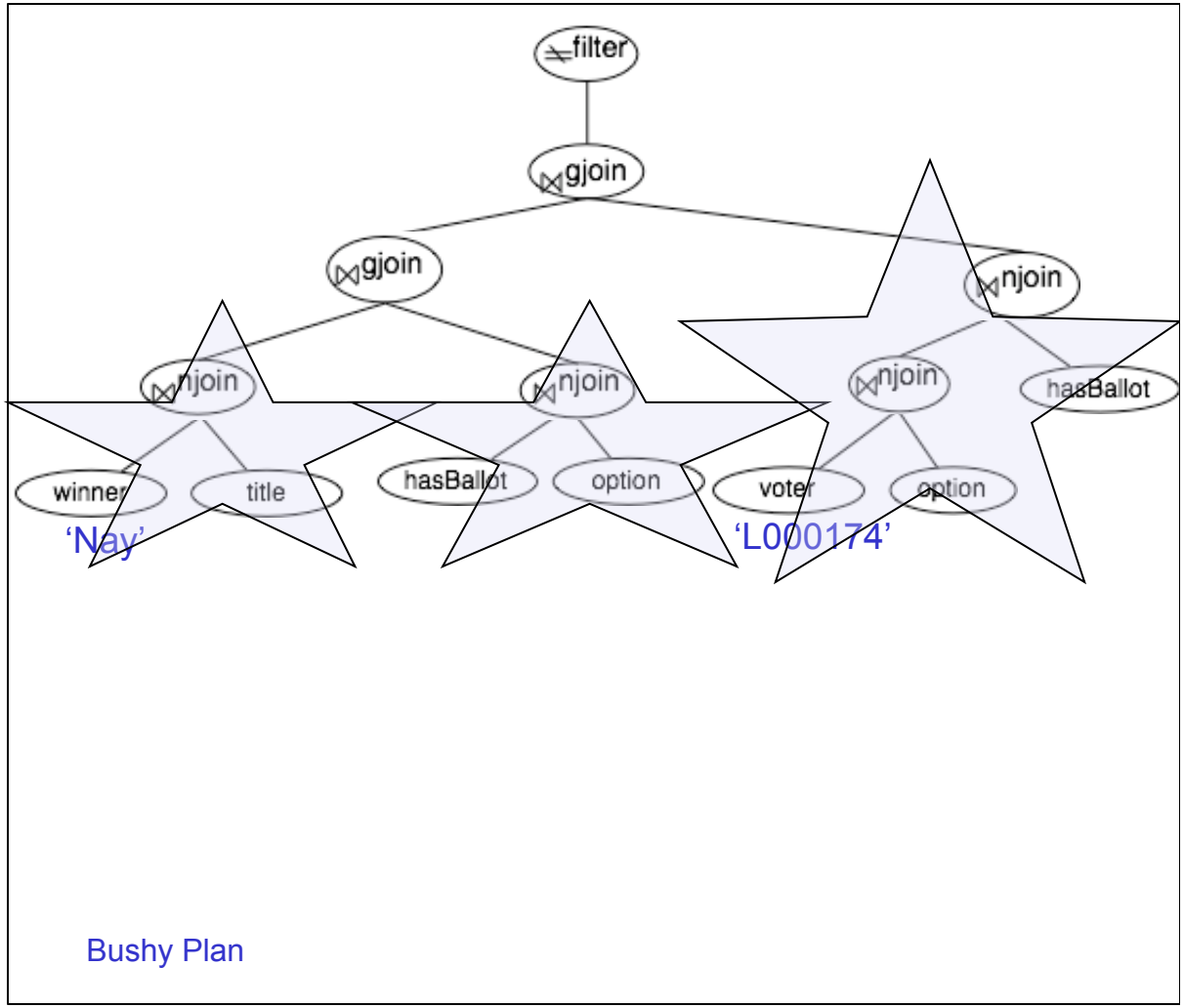
- {A.B} . {C.D} equivalent to {{A.B}.C}.D

Query engine may recognize or generate groupings in order to generate efficient execution plans.

SPARQL Query Execution Plans



Left Linear Plan



Bushy Plan

FILTERING out query results

FILTERs allow to specify FILTER conditions on patterns

Example:

different from Tim B.-L. himself"

"Give me all names of co-authors of Tim Berners-Lee where their homepage starts wit 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" ) &&
        ?CoAuth != <http://dblp.l3s.de/…/authors/Tim_Berners-Lee> )
       }
```

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/>

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 ) }
```

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

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

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

… FILTER in SPARQL is like WHERE in SQL, but SPARQL1.0 doesn't have AS

- Negation (“NOT EXISTS” in SQL)
  - *“Give me all Persons without a homepage”*
  - by combination of OPTIONAL and FILTER(!bound(…))

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

Constructing Graphs

Construct new graphs:

- *"everybody knows their co-authors"*

```
CONSTRUCT { ?X foaf:knows ?Y }
WHERE{ ?D foaf:maker ?X, ?Y .
       FILTER( ?X != ?Y ) }
```

Constructing Graphs

- Limitations
  - Again, no assignment, creation of values
    - How to concatenate first name and last name?
  - No aggregation (e.g. COUNT, SUM, …):
    - How to create a graph that has publication count per person for DBLP?
  - No RDFS/OWL inference (so combining mappings in RDFS/OWL with queries in SPARQL not possible)

SPARQL1.0 Formal Semantics

- *Graph patterns:*
  - BGPs
  - *P1 P2*
  - *P* `FILTER` *R*
  - *P1* `UNION` *P2*
  - *P1* `OPTIONAL` *P2*
- Semantics
  - *eval(D(G), graph pattern)* …

D is a dataset,

G is the "active graph"

recursively defined for all graph patterns in Section 12.5 of http://www.w3.org/TR/rdf-sparql-query/

Spec. semantics is a bit hard to read …

Explained in more "accessible" terms in extended version of this Tutorial: http://www.polleres.net/presentations/20101006SPARQL1.1Tutorial.pptx

Academic works around SPARQL

- 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]
- Proposed Extensions
  - Aggregates [Polleres et al. 2007]
  - Property Paths [Alkhateeb et al. 2009], [Perez et al. 2008]

SPARQL1.1

WG MIGHT STILL CHANGE SOME OF THE SYNTAX/ SEMANTICS DEFINITIONS PRESENTED HERE BASED ON COMMUNITY INPUT

This is where SPARQL1.1 starts

Missing common feature requirements in existing implementations or requested urgently by the community:

- **Assignment/Project Expressions**
- **Aggregate functions (SUM, AVG, MIN, MAX, COUNT, …)**
- **Subqueries**
- **Property paths**
  - complaint: SPARQL1.0 isn't quite a "graph" query language

Ease of use:

- Why is **Negation** "hidden" in SPARQL1.0?

Interplay with other SW standards:

- SPARQL1.0 only defined for simple RDF entailment
- Other Entailment regimes missing:
  - **RDF(S)**, OWL
  - **OWL2**
  - **RIF**

Goals of SPARQL1.1

Per charter (http://www.w3.org/2009/05/sparql-phase-II-charter.html)

- "The scope of this charter is to extend SPARQL technology to include some of the features that the community has identified as both desirable and important for interoperability **based on experience** with the initial version of the standard."

➔ No inclusion of new features that still require research

➔ Upwards compatible with SPARQL1.0

➔ The name SPARQL1.1 shall indicate an incremental change rather than any fundamental changes.

Goals of SPARQL1.1

- List of agreed features:
- **Additions to the Query Language:**
  - **Project Expressions**
  - **Aggregate functions**
  - **Subqueries**
  - **Negation**
  - **Property Paths** ***(time permitting)***
  - Extend the function library *(time permitting)*
  - Basic federated Queries *(time permitting)*
- **Entailment** *(time permitting)*
- SPARQL Update
  - Full Update language
  - plus simple RESTful update methods for RDF graphs (HTTP methods)
- Service Description
  - Method for discovering a SPARQL endpoint's capabilities
  - Summary of its data

We will focus on these in today's Tutorial

Part 1: new query features

- Project Expressions
- Aggregate functions
- Subqueries
- Negation
- Property Paths

Project Expressions

Assignments, Creating new values…

```
PREFIX ex: <http://example.org/>
SELECT ?Item ?NewP
WHERE { ?Item ex:price ?Pr FILTER (?NewP = ?Pr * 1.1) }
```

Data:

```
@prefix ex: <http://example.org/> .

   ex:lemonade1      ex:price 3 .
    ex:beer1          ex:price 3.
  ex:wine1          ex:price 3.50 .
```

Results:

Leaves errors unbound!

?Item	?NewP
lemonade1	3.3
beer1	3.3
wine1	3.85
liqueur1	

Project Expressions

Assignments, Creating new values…

```
PREFIX ex: <http://example.org/>
SELECT ?Item ?NewP
WHERE { ?Item ex:price ?Pr FILTER (?NewP = ?Pr * 1.1) }
```

Data:

```
@prefix ex: <http://example.org/> .

   ex:lemonade1      ex:price 3 .
    ex:beer1          ex:price 3.
  ex:wine1          ex:price 3.50 .
```

Results:

Leaves errors unbound!

?Item	?NewP
lemonade1	3.3
beer1	3.3
wine1	3.85
liqueur1	

Aggregates

Aggregates

"Count items"

```
PREFIX ex: <http://example.org/>
SELECT (Count(?Item) AS ?C)
WHERE { ?Item ex:price ?Pr }
```

Data:

```
@prefix ex: <http://example.org/> .

ex:lemonade1    ex:price 3 ;
                rdf:type ex:Softdrink.
ex:beer1        ex:price 3;
                rdf:type ex:Beer.
ex:wine1        ex:price 3.50 ;
                rdf:type ex:Wine.
ex:wine2        ex:price 4 .
                rdf:type ex:Wine.
ex:wine3        ex:price "n/a";
                rdf:type ex:Wine.
```

Results:

?C
5

Aggregates

"Count categories"

```
PREFIX ex: <http://example.org/>
SELECT (Count(DISTINCT ?T) AS ?C)
WHERE { ?Item rdf:type ?T }
```

Data:

```
@prefix ex: <http://example.org/> .

ex:lemonade1    ex:price 3 ;
                rdf:type ex:Softdrink.
ex:beer1        ex:price 3;
                rdf:type ex:Beer.
ex:wine1        ex:price 3.50 ;
                rdf:type ex:Wine.
ex:wine2        ex:price 4 .
                rdf:type ex:Wine.
ex:wine3        ex:price "n/a";
                rdf:type ex:Wine.
```

Results:

?C
3

Aggregates - Grouping

"Count items per categories"

```
PREFIX ex: <http://example.org/>
SELECT ?T (Count(?Item) AS ?C)
WHERE { ?Item rdf:type ?T }
GROUP BY ?T
```

Data:

```
@prefix ex: <http://example.org/> .

   ex:lemonade1    ex:price 3 ;
                rdf:type ex:Softdrink.
    ex:beer1        ex:price 3;
                  rdf:type ex:Beer.
  ex:wine1         ex:price 3.50 ;
                  rdf:type ex:Wine.
   ex:wine2         ex:price 4 .
                  rdf:type ex:Wine.
  ex:wine3         ex:price "n/a";
                  rdf:type ex:Wine.
```

Results:

?T	?C
Softdrink	1
Beer	1
Wine	3

Aggregates – Filtering Groups

"Count items per categories, for those categories having more than one item"

```
PREFIX ex: <http://example.org/>
SELECT ?T (Count(?Item) AS ?C)
WHERE { ?Item rdf:type ?T }
GROUP BY ?T
HAVING Count(?Item) > 1
```

Data:

```
@prefix ex: <http://example.org/> .

    ex:lemonade1    ex:price 3 ;
                rdf:type ex:Softdrink.
     ex:beer1         ex:price 3;
                   rdf:type ex:Beer.
   ex:wine1          ex:price 3.50 ;
                   rdf:type ex:Wine.
    ex:wine2         ex:price 4 .
                   rdf:type ex:Wine.
   ex:wine3          ex:price "n/a";
                   rdf:type ex:Wine.
```

Results:

?T	?C
Wine	3

Other Aggregates

- SUM … *as usual*
- AVG … *as usual*
- MIN … *as usual*
- MAX … *as usual*
- SAMPLE … *"pick" one non-deterministically*
- GROUP_CONCAT … *concatenate values with a designated separator string*

…this list is extensible … *new built-ins will need to define error-behaviour, extra-parameters (like SEPARATOR in GROUP_CONCAT)*

Example SUM

"Sum Prices per categories"

```
PREFIX ex: <http://example.org/>
SELECT ?T (Sum(IF(isNumeric(?Pr),?Pr,0)) AS ?P)
WHERE { ?Item rdf:type ?T; ex:price ?Pr }
GROUP BY ?T
```

Data:

```
@prefix ex: <http://example.org/> .

ex:lemonade1    ex:price 3 ;
                rdf:type ex:Softdrink.
ex:beer1        ex:price 3;
                rdf:type ex:Beer.
ex:wine1        ex:price 3.50 ;
                rdf:type ex:Wine.
ex:wine2        ex:price 4 .
                rdf:type ex:Wine.
ex:wine3        ex:price "n/a";
                rdf:type ex:Wine.
```

Results:

?T	?C
Softdrink	3
Beer	3
Wine	7.5

Example GROUP_CONCAT, SAMPLE

"pick one sample name per person, plus a concatenated list of nicknames "

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ( SAMPLE(?N) as ?Name)
       ( GROUP_CONCAT(?M; SEPARATOR = ", ") AS ?Nicknames )
WHERE { ?P a foaf:Person ;
           foaf:name ?N ;
           foaf:nick ?M . }
GROUP BY ?P
```

```
@prefix ex: <http://example.org/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

ex:alice a foaf:Person; foaf:name "Alice Wonderland";
          foaf:nick "Alice", "The real Alice".

ex:bob a foaf:Person;
       foaf:name "Robert Doe", "Robert Charles Doe",
                 "Robert C. Doe";
       foaf:nick "Bob","Bobby","RobC","BobDoe".

ex:charles a foaf:Person;
       foaf:name "Charles Charles";
       foaf:nick "Charlie" .
```

Name	Nicknames
Alice Wonderland	The real Alice, Alice
Charles Charles	Charlie
Robert C. Doe	Bob, BobDoe, RobC, Bobby

Subqueries

Subqueries to realise complex mappings

- How to concatenate first name and last name?
- Now possible without problems per subqueries!

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX fn: <http://www.w3.org/2005/xpath-functions#>

CONSTRUCT{ ?P foaf:name ?FullName }
WHERE {
 SELECT ?P ( fn:concat(?F, " ", ?L) AS ?FullName )
 WHERE { ?P foaf:firstName ?F ; foaf:lastName ?L. }
}
```

Subqueries to realise complex mappings

- Give me **all** titles of papers **of 10 persons** who co-authored with Tim Berners-Lee

```
SELECT ?T
WHERE {
 ?D foaf:maker ?P ; rdfs:label ?T .
{
SELECT DISTINCT ?P
WHERE { ?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee> ?P
    FILTER ( ?P != <http://dblp.l3s.de/…/authors/Tim_Berners-Lee> )
       }
 LIMIT 10
 }
}
```

- Returns titles for 10 **persons**, instead of just 10 **rows**

MINUS and NOT EXISTS

- ***Negation as failure*** *in SPARQL1.0 is "ugly":*

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

- ***SPARQL1.1*** *has two alternatives to do the same*
  - *NOT EXISTS in FILTERs*
    - *detect non-existence*
  - *(P1 MINUS P2) as a new binary operator*
    - *"Remove rows with matching bindings"*
    - *only effective when P1 and P2 share variables*

Property Path Expressions

Property Path expressions

- Concatenate property paths, Arbitrary Length paths, etc.
- E.g. names of people Tim Berners-Lee transitively co-authored papers with…

```
SELECT DISTINCT ?N
 WHERE {<http://dblp…/Tim_Berners-Lee>,
        (^foaf:maker/foaf:maker)+/foaf:name ?N
       }
```

Path expressions full list of operators

- elt … Path Element

Syntax Form	Matches
uri	A URI or a prefixed name. A path of length one.
^elt	Inverse path (object to subject).
!uri or *!(uri$_1$| ...|uri$_n$)*	Negated property set. A URI which is not one of *uri$_i$*
!^uri and *!(uri$_1$| ...|uri$_j$|^uri$_{j+1}$| ...|^uri$_n$)*	Negated property set. A URI which is not one of *uri$_i$*, nor uri$_{j+1}$...^uri$_n$ as reverse paths
(elt)	A group path *elt*, brackets control precedence.
elt1 / elt2	A sequence path of *elt1*, followed by *elt2*
elt1 | elt2	A alternative path of *elt1*, or *elt2* (all possibilities are tried).
*elt**	A path of zero or more occurrences of *elt*.
elt+	A path of one or more occurrences of *elt*.
elt?	A path of zero or one *elt*.
elt{n,m}	A path between n and m occurrences of *elt*.
elt{n}	Exactly *n* occurrences of *elt*.
elt{n,}	*n* or more occurrences of *elt*.
elt{,n}	Between 0 and *n* occurrences of *elt*.

- Semantics: by translation to native SPARQL with two core property paths Operators:
  - ArbitraryPath(X, path, Y)
  - ZeroLengthPath(X, path, Y)

Path expressions

- Can be used for some ontological inference (well known since [Perez et al. 2008]
- E.g. Find all Beers in the Beer ontology

```
PREFIX beer: <http://www.purl.org/net/ontology/beer#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?beer
FROM <http://www.purl.org/net/ontology/beer>
WHERE {
   ?beer rdf:type/rdfs:subClassOf* beer:Beer .
}
```

Implementations of SPARQL 1.1 Query:

Some current (partial) SPARQL1.1 implementations:

- ARQ
  - http://sourceforge.net/projects/jena/
  - http://sparql.org/sparql.html
- OpenAnzo
  - http://www.openanzo.org/
- Perl RDF
  - http://github.com/kasei/perlrdf/
- Corese
  - http://www-sop.inria.fr/teams/edelweiss/wiki/wakka.php?wiki=CoreseDownloads
- etc.

Others probably forthcoming…

- Loads of SPARQL1.0 endpoints around
  - Dbpedia: http://dbpedia.org/snorql/
  - DBLP: http://dblp.l3s.de/d2r/snorql/
  - Etc.

Part 2: Entailment Regimes

SPARQL 1.1 querying over RDFS+OWL2 ontologies and RIF rulesets?

SPARQL1.1 Entailment Regimes

- SPARQL1.1 will define SPARQL query answering over OWL2 ontologies and RIF rule sets:
  - http://www.w3.org/TR/sparql11-entailment/

  - RDF Entailment Regime
  - RDFS Entailment Regime
  - D-Entailment Regime
  - OWL 2 RDF-Based Semantics Entailment Regime
  - OWL 2 Direct Semantics Entailment Regime
  - RIF Core Entailment

    - Won't go into details of those, but sketch the main ideas!

RDFS/OWL2 and SPARQL1.1

- General Idea: Answer Queries with implicit answers
- E.g. example from before:

```
PREFIX beer: <http://www.purl.org/net/ontology/beer#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?beer
FROM <http://www.purl.org/net/ontology/beer>
WHERE {
    ?beer rdf:type beer:Beer .
}
```

```
beer:Boddingtons rdf:type beer:Ale .
beer:Grafentrunk  rdf:type  beer:Bock .
beer:Hoegaarden  rdf:type  beer:White .
beer:Jever rdf:type  beer:Pilsner .
beer:Krieger rdf:type  beer:Lager .
beer:Paulaner rdf:type  beer:White .
beer:Tetleys rdf:type  beer:Ale .
```

```
beer:Ale
beer:Boc
beer:Lag
beer:Pil
beer:Whi
beer:TopFermentedBeer rdfs:subClassOf beer:Beer.
beer:BottomFermentedBeer rdfs:subClassOf beer:Beer.
```

beer
<http://www.purl.org/net/ontology/beer#Hoegaarden>
<http://www.purl.org/net/ontology/beer#Boddingtons>
<http://www.purl.org/net/ontology/beer#Grafentrunk>
<http://www.purl.org/net/ontology/beer#Tetleys>
<http://www.purl.org/net/ontology/beer#Jever>
<http://www.purl.org/net/ontology/beer#Krieger>
<http://www.purl.org/net/ontology/beer#Paulaner>

Essential idea behind RDFS inference:

- SPARQL executes “inference” rules on the data, when answering queries, e.g.:

```
rdfs1: { ?S rdf:type ?C } :- { ?S ?P ?O . ?P rdfs:domain ?C . }
rdfs2: { ?O rdf:type ?C } :- { ?S ?P ?O . ?P rdfs:range ?C . }

rdfs3: { ?S rdf:type ?C2 } :- {?S rdf:type ?C1 . ?C1 rdfs:subclassOf ?
   C2 . }
```

```
beer:Boddingtons rdf:type beer:Ale ;
     rdf:type beer:TopFermentedBeer;
     rdf:type beer:Beer.
beer:Grafentrunk  rdf:type  beer:Bock .
     rdf:type beer:BottomFermentedBeer;
     rdf:type beer:Beer.
beer:Hoegaarden  rdf:type  beer:White ;
     rdf:type beer:TopFermentedBeer;
     rdf:type beer:Beer.

…
```

```
beer:Ale   rdfs:subClassOf beer:TopFermentedBeer .

beer:Bock  rdfs:subClassOf beer:BottomFermentedBeer .

beer:Lager rdfs:subClassOf beer:BottomFermentedBeer .

beer:Pilsner rdfs:subClassOf beer:BottomFermentedBeer .

beer:White rdfs:subClassOf beer:TopFermentedBeer .

beer:TopFermentedBeer rdfs:subClassOf beer:Beer.

beer:BottomFermentedBeer rdfs:subClassOf beer:Beer.
```

OWL2 and SPARQL1.1

- General Idea: Answer Queries with implicit answers
- E.g. Graph/Ontology:

```
foaf:Person rdfs:subClassOf foaf:Agent .
foaf:Person rdfs:subclassOf
        [ a owl:Restriction ;
   owl:onProperty :hasFather ;
   owl:someValuesFrom foaf:Person ] .
foaf:knows rdfs:range foaf:Person.

  :jeff a Person .
  :jeff foaf:knows :aidan
```

```
SELECT ?X { ?X a foaf:Person }
```

- Pure SPARQL 1.0 returns only :Jeff,

should also return :aidan

Wrapping up

- SPARQL 1.0
  - UNIONs of Conjunctive Queries, FILTERs, GRAPH queries, OPTIONAL, (hidden) negation
  - contributed largely to the current Linked Data boom
  - Inspired interesting academic work
- SPARQL 1.1
  - A reasonable next step
    - Incorporating highly demanded features
    - Closing the gaps to neighbour standards (OWL2, RIF)
  - Not all of it is trivial → SPARQL1.1 takes a very pragmatic path

- *Hopefully inspiring for more research, more data, and more applications!*

What I didn't talk about…

List of agreed features:

- **Additions to the Query Language:**
  - Project Expressions
  - Aggregate functions
  - Subqueries
  - Negation
  - Property Paths *(time permitting)*
  - **Extend the function library *(time permitting)***
  - **Basic federated Queries *(time permitting)***
- Entailment *(time permitting)*
- **SPARQL Update**
  - Full Update language
  - plus simple RESTful update methods for RDF graphs (HTTP methods)
- **Service Description**
  - Method for discovering a SPARQL endpoint's capabilities
  - Summary of its data

Basic federated Queries (*time permitting*)

- http://www.w3.org/TR/sparql11-federated-query/
  - Will be integrated in Query spec
- Essentially new pattern `SERVICE`
  - Similar to `GRAPH`
  - allows delegate query parts to a specific (remote) endpoint

Recall: *We were cheating in this query before!!*

```
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 }
  UNION
{ [ foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>,
            [ foaf:name ?N ] ] . }
}
```

Tim's FOAF file

DBLP SPARQL endpoint

Introduction

- How many of you have been in the need of making queries to distributed SPARQL endpoints?

Example

- Using the Pubmed references obtained from the Geneid gene dataset, retrieve information about genes and their references in the Pubmed dataset.
- From Pubmed we access the information in the National Library of Medicine's controlled vocabulary thesaurus, stored at the MeSH endpoint, so we have more complete information about such genes.
- Finally, we also access the HHPID endpoint, which is the knowledge base for the HIV-1 protein.

Basic federated Queries (*time permitting*)

- http://www.w3.org/TR/sparql11-federated-query/
  - Will be integrated in Query spec
- Essentially new pattern `SERVICE`
  - Similar to `GRAPH`
  - allows delegate query parts to a specific (remote) endpoint

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N
FROM <http://www.w3.org/People/Berners-Lee/card>
WHERE {
        { <http://www.w3.org/People/Berners-Lee/card#i> foaf:knows ?F .
          ?F foaf:name ?N    }
          UNION
        { SERVICE <http://dblp.l3s.de/d2r/sparql>
    { [ foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>,
                   [ foaf:name ?N ] ] . } }
     }
```

SPARQL1.1 Update

- Like SQL … SPARQL/RDF Stores need a standard Data Manipulation Language http://www.w3.org/TR/sparql11-update
- SPARQL 1.1 Update Language
  - Graph Update
    - INSERT DATA
    - DELETE DATA
    - DELETE/INSERT
    - DELETE
    - INSERT
    - DELETE WHERE
    - LOAD
    - CLEAR
  - Graph Management
    - CREATE
    - DROP
- *Issue: Graph-aware stores vs. Quad Stores*

Service Description

Base vocabulary to describe

- ***features of SPARQL endpoints***
- ***datasets*** *(via vocabularies external to the Spec,e.g. VOID)*

- http://www.w3.org/TR/sparql11-service-description/

3.2 Classes
- 3.2.1 sd:Service
- 3.2.2 sd:Language
- 3.2.3 sd:Function
- 3.2.4 sd:Aggregate
- 3.2.5 sd:EntailmentRegime
- 3.2.6 sd:EntailmentProfile
- 3.2.7 sd:GraphCollection
- 3.2.8 sd:Dataset
- 3.2.9 sd:Graph
- 3.2.10 sd:NamedGraph

3.3 Instances
- 3.3.1 sd:SPARQL10Query
- 3.3.2 sd:SPARQL11Query
- 3.3.3 sd:SPARQL11Update
- 3.3.4 sd:DereferencesURIs
- 3.3.5 sd:UnionDefaultGraph
- 3.3.6 sd:RequiresDataset
- 3.3.7 sd:EmptyGraphs

3.4 Properties
- 3.4.1 sd:url
- 3.4.2 sd:feature
- 3.4.3 sd:defaultEntailmentRegime
- 3.4.4 sd:supportedEntailmentProfile
- 3.4.5 sd:entailmentRegime
- 3.4.6 sd:extensionFunction
- 3.4.7 sd:extensionAggregate
- 3.4.8 sd:languageExtension
- 3.4.9 sd:supportedLanguage
- 3.4.10 sd:propertyFeature
- 3.4.11 sd:defaultDatasetDescription
- 3.4.12 sd:availableGraphDescriptions
- 3.4.13 sd:resultFormat
- 3.4.14 sd:defaultGraph
- 3.4.15 sd:namedGraph
- 3.4.16 sd:name
- 3.4.17 sd:graph

Relevant W3C Specs

- SPARQL Query Language for RDF http://www.w3.org/TR/rdf-sparql-query/
- SPARQL1.1 Query Language for RDF (working draft) http://www.w3.org/TR/sparql11-query/
- SPARQL1.1 Entailment Regimes (working draft) http://www.w3.org/TR/sparql11-entailment/

RDF(S) Entailment/D-Entailment:

- RDF Semantics http://www.w3.org/TR/rdf-mt/

OWL Entailment:

- OWL2 Web Ontology Language Primer http://www.w3.org/TR/owl2-primer/
- OWL2 Web Ontology Language Profiles http://www.w3.org/TR/owl2-profiles/

RIF Entailment:

- RIF Core Dialect http://www.w3.org/TR/rif-core/
- RIF Basic Logic Dialect http://www.w3.org/TR/rif-bld/
- RIF RDF and OWL compatibility http://www.w3.org/TR/rif-rdf-owl/

References: Academic Results on SPARQL

[Alkhateeb et al. 2009] Faisal Alkhateeb, Jean-Francois Baget, and Jerome Euzenat. Extending SPARQL with regular expression patterns (for querying RDF). JWS, 7(2), 2009.

[Angles & Gutierrez, 2008] Renzo Angles and Claudio Gutierrez. The expressive power of SPARQL, ISWC 2008.

[Eiter et al. 2006] Thomas Eiter, Giovambattista Ianni, Roman Schindlauer and Hans Tompits. Effective Integration of Declarative Rules with External Evaluations for Semantic-Web Reasoning, ESWC 2006.

[Perez et al. 2006] Jorge Perez, Marcelo Arenas, Claudio Gutierrez. Semantics and complexity of SPARQL. ISWC 2006.

[Perez et al. 2009] Jorge Perez, Marcelo Arenas, Claudio Gutierrez. Semantics and complexity of SPARQL. ACM ToDS 34(3), 2009.

[Perez et al. 2008] Jorge Perez, Marcelo Arenas, and Claudio Gutierrez. nSPARQL: A navigational language for RDF. In 7th International Semantic Web Conference, ISWC 2008.

[Polleres 2007] Axel Polleres From SPARQL to Rules (and back). WWW 2007

[Polleres et al. 2007] Axel Polleres,Francois Scharffe,and Roman Schindlauer. SPARQL++ for mapping between RDF vocabularies. ODBASE 2007.

[Schmidt et al. 2010] Michael Schmidt, Michael Meier, and Georg Lausen. Foundations of sparql query optimization. ICDT2010