SPARQL1.1: An introduction

@AxelPolleres

Digital Enterprise Research Institute, National University of Ireland, Galway

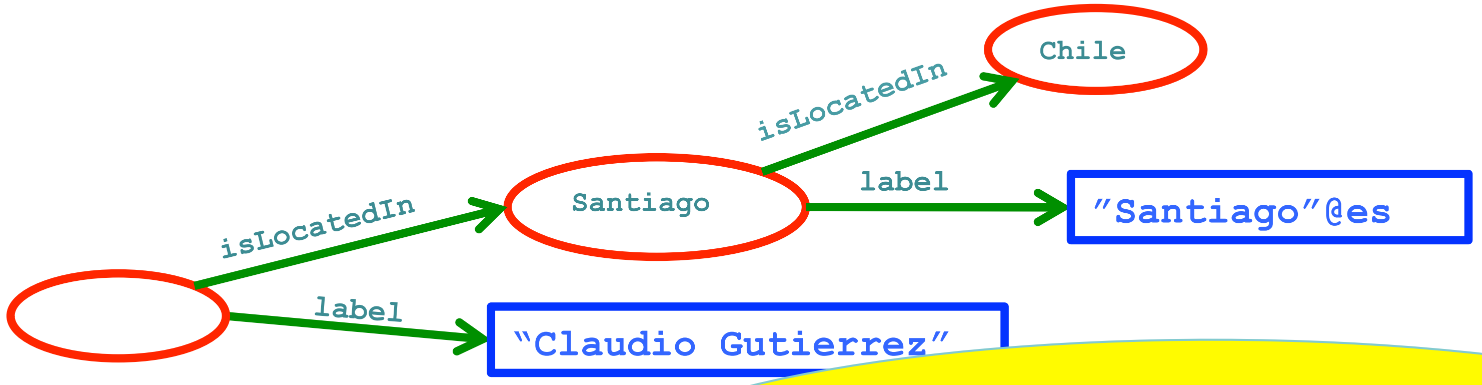
What is SPARQL?

- 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
  - SPARQL 1.1 Uniform HTTP Protocol for Managing RDF Graphs
  - SPARQL 1.1 Service Descriptions

What you'll hear

- Run through SPARQL1.0
- New features in SPARQL 1.1 Query
- SPARQL 1.1 Entailment Regimes
- Implementations, Status

RDF a plain data format for the Web



Various syntaxes, RDF/XML, Turtle, N3, RDFa,…

U x B

URIs, e.g.

```
http://www.w3.org/2000/01/rdf-schema#label
http://ontology.dumontierlab.com/isLocatedIn
http://dbpedia.org/resource/Santiago
http://dbpedia.org/resource/Chile
```

```
<http://dbpedia.org/resource/Santiago> <htt          ogy.dumontie
                           <http://dbped          esource/Chile>
<http://dbpedia.org/resource/Santiago>          ://www.w3.org/2000/0
                           "Santiago          .

_:x <http://www.w3.org/2000/01/ df-schema#label> "Claudio Gutierrez" .
_:x <http://ontology.dumontierlab.com/isLoca      In> <http://dbpedia.org/
```

Blanknodes:

"existential variables in the data" to express incomplete information, written as _:x or []

Literals, e.g.

```
"2010"^^xsd:gYear
"Brixen"@de
"Bressanone"@it
"Santiago"@es
"Claudio Gutierres
```

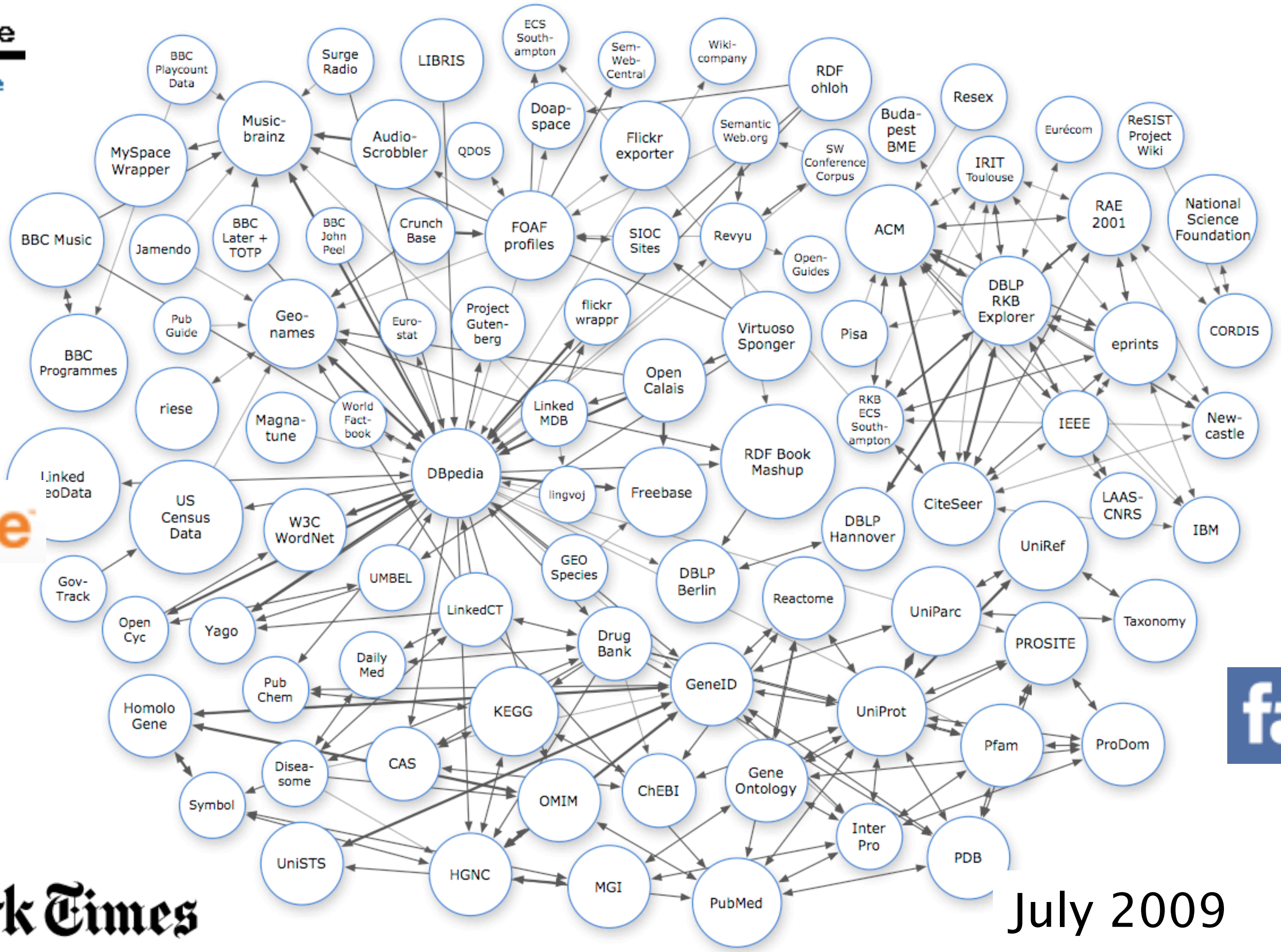
RDF Data on the Web: Linked Open Data

dblp.uni-trier.de
Computer Science Bibliography

Freebase

...

The New York Times



WIKIPEDIA

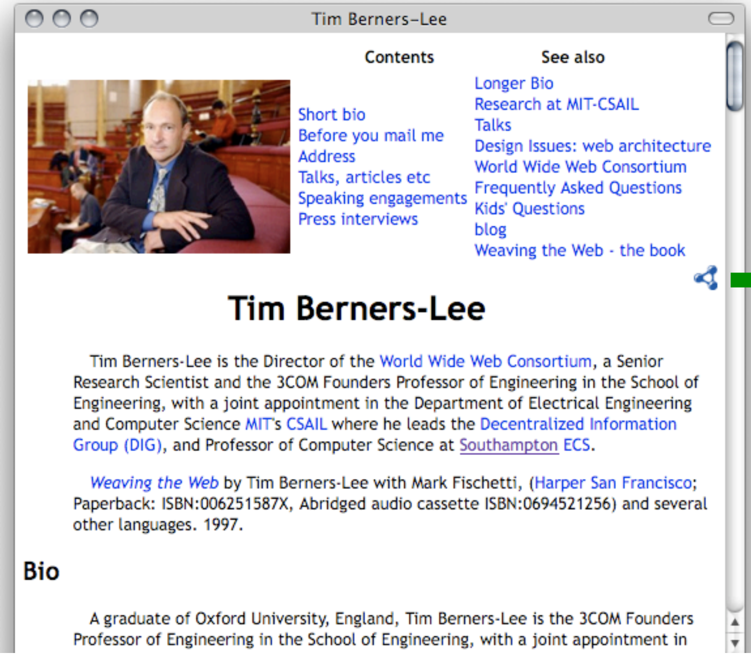
facebook

July 2009

RDF Data online: Example 1/4

- **(i) directly by the publishers**
- (ii) by exporters

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



Tim Berners-Lee

Contents

Short bio
Before you mail me
Address
Talks, articles etc
Speaking engagements
Press interviews

See also

Longer Bio
Research at MIT-CSAIL
Talks
Design Issues: web architecture
World Wide Web Consortium
Frequently Asked Questions
Kids' Questions
blog
Weaving the Web - the book

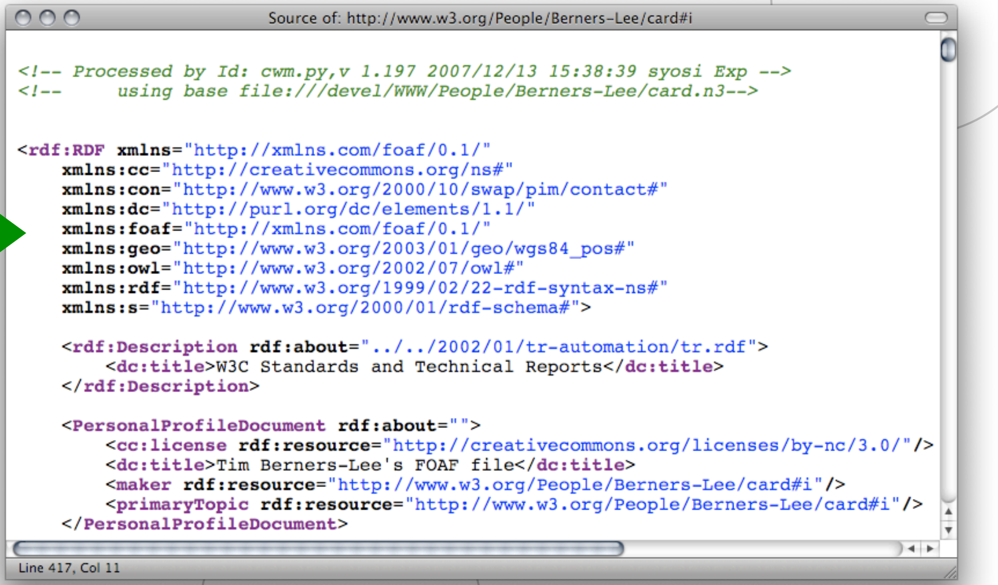
Tim Berners-Lee

Tim Berners-Lee is the Director of the World Wide Web Consortium, a Senior Research Scientist and the 3COM Founders Professor of Engineering in the School of Engineering, with a joint appointment in the Department of Electrical Engineering and Computer Science MIT's CSAIL where he leads the Decentralized Information Group (DIG), and Professor of Computer Science at Southampton ECS.

Weaving the Web by Tim Berners-Lee with Mark Fischetti, (Harper San Francisco; Paperback: ISBN:006251587X, Abridged audio cassette ISBN:0694521256) and several other languages. 1997.

Bio

A graduate of Oxford University, England, Tim Berners-Lee is the 3COM Founders Professor of Engineering in the School of Engineering, with a joint appointment in



Source of: http://www.w3.org/People/Berners-Lee/card#i

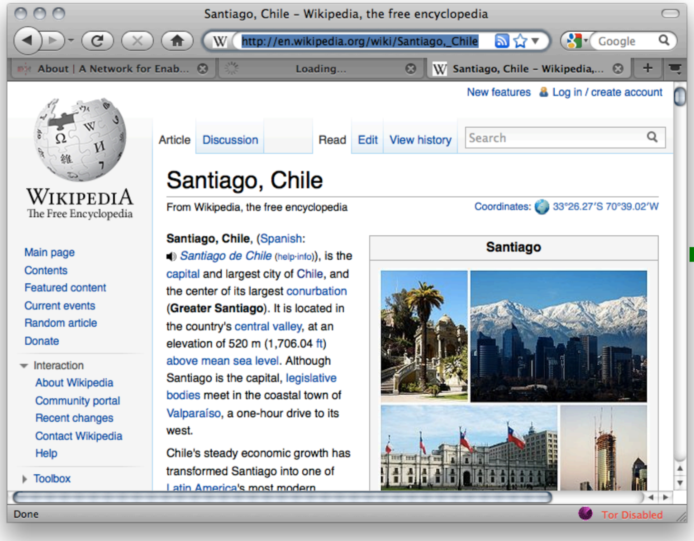
```
<!-- Processed by Id: cwm.py,v 1.197 2007/12/13 15:38:39 syosi Exp -->
<!--     using base file:///devel/WWW/People/Berners-Lee/card.n3-->


<rdf:RDF xmlns="http://xmlns.com/foaf/0.1/"
    xmlns:cc="http://creativecommons.org/ns#"
    xmlns:con="http://www.w3.org/2000/10/swap/pim/contact#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:s="http://www.w3.org/2000/01/rdf-schema#">

    <rdf:Description rdf:about="../../2002/01/tr-automation/tr.rdf">
        <dc:title>W3C Standards and Technical Reports</dc:title>
    </rdf:Description>

    <PersonalProfileDocument rdf:about="">
        <cc:license rdf:resource="http://creativecommons.org/licenses/by-nc/3.0/"/>
        <dc:title>Tim Berners-Lee's FOAF file</dc:title>
        <maker rdf:resource="http://www.w3.org/People/Berners-Lee/card#i"/>
        <primaryTopic rdf:resource="http://www.w3.org/People/Berners-Lee/card#i"/>
    </PersonalProfileDocument>
```

Line 417, Col 11

RDF Data online: Example 2/4

- (i) directly by the publishers
- **(ii) by exporters, e.g. OpenLink's Virtuoso.**

e.g. DBPedia, an export of Wikipedia's structured Data, using OpenLink's Virtuoso (http://dbpedia.org)



Santiago, Chile – Wikipedia, the free encyclopedia

http://en.wikipedia.org/wiki/Santiago,_Chile

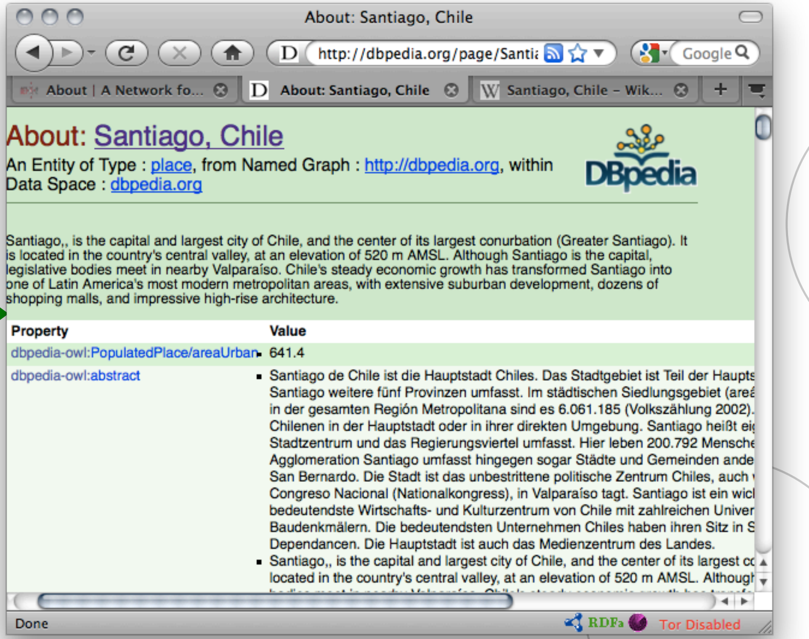
Santiago, Chile

From Wikipedia, the free encyclopedia

Coordinates: 33°26.27′S 70°39.02′W

Santiago, Chile, (Spanish: *Santiago de Chile* (help·info)), is the capital and largest city of Chile, and the center of its largest conurbation (**Greater Santiago**). It is located in the country's central valley, at an elevation of 520 m (1,706.04 ft) above mean sea level. Although Santiago is the capital, legislative bodies meet in the coastal town of Valparaíso, a one-hour drive to its west.

Chile's steady economic growth has transformed Santiago into one of Latin America's most modern



About: Santiago, Chile

http://dbpedia.org/page/Santi

About: Santiago, Chile

An Entity of Type : place, from Named Graph : http://dbpedia.org, within Data Space : dbpedia.org

Santiago,, is the capital and largest city of Chile, and the center of its largest conurbation (Greater Santiago). It is located in the country's central valley, at an elevation of 520 m AMSL. Although Santiago is the capital, legislative bodies meet in nearby Valparaíso. Chile's steady economic growth has transformed Santiago into one of Latin America's most modern metropolitan areas, with extensive suburban development, dozens of shopping malls, and impressive high-rise architecture.

Property	Value
dbpedia-owl:PopulatedPlace/areaUrban	■ 641.4
dbpedia-owl:abstract	■ Santiago de Chile ist die Hauptstadt Chiles. Das Stadtgebiet ist Teil der Haupts Santiago weitere fünf Provinzen umfasst. Im städtischen Siedlungsgebiet (area in der gesamten Región Metropolitana sind es 6.061.185 (Volkszählung 2002). Chilenen in der Hauptstadt oder in ihrer direkten Umgebung. Santiago heißt ei Stadtzentrum und das Regierungsviertel umfasst. Hier leben 200.792 Mensche Agglomeration Santiago umfasst hingegen sogar Städte und Gemeinden ande San Bernardo. Die Stadt ist das unbestrittene politische Zentrum Chiles, auch Congreso Nacional (Nationalkongress), in Valparaíso tagt. Santiago ist ein wic bedeutendste Wirtschafts- und Kulturzentrum von Chile mit zahlreichen Univer Baudenkmälern. Die bedeutendsten Unternehmen Chiles haben ihren Sitz in S Dependancen. Die Hauptstadt ist auch das Medienzentrum des Landes. ■ Santiago,, is the capital and largest city of Chile, and the center of its largest c located in the country's central valley, at an elevation of 520 m AMSL. Althoug

Gives unique URIs to cities, countries, persons, etc. from wikipedia! E.g.,

http://dbpedia.org/resource/Santiago%2C_Chile

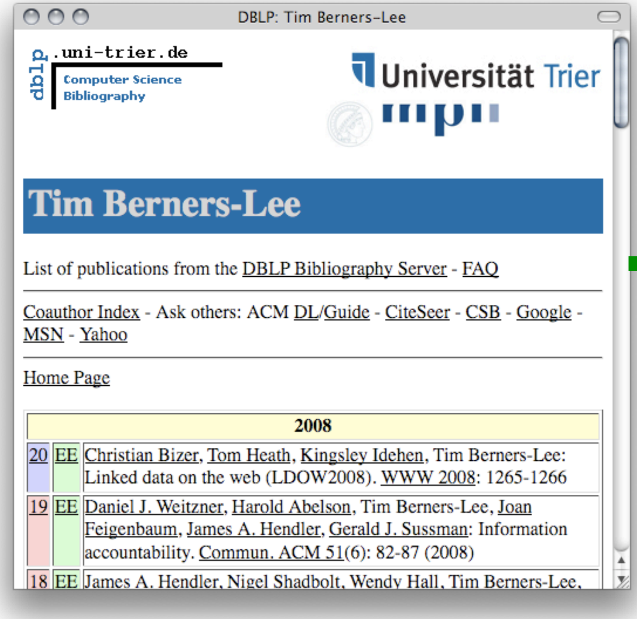
http://dbpedia.org/resource/Chile

Provides RDF version of all wikipedia structured data (infoboxes) and even a SPARQL query interface!

RDF Data online: Example 3/4

- (i) directly by the publishers
- **(ii) by exporters, e.g. D2R.**

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



DBLP: Tim Berners-Lee

dblp.uni-trier.de Computer Science Bibliography

Universität Trier

mpii

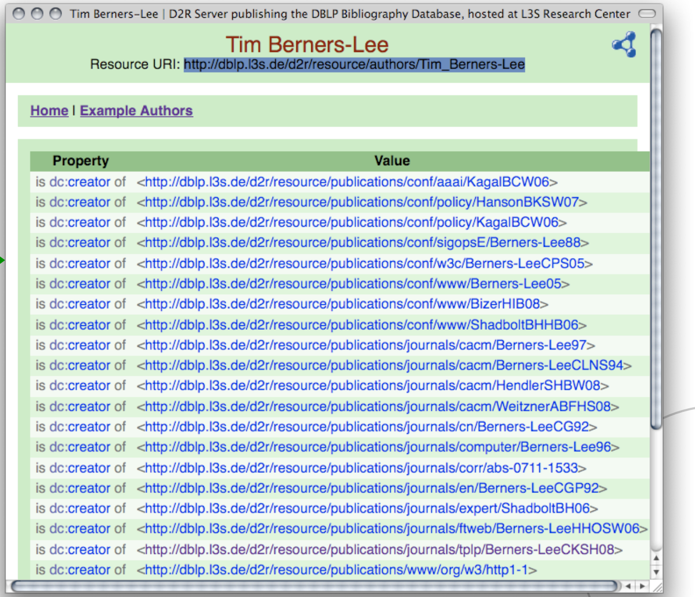
Tim Berners-Lee

List of publications from the DBLP Bibliography Server - FAQ

Coauthor Index - Ask others: ACM DL/Guide - CiteSeer - CSB - Google - MSN - Yahoo

Home Page

2008		
20	EE	Christian Bizer, Tom Heath, Kingsley Idehen, Tim Berners-Lee: Linked data on the web (LDOW2008). WWW 2008: 1265-1266
19	EE	Daniel J. Weitzner, Harold Abelson, Tim Berners-Lee, Joan Feigenbaum, James A. Hendler, Gerald J. Sussman: Information accountability. Commun. ACM 51(6): 82-87 (2008)
18	EE	James A. Hendler, Nigel Shadbolt, Wendy Hall, Tim Berners-Lee,



Tim Berners-Lee | D2R Server publishing the DBLP Bibliography Database, hosted at L3S Research Center

Tim Berners-Lee

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

Home | Example Authors

Property	Value
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/conf/aaai/KagalBCW06>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/conf/policy/HansonBKSW07>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/conf/policy/KagalBCW06>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/conf/sigopsE/Berners-Lee88>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/conf/w3c/Berners-LeeCPS05>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/conf/www/Berners-Lee05>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/conf/www/BizerHIB08>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/conf/www/ShadboltBHHB06>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/journals/cacm/Berners-Lee97>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/journals/cacm/Berners-LeeCLNS94>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/journals/cacm/HendlerSHBW08>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/journals/cacm/WeitznerABFHS08>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/journals/cn/Berners-LeeCG92>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/journals/computer/Berners-Lee96>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/journals/corr/abs-0711-1533>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/journals/en/Berners-LeeCGP92>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/journals/expert/ShadboltBH06>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/journals/ftweb/Berners-LeeHHOSW06>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/journals/tplp/Berners-LeeCKSH08>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/www/org/w3/http1-1>

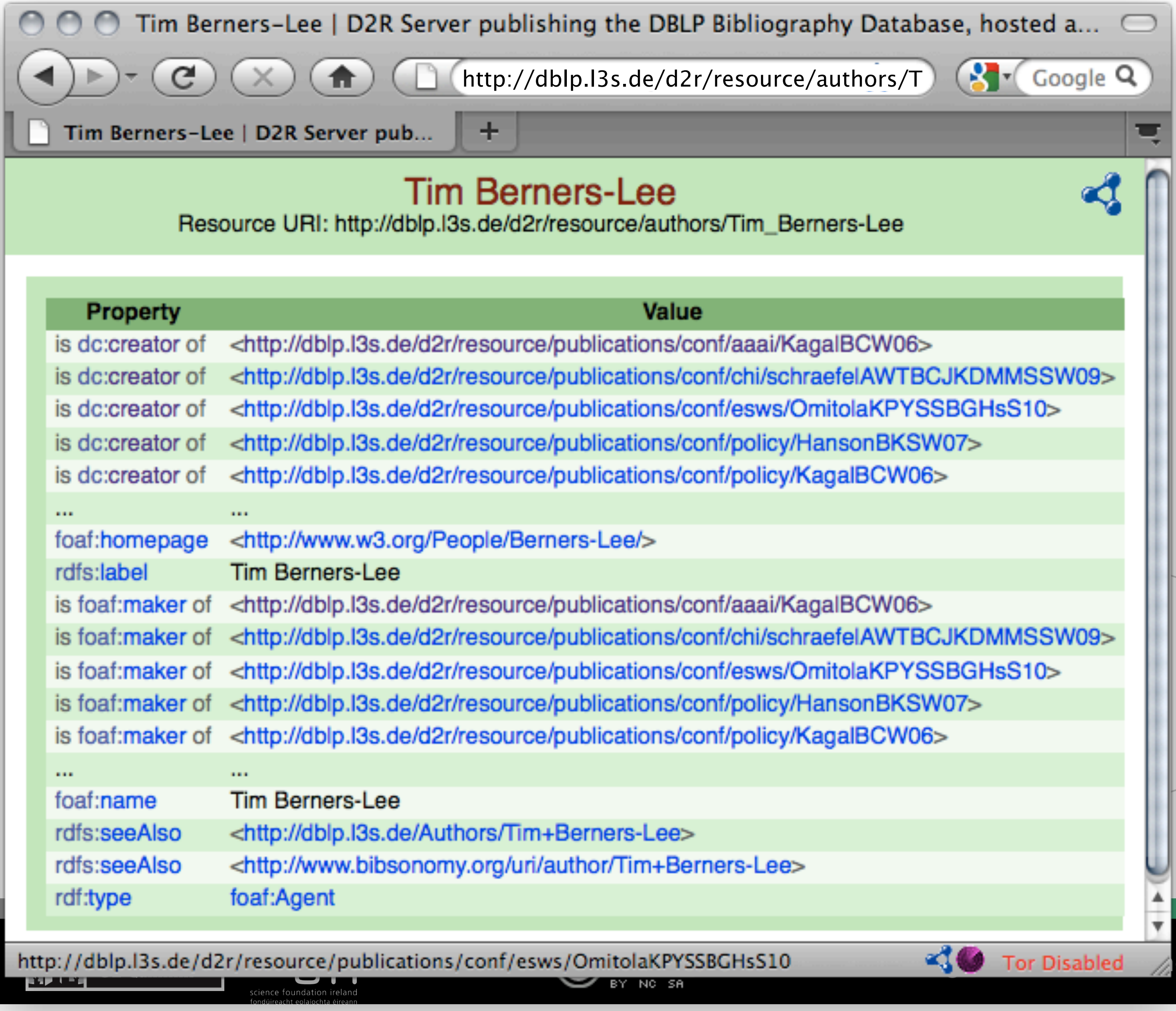
Gives unique URIs to authors, documents, etc. on DBLP! E.g.,

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 online: Example 4/4



Tim Berners-Lee | D2R Server publishing the DBLP Bibliography Database, hosted a...
http://dblp.l3s.de/d2r/resource/authors/T
Google
Tim Berners-Lee | D2R Server pub...
+

Tim Berners-Lee

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

Property	Value
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/conf/aaai/KagalBCW06>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/conf/chi/schraefelAWTBCJKDMMSSW09>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/conf/esws/OmitolaKPYSSBGHsS10>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/conf/policy/HansonBKSW07>
is dc:creator of	<http://dblp.l3s.de/d2r/resource/publications/conf/policy/KagalBCW06>
...	...
foaf:homepage	<http://www.w3.org/People/Berners-Lee/>
rdfs:label	Tim Berners-Lee
is foaf:maker of	<http://dblp.l3s.de/d2r/resource/publications/conf/aaai/KagalBCW06>
is foaf:maker of	<http://dblp.l3s.de/d2r/resource/publications/conf/chi/schraefelAWTBCJKDMMSSW09>
is foaf:maker of	<http://dblp.l3s.de/d2r/resource/publications/conf/esws/OmitolaKPYSSBGHsS10>
is foaf:maker of	<http://dblp.l3s.de/d2r/resource/publications/conf/policy/HansonBKSW07>
is foaf:maker of	<http://dblp.l3s.de/d2r/resource/publications/conf/policy/KagalBCW06>
...	...
foaf:name	Tim Berners-Lee
rdfs:seeAlso	<http://dblp.l3s.de/Authors/Tim+Berners-Lee>
rdfs:seeAlso	<http://www.bibsonomy.org/uri/author/Tim+Berners-Lee>
rdf:type	foaf:Agent

http://dblp.l3s.de/d2r/resource/publications/conf/esws/OmitolaKPYSSBGHsS10
Tor Disabled

RDF Data online: Example – 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" .
```

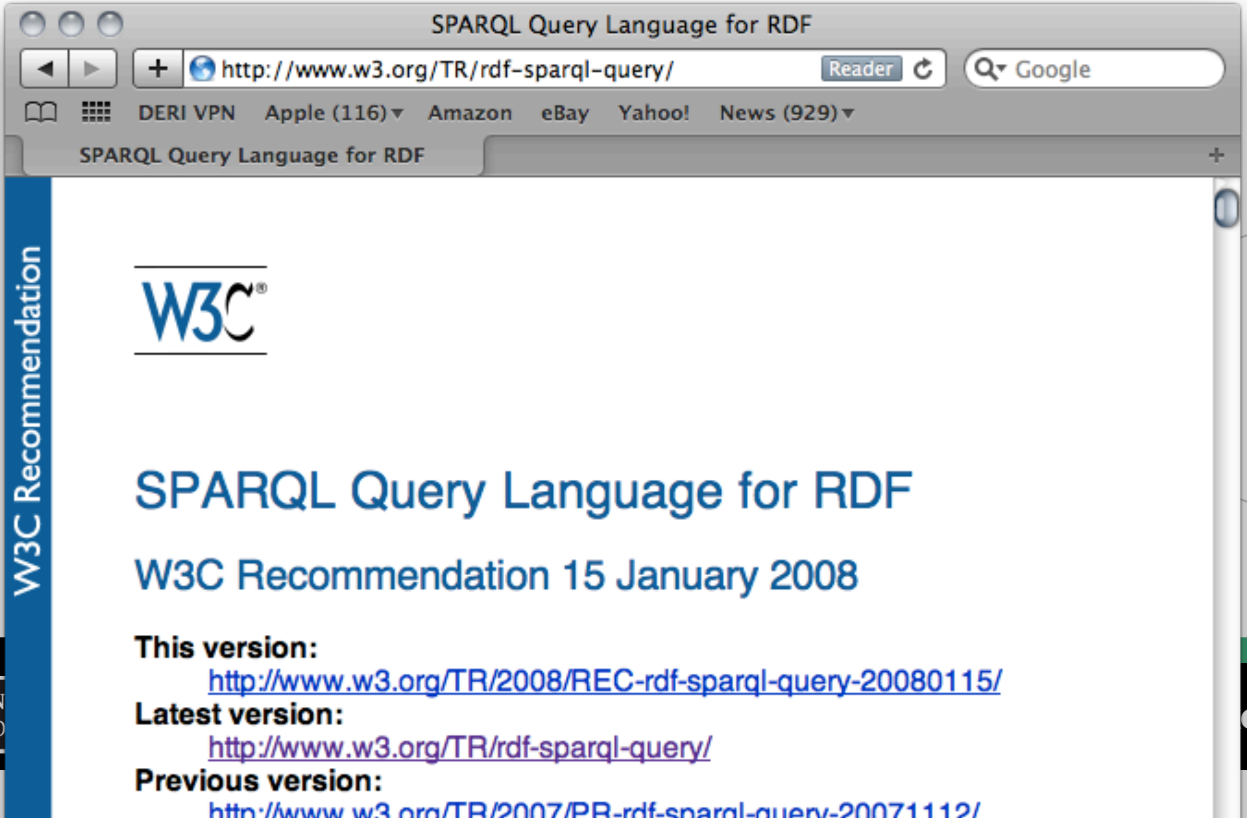
RDF Data online: Example – Turtle Syntax

□ 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 ;
                                                   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" .
```

Linked Data: What’s the point?

- Loads of **structured data** out there
- You want to do **structured queries** on top of it …
- SPARQL1.0 W3C Rec 15 January 2008… Now you can!
- Without exaggeration, SPARQL is probably a not too small a part of the LOD success story! … at least an important building block


SPARQL Query Language for RDF
http://www.w3.org/TR/rdf-sparql-query/
Reader
Google
DERI VPN
Apple (116)
Amazon
eBay
Yahoo!
News (929)
SPARQL Query Language for RDF
W3C Recommendation
W3C®

SPARQL Query Language for RDF

W3C Recommendation 15 January 2008

This version:
http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/
Latest version:
http://www.w3.org/TR/rdf-sparql-query/
Previous version:
http://www.w3.org/TR/2007/PR-rdf-sparql-query-20071112/

How can I query that data? SPARQL

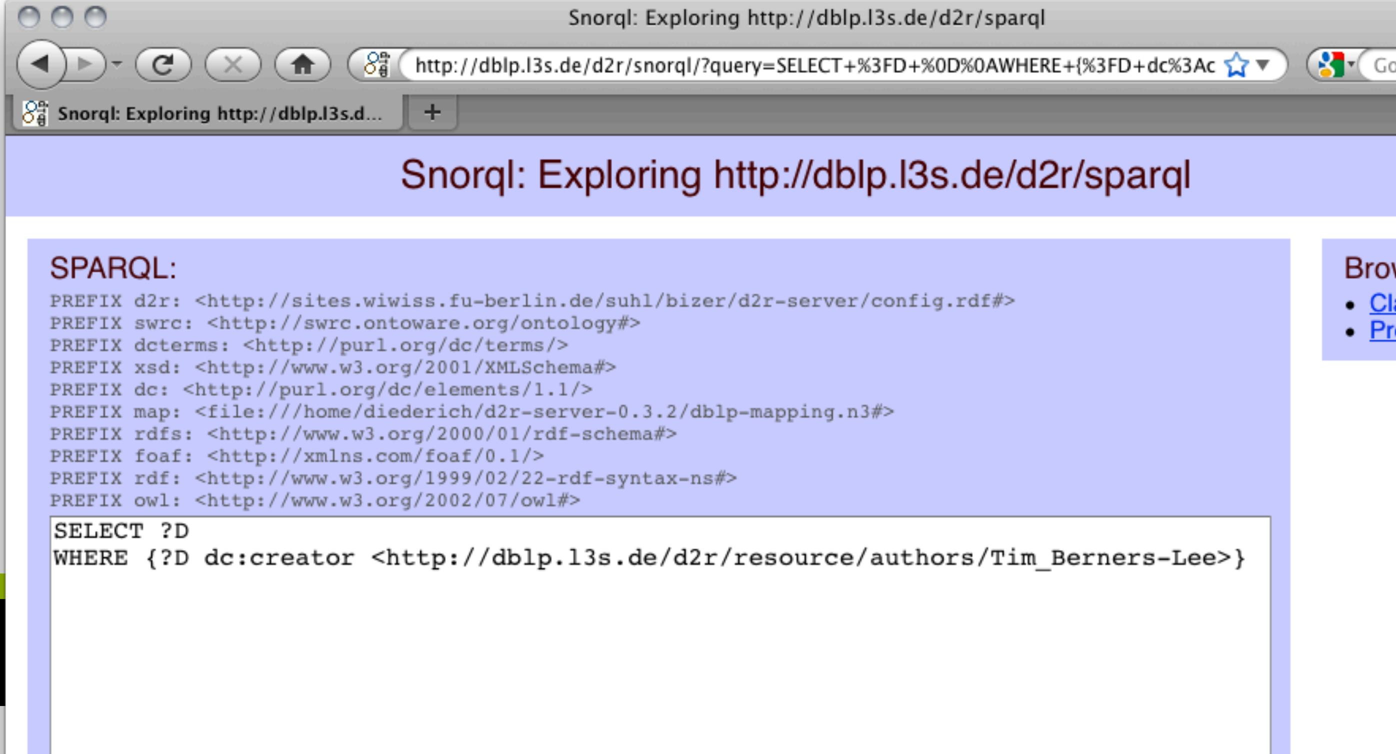
Basic graph pattern matching ~ Conjunctive queries

Example DBLP:

“Give me all documents by Tim Berners-Lee”

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?D
FROM <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>
WHERE {?D foaf:maker <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>}
```

FROM clause/Dataset can be implicit, e.g. when querying DBLP’s SPARQL endpoint


Snorql: Exploring http://dblp.l3s.de/d2r/sparql
http://dblp.l3s.de/d2r/snorql/?query=SELECT+%3FD+%0D%0AWHERE+{%3FD+dc%3Ac
Snorql: Exploring http://dblp.l3s.d...
Bro
Cl
Pr

Snorql: Exploring http://dblp.l3s.de/d2r/sparql

SPARQL:

```
PREFIX d2r: <http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2r-server/config.rdf#>
PREFIX swrc: <http://swrc.ontoware.org/ontology#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX map: <file:///home/diederich/d2r-server-0.3.2/dblp-mapping.n3#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?D
WHERE {?D dc:creator <http://dblp.l3s.de/d2r/resource/authors/Tim_Berners-Lee>}
```

SPARQL: Basic Graph Patterns

Basic graph pattern matching ~ Conjunctive queries

Example DBPEDIA:

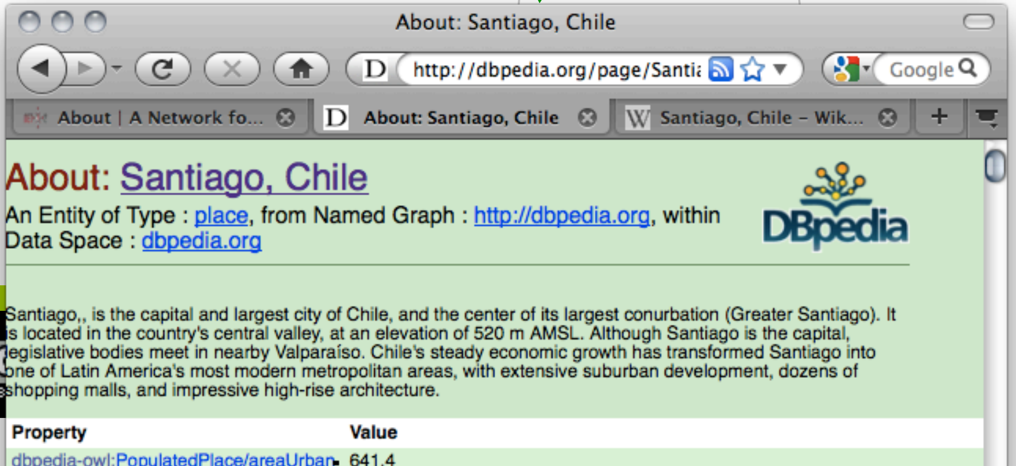
“Give me all names of people born in Santiago”

Basic Graph Pattern (BGP) *… set of RDF triples with variables in S,P,O , e.g.:*

```
{?P "born in" <http://dbpedia.org/resource/Santiago%2C_Chile>;
    "name" ?N }
```

How can I find the right properties for my query?

→ Look at the data!



About: Santiago, Chile

An Entity of Type : place, from Named Graph : http://dbpedia.org, within Data Space : dbpedia.org

Santiago,, is the capital and largest city of Chile, and the center of its largest conurbation (Greater Santiago). It is located in the country's central valley, at an elevation of 520 m AMSL. Although Santiago is the capital, legislative bodies meet in nearby Valparaíso. Chile's steady economic growth has transformed Santiago into one of Latin America's most modern metropolitan areas, with extensive suburban development, dozens of shopping malls, and impressive high-rise architecture.

Property	Value
dbpedia-owl:PopulatedPlace/areaUrban	641.4

SPARQL: Basic Graph Patterns

Basic graph pattern matching ~ Conjunctive queries

Example DBPEDIA:

“Give me all names of people born in Santiago”

Basic Graph Pattern (BGP) *… set of RDF triples with variables in S,P,O , e.g.:*

```
{?P dbpedia-owl:birthPlace <http://dbpedia.org/resource/Santiago%2C_Chile>;
    rdfs:label ?N }
```

SPARQL: Basic Graph Patterns

Basic graph pattern matching ~ Conjunctive queries

Example DBPEDIA:

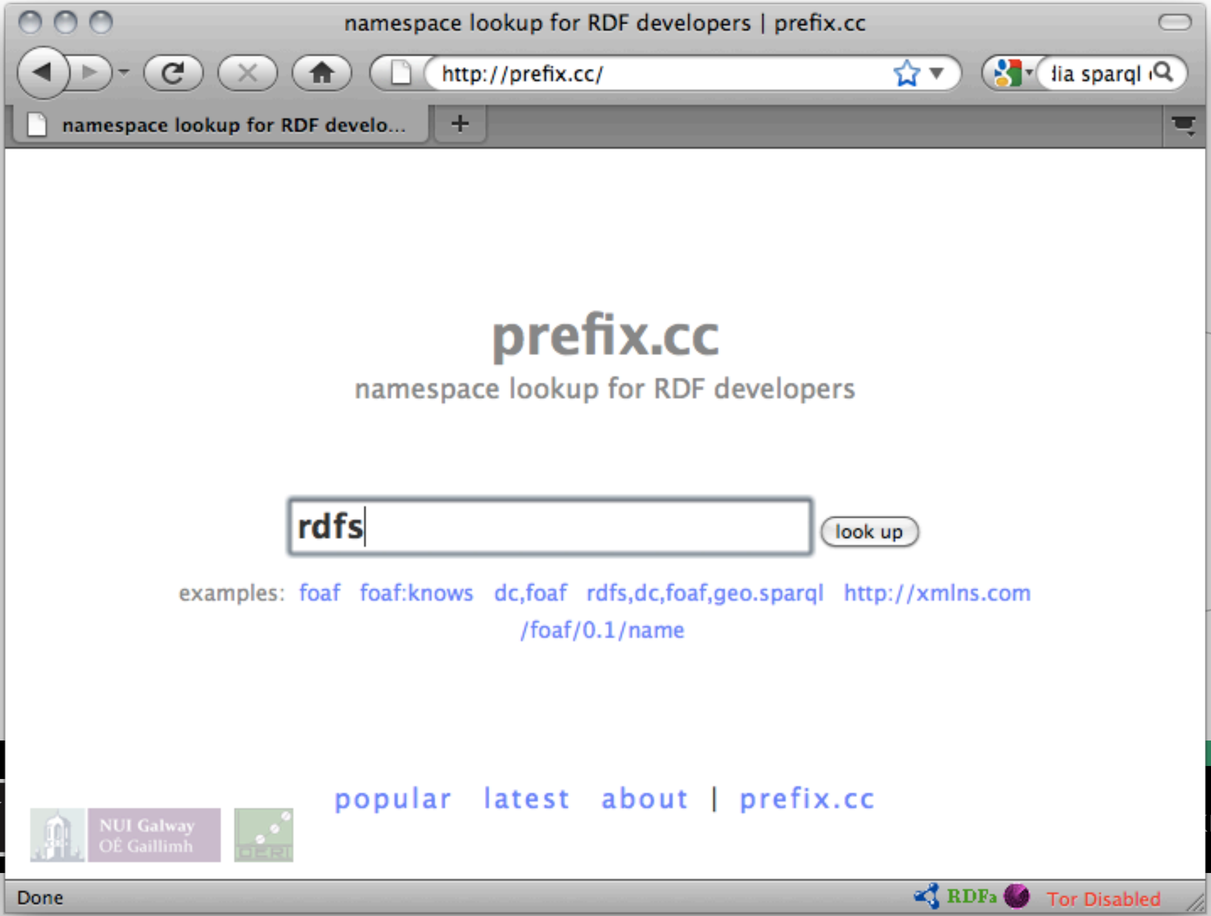
“Give me all names of people born in Santiago”

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?N
{?P dbpedia-owl:birthPlace <http://dbpedia.org/resource/Santiago%2C_Chile>;
    rdfs:label ?N }
```

Lesson learned: I can build SPARQL queries, from looking at the data and the URIs used (for properties and classes) in the data!

SPARQL: and how should I know all those prefixes? E.g. use prexif.cc !!!

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?N
{?P dbpedia-owl:birthPlace <http://dbpedia.org/resource/Santiago
%2C_Chile>;
    rdfs:label ?N }
```


namespace lookup for RDF developers | prefix.cc
http://prefix.cc/
dia sparql
namespace lookup for RDF develo...
prefix.cc
namespace lookup for RDF developers
rdfs
look up
examples: foaf foaf:knows dc,foaf rdfs,dc,foaf,geo.sparql http://xmlns.com/foaf/0.1/name
popular latest about | prefix.cc
NUI Galway OÉ Gaillimh
Done
RDFa
Tor Disabled

SPARQL: Basic Graph Patterns

Basic graph pattern matching ~ **Conjunctive** queries

Example DBLP:

“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

SPARQL: Basic Graph Patterns

Avoid Duplicates: keyword DISTINCT

Example DBLP:

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

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

SPARQL: Basic Graph Patterns

Basic graph pattern matching ~ **Conjunctive** queries

Example DBLP:

“Give me all names of co-authors of Tim Berners-Lee, their identifiers and their authored documents”

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

*“ SELECT * ” outputs all variables in the pattern*

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…

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 {



      }
```

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”
"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_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”
…

∪

?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_Berners-Lee
                       [ 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

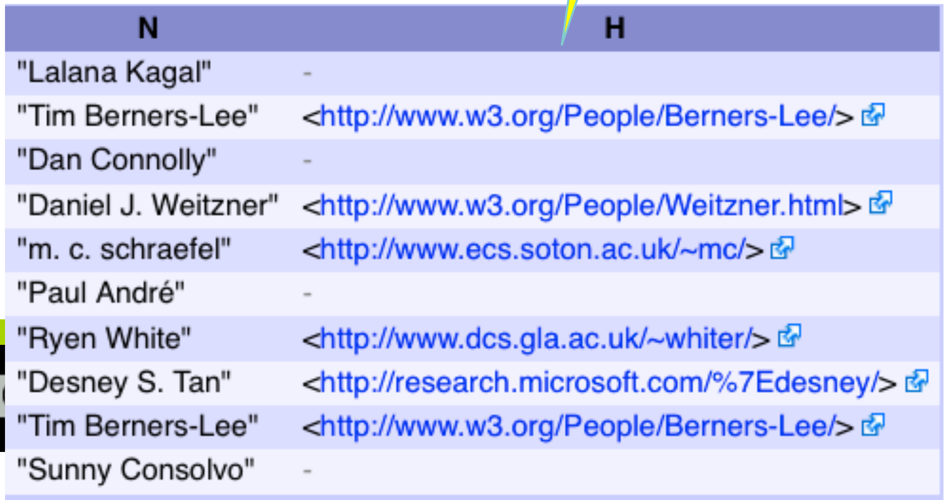
Optional parts in queries (Left Outer Join)

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_Berner -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"	-

FILTERING out query results

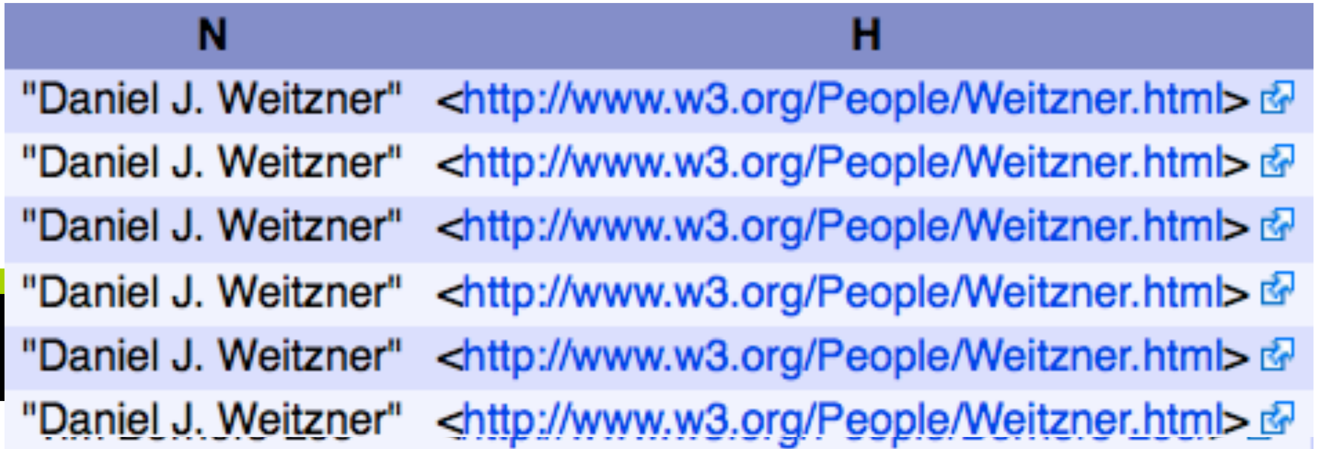
FILTERs allow to specify FILTER conditions on patterns

Example:

“Give me all names of co-authors of Tim Berners-Lee

*and **whose homepage starts with http://www.w3 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> )
     }
```

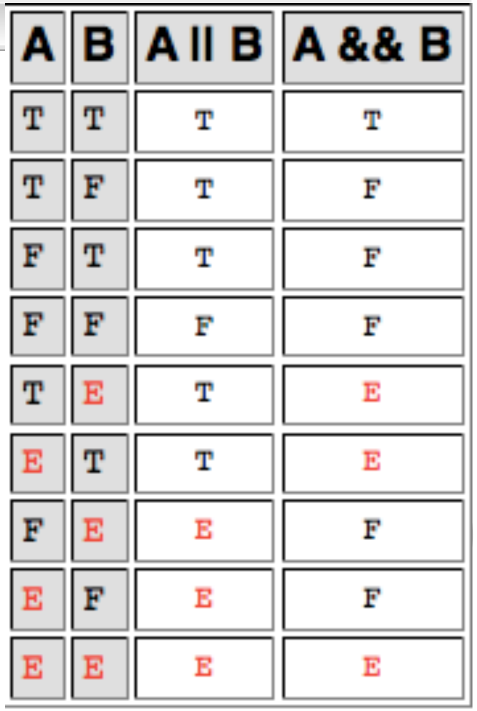


N	H
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>
"Daniel J. Weitzner"	<http://www.w3.org/People/Weitzner.html>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

FILTERING out query results

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**



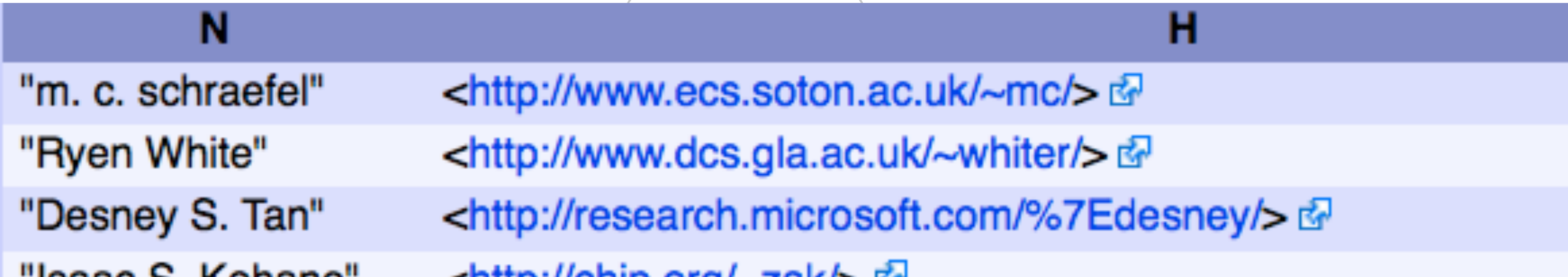
A	B	A ll B	A && B
T	T	T	T
T	F	T	F
F	T	T	F
F	F	F	F
T	E	T	E
E	T	T	E
F	E	E	F
E	F	E	F
E	E	E	E

A	!A
T	F
F	T
E	E

Example:

```
PREFIX foaf: <http://xmlns.com
SELECT ?N ?H
WHERE {
       ?D foaf:maker <http://dblp.l3      e/…/authors/Tim_Berners-Lee>.
       ?D foaf:maker ?CoAuth . ?Co    n foaf:name ?N .
       OPTIONAL { ?CoAuth foaf:h   mepage ?H . }
       FILTER( ! regex( str(?H) , "^http://www.w3" ) &&
       ?CoAuth != <http://dblp.l3s.de/…/authors/Tim_Berners-Lee> )
      }
```

Will result in E for unbound ?H
➔Whole FILTER expr
always E for unbound ?H



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/>

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 ) }
```

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 AS

Querying named GRAPHs

- *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.

Link

Slicing and Dicing results

- Solution Modifiers
  - □ DISTINCT/REDUCED
  - □ ORDER BY
  - □ LIMIT/OFFSET

- Example:

```
SELECT DISTINCT ?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) . }
ORDER BY ?person
LIMIT 10
```

Link

- ASC, DESC, ORDER BY Expressions

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

- “Conditional OPTIONAL”
  - □ *“Give me the names and - **only of those whose name sta s with ‘D’** - the homepage”*

```
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.**

More complex query examples 2/2

- Negation (“NOT EXISTS” in SQL)
  - □ *“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 even weirder combination of OPTIONAL with GRAPH queries…

```
SELECT ?X
WHE
```

Please forget this immediately again…

“These aren’t the droids you’re looking for”

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

Constructing Graphs

Construct new graphs:

- *“everybody knows their co-authors”*

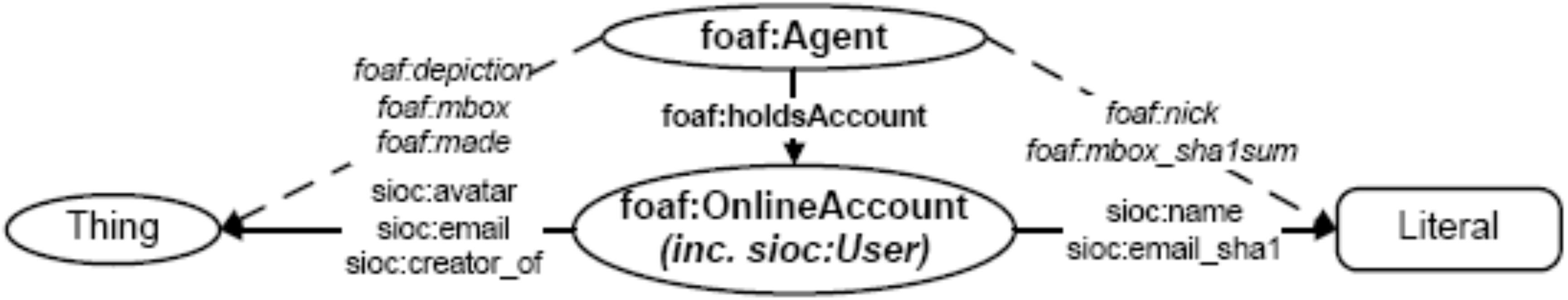
```
CONSTRUCT { ?X foaf:knows ?Y }
WHERE{ ?D foaf:maker ?X, ?Y .
               FILTER( ?X != ?Y ) }
```

Constructing Graphs

- Map between ontologies:
- E.g. for expressing complex ontology mappings between FOAF and SIOC
- "an sioc:name of a sioc:User is a foaf:nick"

Actually, expressible in new OWL2 (but not in OWL1):

```
foaf:nick owl:propertyChainAxiom (foaf:holdsAccount sioc:name)
```



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 (?Pr * 1.1 AS ?NewP )
WHERE { ?Item ex:price ?Pr }
```

Data:

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

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

ex:liqueur1      ex:price "n/a".
```

Results:

Leaves errors unbound!

?Item	?NewP
lemonade1	3.3
beer1	3.3
wine1	3.85
liqueur1	

Project expressions – Restriction:

- Assignments, Creating new values…

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

Note: Variables "already bound" cannot be used for project expressions!

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
```

Dat

```
@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.
```

?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:

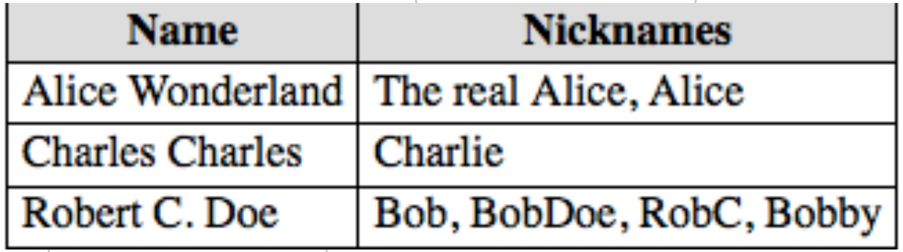
?T	?C
Softdrink	3
Beer	3
Wine	7.5

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 “Limit per resource”

- 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**

Subqueries – Known Limitations

- Attention: Subqueries do not allow to “inject values” from outside, but that limits some use cases, one might think of… e.g. an alternative “limit per resource” query:

```
SELECT ?P ?T
WHERE {  ?P rdf:type Person .
         { SELECT ?T
           WHERE { ?D foaf:maker ?P ; dc:title ?T }
           LIMIT 3 }
      }
```

Different ?P/ different scope than the ?P outside of the subquery… i.e. no correlation

```
:jim a foaf:Person .
:tim a foaf:Person .

:d1 foaf:maker :tim, :jim; dc:title "Doc1" .
:d2 foaf:maker :tim, :jim; dc:title "Doc2" .
:d3 foaf:maker :jim; dc:title "Doc3" .
:d4 foaf:maker :tim; dc:title "Doc4" .
```

?P	?T
:jim	“Doc1”
:jim	“Doc2”
:jim	“Doc3”
:tim	“Doc1”
:tim	“Doc2”
:tim	“Doc3”

- … does **NOT** return 3 titles per author!

FROM in Subqueries? NO!

- Note: At this point, no Dataset Clauses in Subselects, i.e.:

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

MINUS and NOT EXISTS

MINUS and NOT EXISTS

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

```
SELECT ?X
WHERE{ ?X rdf:type foaf:Person
       MINUS { ?X foaf:homepage ?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

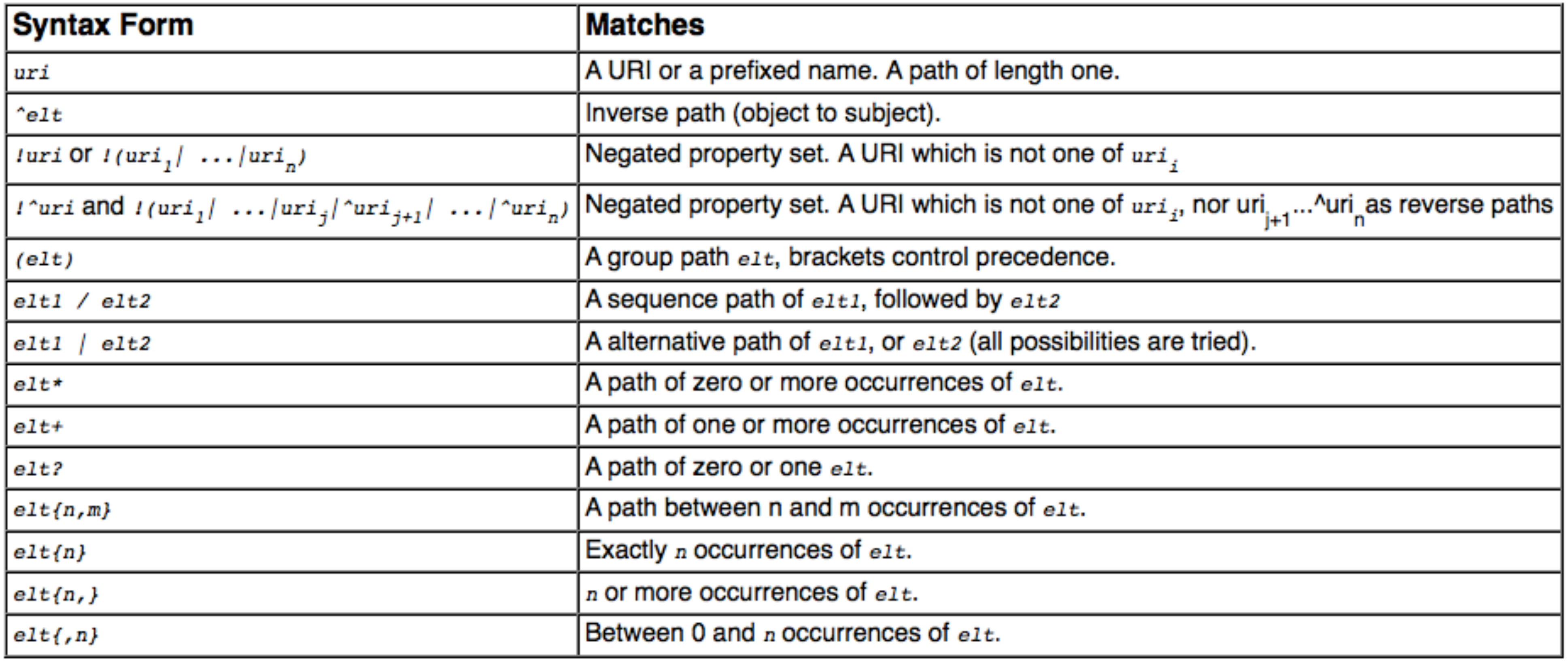
Property Path expressions

- 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 .
}
```

Link

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

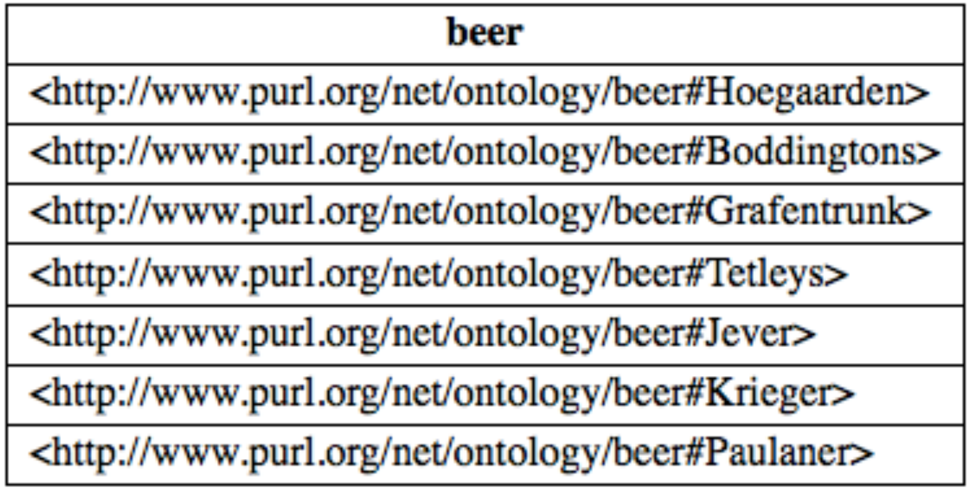
- SPARQL1.1 will define SPARQL query answering over OWL2 ontologies and RIF rule sets:
  - □ http://www.w3.org/TR/sparql11-entailment/
  - □ RDF Entailment Regime
  - □ RDFS Entailment Regime
  - □ D-Entailment Regime
  - □ OWL 2 RDF-Based Semantics Entailment Regime
  - □ OWL 2 Direct Semantics Entailment 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
<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>

```
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.
```

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`

SPARQL1.1+RDFS/OWL: Challenges+Pitfalls

- Challenges+Pitfalls:
  - □ Possibly Infinite answers (by RDFS ContainerMembership properties, OWL datatype reasoning, etc.)
  - □ Conjunctive Queries: non-distinguished variables
  - □ SPARQL 1.1 features: Aggregates

SPARQL1.1+RDFS/OWL: Challenges+Pitfalls

- Current Solution:
  - □ Possibly Infinite answers (by RDFS ContainerMembership properties, OWL datatype reasoning, etc.)
    - – Restrict answers to rdf:/rdfs:/owl:vocabulary minus rdf:_1 … rdf:_n plus terms occurring in the data graph
  - □ Non-distinguished variables
    - – *No non-distinguished variables, answers must result from BGP matching, projection a post-processing step not part of SPARQL entailment regimes.*
  - □ SPARQL 1.1 other features: e.g. Aggregates, etc.
    - – *Again not affected, answers must result from BGP matching, projection a post-processing step not part of entailment.*
  - □ Simple, BUT: maybe not always entirelty intuitive, so
    - – Good to know ;-)

Possibly Infinite answers RDF(S): Container Membership

- Graph:

```
:rr2010Proceedings :hasEditors [ a rdf:Seq;
                                   rdf:_1 :pascal_hitzler;
                                   rdf:_2 :thomas_lukasiewicz
                                 ] .
```

Query with RDFS Entailment in mind:

```
SELECT ?CM { ?CM a rdfs:ContainerMembershipProperty}
```

Entailed by RDFS (axiomatic Triples):

```
rdfs:_1 a rdfs:ContainerMembershipProperty .
rdfs:_2 a rdfs:ContainerMembershipProperty .
rdfs:_3 a rdfs:ContainerMembershipProperty .
rdfs:_4 a rdfs:ContainerMembershipProperty .
...
```

Possibly Infinite answers RDF(S): Container Membership

- Graph:

```
:rr2010Proceedings :hasEditors [ a rdf:Seq;
                                 rdf:_1 :pascal_hitzler;
                                 rdf:_2 :thomas_lukasiewicz
                               ] .
```

Query with RDFS Entailment in mind:

```
SELECT ?CM { ?CM a rdfs:ContainerMembershipProperty}
```

SPARQL 1.1 restricts answers to rdf:/rdfs:/owl:vocabulary minus rdf:_1 … rdf:_n **plus terms occurring in the data graph**

So, the only answers in SPARQL1.1 are:

```
{ ?CM/rdfs:_1, ?CM/rdfs:_2, }
```

Non-distinguished variables:

- E.g. Graph

```
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 ?Y { ?X :hasFather ?Y }
```

No answer, because no known value for ?Y in the data graph.

Non-distinguished variables:

- E.g. Graph

```
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 :hasFather ?Y }
```

But what about this one? ?Y looks like a "non-distinguished" variable

Solution: In SPARQL 1.1 answers must result from BGP matching, projection a post-processing step not part of entailment ➔ so, still no answer.

SPARQL 1.1/OWL other features: Aggregates

- Similar as before… aggregates are evaluated within algebra **after** BGP matching, so, no effect:

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

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

Under RDFS/OWL entailment returns : {?X/jeff, ?X/aidan}

SPARQL 1.1/OWL other features: Aggregates

- Similar as before… aggregates are evaluated as post-processing **after** BGP matching, so, no effect:

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

Attention! owl:sameAs inference does **NOT** affect counting!!! …
But bnodes do!

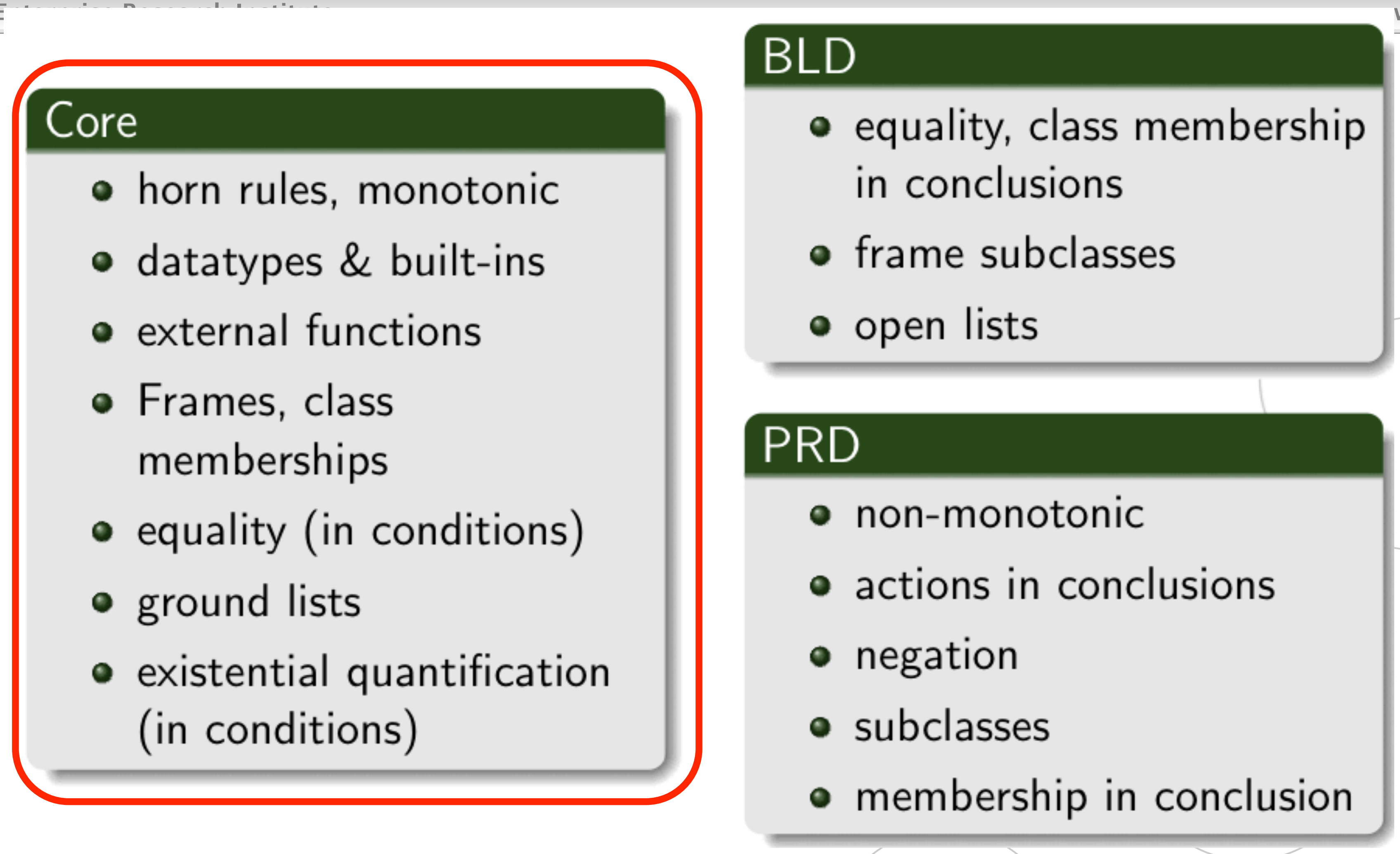
```
SELECT (Count(?X) AS ?Y) { ?X a foaf:Person }
```

Under RDFS/OWL entailment returns : {?Y/3}

SPARQL1.1 + RIF

- RIF … Rule Interchange format, Rec. since 2010
  - □ RIF: Rule Interchange Format (rather than Rule language)
    - Framework for Rule Languages
    - Support RDF import: interesting for rule languages on top of RDF
    - Built-Ins support (close to XPath/XQuery functions and operators)
    - RIF Dialects:
      - RIF BLD: basic logic dialect = Horn rules with Built-ins, Equality
      - RIF Core: Datalog fragment (no logical function symbols, no head-equality)
      - RIF PRD: Production rules dialect
    - Normative XML syntax
  - □ Commonalities with OWL:
    - RIF can model OWL2 RL
    - Share same Datatypes (XSD Datatypes, most OWL2 Datatypes)
    - Combinations of RIF with RDF, RDFS, and OWL defined in: http://www.w3.org/TR/rif-rdf-owl/

RIF Dialects



Core

- horn rules, monotonic
- datatypes & built-ins
- external functions
- Frames, class memberships
- equality (in conditions)
- ground lists
- existential quantification (in conditions)

BLD

- equality, class membership in conclusions
- frame subclasses
- open lists

PRD

- non-monotonic
- actions in conclusions
- negation
- subclasses
- membership in conclusion

SPARQL1.1 so far only defines
Entailment for RIF Core… room for improvement (cf. e.g. Demo Obermeier et al. RR2010)

SPARQL1.1 + RIF Core + RDFS/OWL

- RIF Core allows to encode RDFS, 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 . }
```

- RIF Core allows to encode OWL2 RL, e.g. :

```
owl1: { ?S1 owl:SameAs ?S2 } :-
        { ?S1 ?P ?O . ?S2 ?P ?O . ?P rdf:type owl:InverseFunctionalProperty}
owl2: { ?Y ?P ?O } :- { ?X owl:SameAs ?Y . ?X ?P ?O }
owl3: { ?S ?Y ?O } :- { ?X owl:SameAs ?Y . ?S ?X ?O }
owl4: { ?S ?P ?Y } :- { ?X owl:SameAs ?Y . ?S ?P ?X }
```

- Plus more (custom rules, including Built-ins):

```
{?X foaf:name ?FullN } :- { ?X foaf:firstName ?F. ?X foaf:lastName ?L }
                            AND ?FullN = fn:concat(?F, " ", ?L)
```

<http://ruleset1.rif>

How to reference to a RIF Ruleset from SPARQL?

- In OWL Entailment Regime, OWL is assumed to be part of the RDF Graph (OWL/RDF)

- RIF's so far only a normative syntax is RIF/XML
  - □ RIF encoding in RDF (RIF/RDF) underway: http://www.w3.org/2005/rules/wiki/RIF_In_RDF
  - □ Will also provide a new RDF property `rif:usedWithProfile` to import RIF rulesets (in RIF/XML or RIF/RDF). e.g.

In current draft called `rif:imports`

```
<http://ruleset1.rif> rif:usedWithProfile
         <http://www.w3.org/ns/entailment/Simple> .
<http://dblp.l3s…/Tim_Berners-Lee>
         foaf:homepage <http://www.w3.org/People/Berners-Lee/> ;
         foaf:name "Tim Berners-Lee" .
<http://www.w3.org/People/Berners-Lee/card#i>
         foaf:homepage <http://www.w3.org/People/Berners-Lee/> ;
         foaf:firstName "Timothy”;
         foaf:lastName "Berners-Lee" .
```

```
SELECT ?P ?N { ?P foaf:name ?N }
```

?P	?N
<dblp/Tim>	Tim Berners-Lee
<w3/B-Lee/card#i>	Tim Berners-Lee
<dblp/Tim>	Timothy Berners-Lee
<w3/B-Lee/card#i>	Timothy Berners-Lee

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

Extended Function Library

- Functions Library in SPARQL1.0 is insufficient:
  - □ Bound(.)
  - □ isLiteral(.)
  - □ isBlank(.)
  - □ isIRI(.)
  - □ Str(.)
  - □ Regex(. , .)
  - □ +,-,*, <, >, =
- New functions to be included in standard library:
  - □ COALESCE, IF
  - □ Functions from the Xpath/Xquery function library
    - – String manipulation, more math, etc. … e.g. fn:concat

Essentially: rubber-stamp common functions present in current implementations

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 {
```

Tim's FOAF file

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

DBLP SPARQL endpoint

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

}
```

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 ] ] . } }
      }
```

Basic Federated Queries - BINDINGS

- Sometimes you want to “inject” or “fix” some bindings into the query to be sent to an external endpoint.
- Goal: reduce data to be transferred:

- Example:

```
… WHERE { ?s :p2 ?v2 } BINDINGS ?s ?v2 { ( <s1> 7 ) ( <s2> UNBOUND ) }
```

```
… WHERE { { ?s :p2 ?v2 }
          { {SELECT ( <s1> AS ?s ) ( 7 AS ?v2 ) WHERE {} }
            UNION
            {SELECT ( <s2> AS ?s ) WHERE {} }
```

➔ i.e. can be viewed as “syntactic sugar”, may be helpful…

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

Acknowledgements

- The members of the W3C SPARQL WG, particularly Lee Feigenbaum, who I stole some examples from
- The members of the W3C RIF WG
- The EU project NET2 which sponsored my visit.