

SPARQL1.1: new features and friends (OWL2, RIF)

@AxelPolleres

Digital Enterprise Research Institute, National University of Ireland, Galway

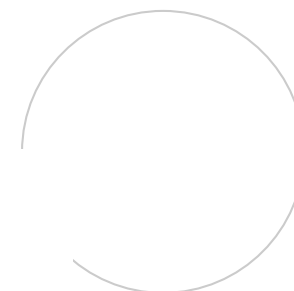
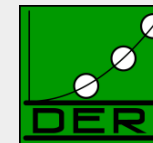


These slides are provided under creative commons
Attribution-NonCommercial-ShareAlike 3.0 Unported License!

© Copyright 2009 Digital Enterprise Research Institute. All rights reserved.

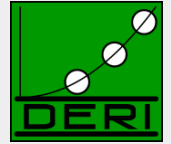


Disclaimer



Chair-hat off: While trying to convey best I can the current status of the SPARQL Working Group, in this Tutorial I don't claim to speak officially for the group and will freely blend in my personal opinions... ;-)

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

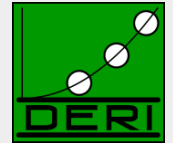
■ SPARQL 1.0 (standard since 2008):

- Query Language
- Protocol
- Result Format

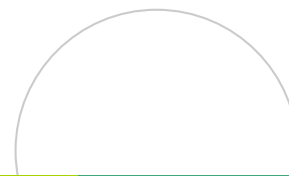
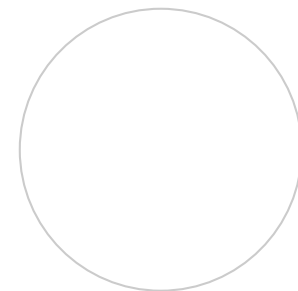
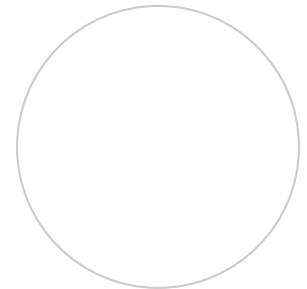
■ SPARQL 1.1 (in progress):

- **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

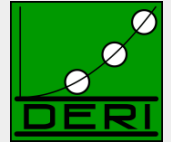
What you'll hear



- Run through SPARQL1.0
- SPARQL 1.0 Formal Semantics, and theoretical results
- New features in SPARQL 1.1 Query
- SPARQL 1.1 Entailment Regimes
 - Relation to OWL2
 - Relation to RIF
- Implementations, Status



RDF a plain data format for the Web



```
<http://dbpedia.org/resource/Brixen> <http://ontology.dumontierlab.com/isLocatedIn>  
    <http://dbpedia.org/resource/Italy> .  
<http://dbpedia.org/resource/Brixen> <http://www.w3.org/2000/01/rdf-schema#label>  
    "Bressanone"@it .  
  
_:x <http://www.w3.org/2000/01/rdf-schema#label> "Diego Calvanese" .  
_:x <http://ontology.dumontierlab.com/isLocatedIn> <http://dbpedia.org/resource/Brixen> .
```

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

Subject U x B

Predicate U

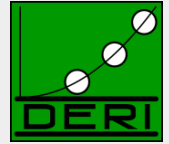
Object U x B x L

URIs, e.g.
<http://www.w3.org/2000/01/rdf-schema#label>
<http://ontology.dumontierlab.com/isLocatedIn>
<http://dbpedia.org/resource/Brixen>
<http://dbpedia.org/resource/Italy>

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

Literals, e.g.
`"2010"^^xsd:gYear`
`"Brixen"@de`
`"Bressanone"@it`
`"Diego Calvanese"`

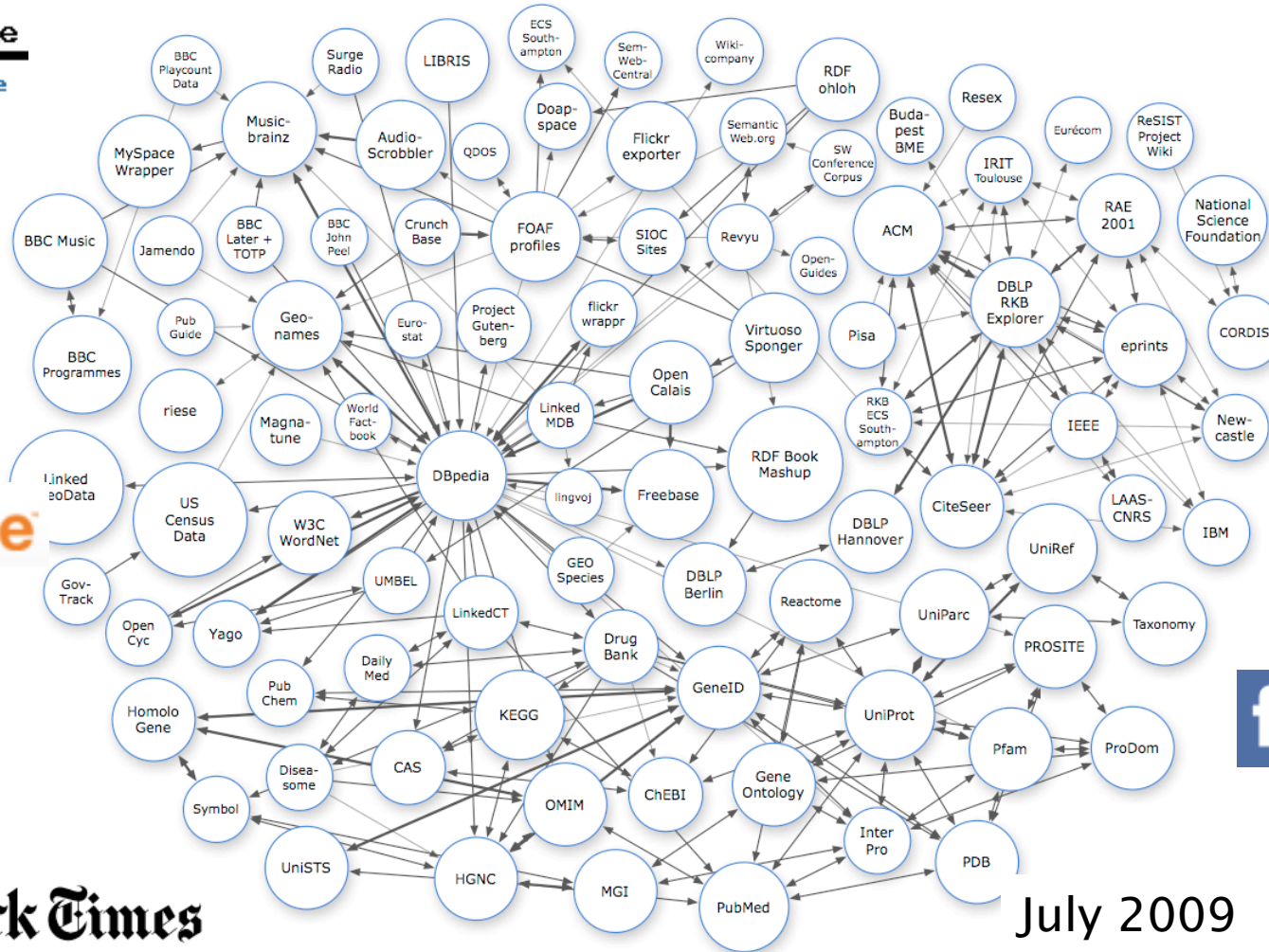
RDF Data on the Web: Linked Open Data



Digital Enterprise Research Institute

www.deri.ie

dblp.uni-trier.de
Computer Science
Bibliography



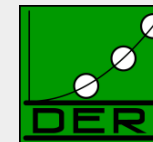
Freebase

facebook

The New York Times

July 2009

RDF Data online: Example 1/3



- (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

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-->

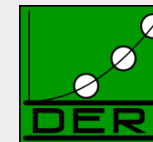
<rdf:RDF xmlns="http://xmlns.com/foaf/0.1/"
  xmlns:cc="http://creativecommons.org/ns#"
  xmlns:con="http://www.w3.org/2000/10/swap/pim/contact#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  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="http://www.w3.org/2002/01/tr-automation/tr.rdf">
    <dc:title>W3C Standards and Technical Reports</dc:title>
  </rdf:Description>

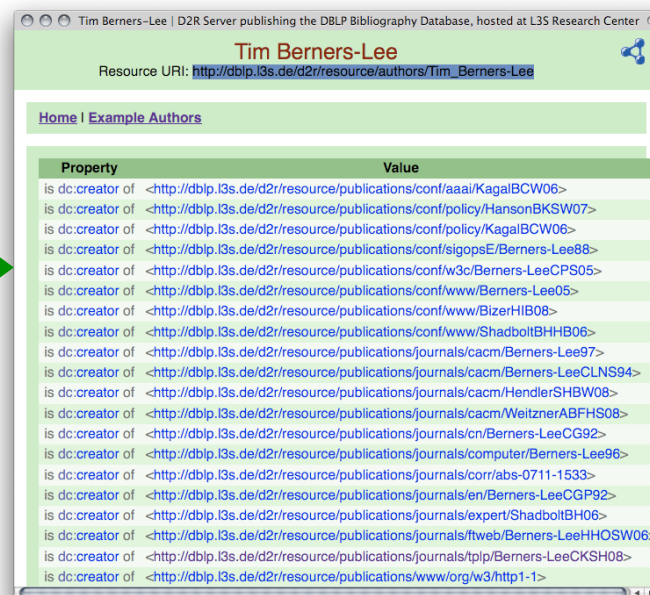
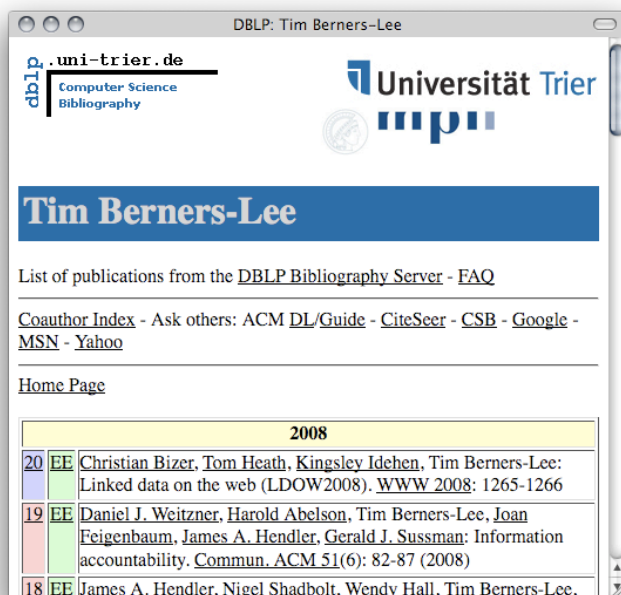
  <PersonalProfileDocument rdf:about="">
    <cc:license rdf:resource="http://creativecommons.org/licenses/by-nc/3.0/" />
    <dc:title>Tim Berners-Lee's FOAF file</dc:title>
    <maker rdf:resource="http://www.w3.org/People/Berners-Lee/card#i" />
    <primaryTopic rdf:resource="http://www.w3.org/People/Berners-Lee/card#i"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

Line 417, Col 11
```

RDF Data online: Example 2/3



- (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/>)



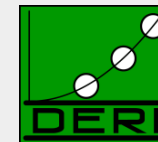
Gives unique URIs to authors, documents, etc. on DBLP! E.g.,

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

<http://dblp.l3s.de/d2r/resource/publications/journals/tlp/Berners-LeeCKSH08>

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

RDF Data online: Example 3/3



Tim Berners-Lee | D2R Server publishing the DBLP Bibliography Database, hosted a...

http://dblp.l3s.de/d2r/resource/authors/T

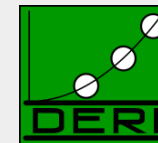
Tim Berners-Lee | D2R Server pub...

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/HansonBKS07
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/HansonBKS07
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

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.13s.../journals/tplp/Berners-LeeCKSH08> rdf:type swrc:Article.
```

```
<http://dblp.13s.../journals/tplp/Berners-LeeCKSH08> dcterms:issued "2008"^^xsd:gYear .
```

```
<http://dblp.13s.../journals/tplp/Berners-LeeCKSH08> foaf:maker <http://dblp.13s.../Tim_Berners-Lee> .
```

```
<http://dblp.13s.../journals/tplp/Berners-LeeCKSH08> foaf:maker <http://dblp.13s.../Dan_Connolly> .
```

```
<http://dblp.13s.../journals/tplp/Berners-LeeCKSH08> foaf:maker <http://dblp.13s.../Jim_Hendler> .
```

```
<http://dblp.13s.../journals/tplp/Berners-LeeCKSH08> foaf:maker <http://dblp.13s.../Lalana_Kagal> .
```

```
<http://dblp.13s.../journals/tplp/Berners-LeeCKSH08> foaf:maker <http://dblp.13s.../Yosi_Scharf> .
```

...

```
<http://dblp.13s.../conf/aaai/KagalBCW06> rdf:type swrc:inProceedings .
```

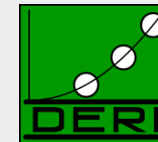
```
<http://dblp.13s.../conf/aaai/KagalBCW06> foaf:maker <http://dblp.13s.../Tim_Berners-Lee> .
```

...

```
<http://dblp.13s.../Tim_Berners-Lee> foaf:homepage <http://www.w3.org/People/Berners-Lee/> .
```

```
<http://dblp.13s.../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/> .
```

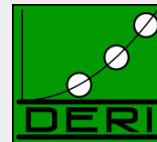
```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
@prefix swrc: <http://swrc.ontoware.org/ontology#> .
```

```
<http://dblp.13s.../journals/tplp/Berners-LeeCKSH08> rdf:type swrc:Article ;  
dcterms:issued "2008"^^xsd:gYear ;  
foaf:maker <http://dblp.13s.../Tim_Berners-Lee> ,  
          <http://dblp.13s.../Dan_Connolly> ,  
          <http://dblp.13s.../Jim_Hendler> ,  
          <http://dblp.13s.../Lalana_Kagal> ,  
          <http://dblp.13s.../Yosi_Scharf> .  
...  
<http://dblp.13s.../conf/aai/KagalBCW06> rdf:type swrc:inProceedings ;  
foaf:maker <http://dblp.13s.../Tim_Berners-Lee> .  
...  
<http://dblp.13s.../Tim_Berners-Lee> foaf:homepage <http://www.w3.org/People/Berners-Lee/> ;  
foaf:name "Tim Berners-Lee" .
```

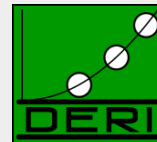
Linked Data: What's the point?



- 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



How can I query that data? SPARQL



Basic graph pattern matching ~ Conjunctive queries

Example:

“Give me all documents by Tim Berners-Lee”

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?D
FROM <http://dblp.13s.de/.../authors/Tim_Berners-Lee>
WHERE {?D foaf:maker <http://dblp.13s.de/.../authors/Tim_Berners-Lee>}
```

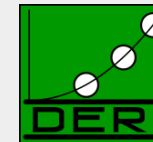
FROM clause/Dataset can be implicit, e.g. when querying DBLP's SPARQL endpoint

Snorql: Exploring http://dblp.13s.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.13s.de/d2r/resource/authors/Tim_Berners-Lee>}
```

How can I query that data? SPARQL



Basic graph pattern matching ~ Conjunctive queries

Example:

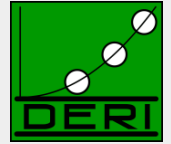
“Give me all names of co-authors of Tim Berners-Lee”

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N
WHERE { [ foaf:maker <http://dblp.13s.de/.../authors/Tim_Berners-Lee>,
        [ foaf:name ?N ] ] . }
```

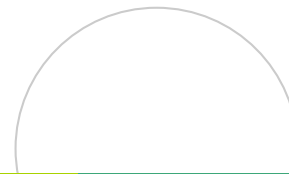
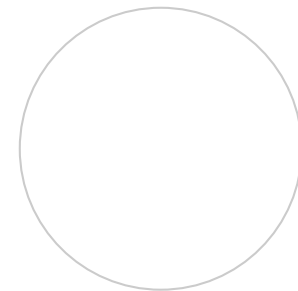
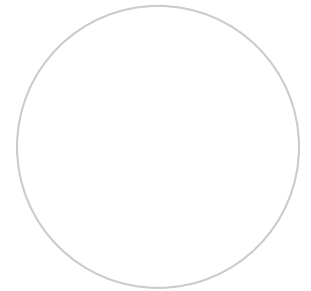
- Blank nodes in Queries play a *similar* role as (non-distinguished) variables.
- Turtle style shortcuts are allowed (*a bit extreme here, admittedly*)

[Link](#)

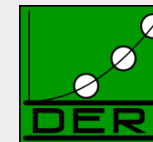
More complex patterns in SPARQL 1.0



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



UNIONS of conjunctive queries...



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: Duplicates possible, bag/multiset semantics!

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

U

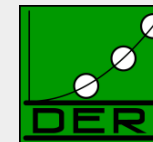
?N
"Michael Hausenblas"
"Jim Hendler"
"Charles McCathieNevile"
...

=

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



UNIONS of conjunctive queries...



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.13s.de/.../authors/Tim_Berners-Lee>,
    [ foaf:name ?N ] ] . }
  UNION
  { <http://www.w3.org/People/Berners-Lee/card#i> foaf:knows ?F .
    ?F foaf:name ?N }
}
```

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

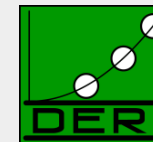
U

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

=

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

UNIONS of conjunctive queries...



Unions of conjunctive queries

Example:

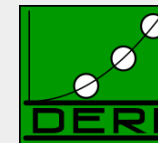
“Give me all names of co-authors or friends of Tim Berners-Lee”

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?CoAuthN ?FrN
WHERE {
  { [ foaf:maker <http://dblp.13s.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

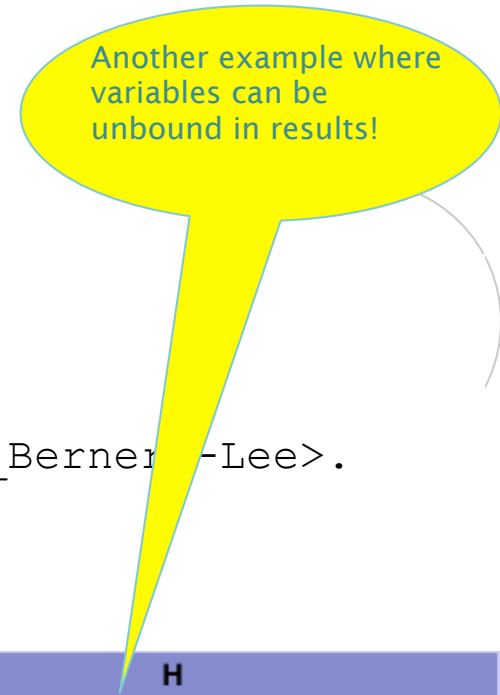


Optional parts in queries (Left Outer Join)

Example:

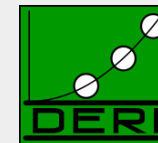
“Give me all names of co-authors of Tim Berners-Lee and optionally their homepage”

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N ?H
WHERE {
  ?D foaf:maker <http://dblp.13s.de/.../authors/Tim_Berners-Lee>.
  ?D foaf:maker ?CoAuth .
  ?CoAuth foaf:name ?N .
  OPTIONAL { ?CoAuth foaf:homepage ?H }
}
```



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

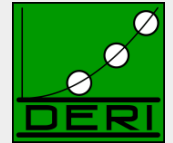
Example:

“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.13s.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.13s.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 B	A && B
T	T	T	T
T	F	T	F
F	T	T	F
F	F	F	F
T	E	T	E
E	T	T	E
F	E	E	F
E	F	E	F
E	E	E	E

Example:

```

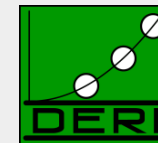
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N ?H
WHERE {
    ?D foaf:maker <http://dblp.13s.de/.../authors/Tim_Berners-Lee>.
    ?D foaf:maker ?CoAuth . ?CoAuth foaf:name ?N .
    OPTIONAL { ?CoAuth foaf:homepage ?H . }
    FILTER( ! regex( str(?H) , "^http://www.w3" ) &&
    ?CoAuth != <http://dblp.13s.de/.../authors/Tim_Berners-Lee> )
}
    
```

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

A	!A
T	F
F	T
E	E

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/>
"Igor S. Kabanov"	<http://chip.org/~zsk/>

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 ) }
```

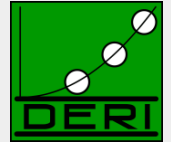
Non-safe variable in FILTERs are considered unbound. The Filter will just always result in **E**
→ Result always empty

- 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 SPARQL 1.0 doesn't have AS

Querying named GRAPHS

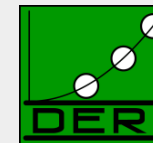


- *Find me people who have been involved with at least three ISWC or ESWC conference events.
(from SPARQL endpoint at data.semanticweb.org)*

```
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.

Slicing and Dicing results



■ Solution Modifiers

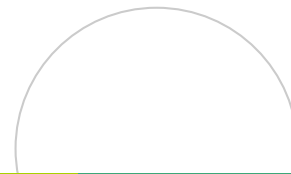
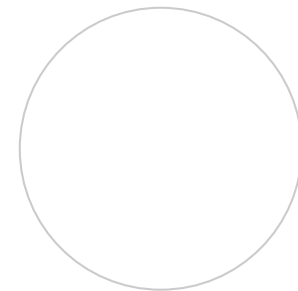
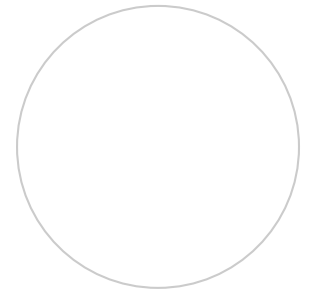
- DISTINCT/REDUCED
- ORDER BY
- LIMIT/OFFSET

■ Example:

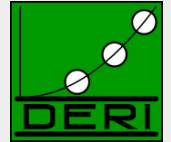
```
SELECT DISTINCT ?person WHERE {  
  GRAPH ?g1 { ?person a foaf:Person }  
  GRAPH ?g2 { ?person a foaf:Person }  
  GRAPH ?g3 { ?person a foaf:Person }  
  FILTER(?g1 != ?g2 && ?g1 != ?g3 && ?g2 != ?g3) . }  
  
ORDER BY ?person  
LIMIT 10
```

[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!
(?N is not considered
unsafe in this FILTER)*

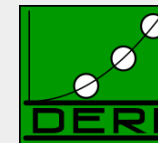
■ “Conditional OPTIONAL”

- *“Give me the names and - only of those whose name starts with ‘D’ - the homepage”*

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

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

More complex query examples 2/2



■ Negation

□ “Give me all Persons without a homepage”

□ **Option 1:** by combination of OPTIONAL and FILTER(!bound(...))

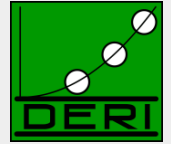
```
SELECT ?X
WHERE{ ?X rdf:type foaf:Person
      OPTIONAL { ?X foaf:homepage ?H }
      FILTER( !bound( ?H ) ) }
```

□ **Option 2:** by even weirder combination of OPTIONAL with GRAPH queries...

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

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

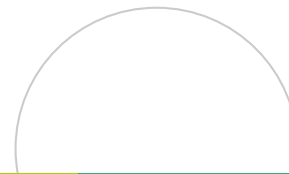
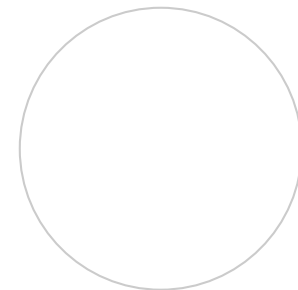
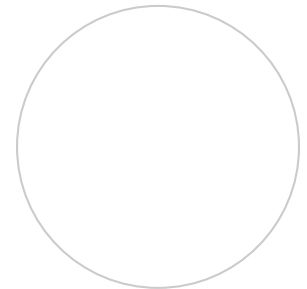
Constructing Graphs



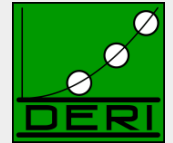
Construct new graphs:

- *“everybody knows their co-authors”*

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



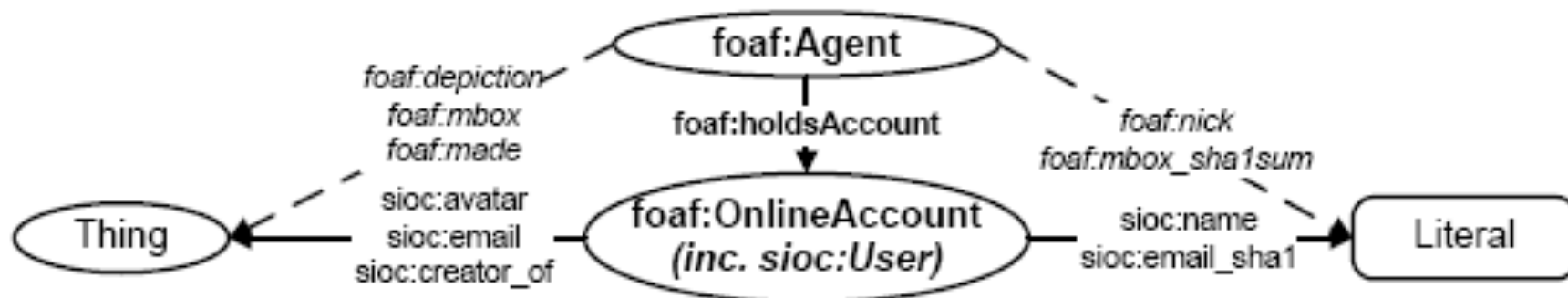
Constructing Graphs



- 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)`



■ 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)

■ *Graph patterns:*

- BGP
- $P1 P2$
- $P \text{ FILTER } R$
- $P1 \text{ UNION } P2$
- $P1 \text{ OPTIONAL } P2$

■ *Semantics*

- $eval(D(G), \text{graph pattern}) \dots$ 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 ...

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

Definition 1:

The evaluation of the BGP P over a graph G , denoted by $\text{eval}(P,G)$, is the set of all mappings μ such that:

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

$\mu(P) \subseteq G$

Example Graph:

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

Example Pattern:

$P = \{ ?X \text{ foaf:knows } ?Y \}.$

$\text{eval}(P,G) = \{ \mu_1 = \{ ?x \rightarrow :tim, ?y \rightarrow :jim \}, \mu_2 = \{ ?x \rightarrow :jim, ?y \rightarrow :tim \}, \mu_3 = \{ ?x \rightarrow :tim, ?y \rightarrow :lalana \} \}$

Definition 2:

mappings μ_1, μ_2 are compatible iff they agree in their shared variables.

Let M_1, M_2 be sets of mappings

Definition 3:

Join:

$M_1 \bowtie M_2 = \{ \mu_1 \cup \mu_2 \mid \mu_1 \in M_1, \mu_2 \in M_2, \text{ and } \mu_1, \mu_2 \text{ are compatible} \}$

Union:

$M_1 \cup M_2 = \{ \mu \mid \mu \in M_1 \text{ or } \mu \in M_2 \}$

Minus:

$M_1 \setminus M_2 = \{ \mu \in M_1 \mid \text{for all } \mu' \in M_2, \mu \text{ and } \mu' \text{ are not compatible} \}$

LeftJoin:

$M_1 \ltimes M_2 = (M_1 \bowtie M_2) \cup (M_1 \setminus M_2)$

Filter:

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

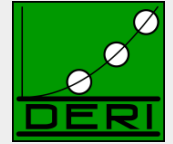
$P_1 \ P_2$

$P_1 \text{ UNION } P_2$

$P_1 \text{ OPTIONAL } P_2$

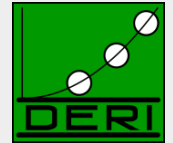
$P \text{ FILTER } R$

Semantics full as per [Perez et al.2006]



$$\begin{aligned} eval(BGP, G) & \quad \dots \text{ see Definition 1} \\ eval(P1 \text{ P2}, G) & \quad = eval(P1, G) \bowtie eval(P2, G) \\ eval(P1 \text{ UNION } P2, G) & \quad = eval(P1, G) \cup eval(P2, G) \\ eval(P1 \text{ OPTIONAL } P2, G) & \quad = eval(P1, G) \boxtimes eval(P2, G) \\ eval(P \text{ FILTER } R, G) & \quad = eval(P, G) |_R \end{aligned}$$

ISSUE 1) Recall from before: SPARQL allows duplicates !



Unions of conjunctive queries

Example:

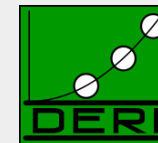
“Give me all names of co-authors or friends of Tim Berners-Lee”

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?N
WHERE {
  { [ foaf:maker <http://dblp.13s.de/.../authors/Tim_Berners-Lee>,
    [ foaf:name ?N ] ] . }
  UNION
  { <http://www.w3.org/People/Berners-Lee/card#i> foaf:knows ?F .
    ?F foaf:name ?N }
}
```

Note: Duplicates possible, bag/multiset semantics!

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

ISSUE 2) Recall from before: FILTERS can make OPTIONAL non-compositional!



■ “Conditional OPTIONAL”

- “Give me the names and only of those whose name starts with ‘D’ the homepage”

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

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

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

Definition 1:

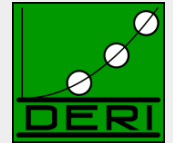
The evaluation of the BGP P over a graph G , denoted by $\text{eval}(P,G)$, is the set of all mappings μ such that:

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

$\mu(P) \models G$

- Remark: This behavior can likewise be achieved if you replace all bnodes in queries simply with variables ...
 - Definition in [Section 12.3.1 of SPARQL](#) spec counts cardinality of mappings exactly by number of instance mappings
 - i.e. behavior exactly the same as replacing query bnodes with variables
 - in fact, Perez et al. (for that reason) don't consider Bnodes in triple patterns

Adapting [Perez et al. 2006]



ISSUE1: Algebra operations need to be adapted to multiset/bag semantics:

Definition 3:

Join:

$$M1 \bowtie M2 = \{ \mu1 \cup \mu2 \mid \mu1 \in M1, \mu2 \in M2, \text{ and } \mu1, \mu2 \text{ are compatible} \}$$

Union:

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

Minus:

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

LeftJoin:

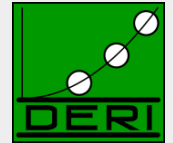
~~$$M1 \bowtie M2 = (M1 \bowtie M2) \cup (M1 \setminus M2)$$~~

Filter:

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

ISSUE2: non-compositionality of FILTERs in OPTIONAL

Semantics as per SPARQL1.0 spec:



$eval(BGP, G)$... see Definition 1

$eval(P1 \text{ } P2, G)$ = $eval(P1, G) \bowtie eval(P2, G)$

$eval(P1 \text{ UNION } P2, G)$ = $eval(P1, G) \cup eval(P2, G)$

$eval(P \text{ FILTER } R, G)$ = $eval(P, G) \upharpoonright_R$

$eval(P1 \text{ OPTIONAL } \{P2 \text{ FILTER } R\}, G)$ consists of all μ such that:

1. $\mu = \mu1 \cup \mu2$, such that $\mu1 \in eval(P1, G)$ and $\mu2 \in eval(P2, G)$ are compatible and $\mu(R) = \text{true}$, or
2. $\mu \in eval(P1, G)$ and there is no compatible $\mu2 \in eval(P2, G)$ for μ , or
3. $\mu \in eval(P1, G)$ and for any compatible $\mu2 \in eval(P2, G)$, $\mu \cup \mu2$ does not satisfy R .

■ 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



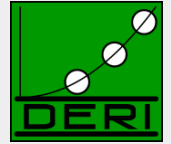
WG might still change some of the syntax/semantics definitions presented here based on community input

This is where SPARQL1.1 starts



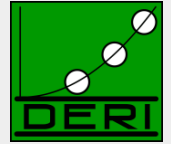
- **Missing common feature requirements in existing implementations or requested urgently by the community:**
 - **Assignment/Project Expressions**
 - **Aggregate functions (SUM, AVG, MIN, MAX, COUNT, ...)**
 - **Subqueries**
 - **Property paths**
 - complaint: SPARQL1.0 isn't quite a "graph" query language
- **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

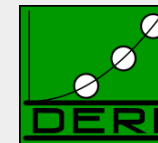
- Full Update language
- plus simple RESTful update methods for RDF graphs (HTTP methods)

■ Service Description

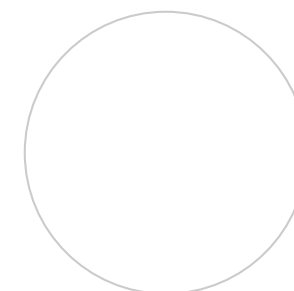
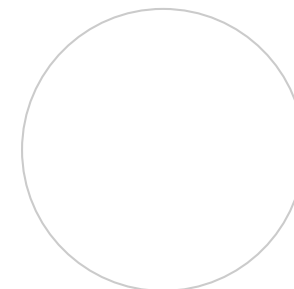
- Method for discovering a SPARQL endpoint's capabilities
- Summary of its data

We will focus on these in today's Tutorial

Part 1: new query features



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



■ 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:

Leave unbound!

?Item	?NewP
lemonade1	3.3
beer1	3.3
wine1	3.85
liqueur1	

■ Assignments, Creating new values...

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

Semantics:

$\text{extend}(\mu, \text{var}, \text{expr}) = \mu$ if var not in $\text{dom}(\mu)$ and $\text{eval}(\text{expr})$ is an error

$\text{extend}(\mu, \text{var}, \text{expr}) = \mu \cup \{ \text{var} \rightarrow \text{value} \mid \text{var}$ not in $\text{dom}(\mu)$ and $\text{value} = \text{eval}(\text{expr})$ is defined}

$\text{extend}(\mu, \text{var}, \text{expr})$ undefined if var in $\text{dom}(\mu)$

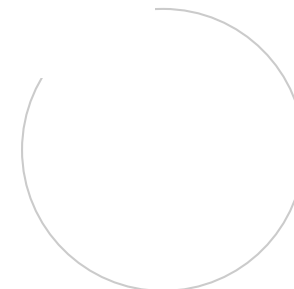
For sets of solutions:

$\text{extend}(M, \text{var}, \text{term}) = \{ \text{extend}(\mu, \text{var}, \text{term}) \mid \mu \text{ in } M \}$

■ Assignments, boolean values...

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

- In general: result type as of [Section 16.3 Operator Mapping in SPARQL1.1 Query spec.](#)



Data:

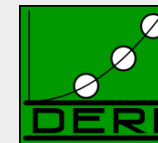
```
@prefix ex: <http://example.org/> .
```

```
ex:lemonade1      ex:price 3 .  
ex:beer1         ex:price 3.  
ex:wine1         ex:price 3.50 .  
ex:liqueur1     ex:price "n/a".
```

Results:

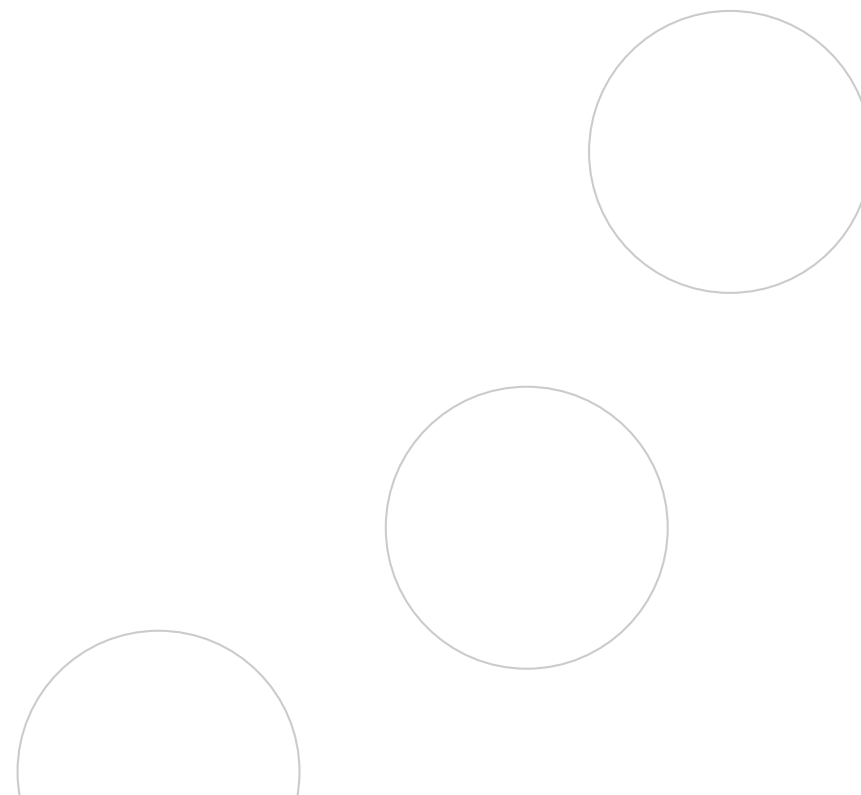
?Item	?NewP
lemonade1	"1"^^xsd:boolean
beer1	"1"^^xsd:boolean
wine1	"1"^^xsd:boolean
liqueur1	"0"^^xsd:boolean

Aggregates

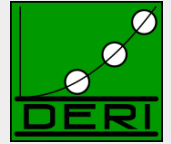


Digital Enterprise Research Institute

www.deri.ie



Aggregates



■ *“Count items”*

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

Data:

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

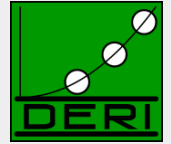
ex:lemonade1    ex:price 3 ;
                rdf:type ex:Softdrink.
ex:beer1       ex:price 3;
                rdf:type ex:Beer.
ex:wine1       ex:price 3.50 ;
                rdf:type ex:Wine.
ex:wine2       ex:price 4 .
                rdf:type ex:Wine.
ex:wine3       ex:price "n/a";
             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

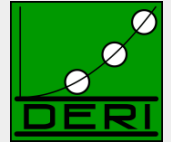
ex:lemonade1    ex:price 3 ;
                rdf:type ex:Softdrink.
ex:beer1       ex:price 3;
                rdf:type ex:Beer.
ex:wine1       ex:price 3.50 ;
                rdf:type ex:Wine.
ex:wine2       ex:price 4 .
                rdf:type ex:Wine.
ex:wine3       ex:price "n/a";
             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

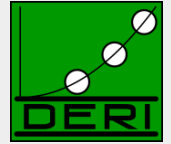
```
@prefix ex: <http://example.org/> .

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
```

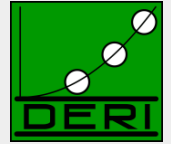
Dat

```
@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

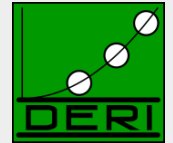


- **SUM** *... as usual*
 - **AVG** *... as usual*
 - **MIN** *... as usual*
 - **MAX** *... as usual*
 - **SAMPLE** *... “pick” one non-deterministically*
 - **GROUP_CONCAT** *... concatenate values with a designated separator string*
- ...this list is extensible* *... new built-ins will need to define error-behaviour, extra-parameters (like SEPARATOR in GROUP_CONCAT)*

Example SUM



*Under discussion:
WG might decide to simply ignore non-
numerics in Sum/Avg, but at the
moment just delegates to "+"*



■ “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:

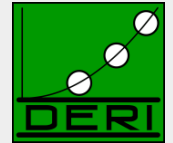
```
@prefix ex: <http://example.org/> .

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/> .

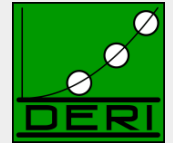
ex:alice a foaf:Person; foaf:name "Alice Wonderland";
        foaf:nick "Alice", "The real Alice".

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

Aggregates - Semantics



- Evaluate a list of (GROUP BY) expressions:

$\text{ListEval}(\text{ExprList}, \mu)$ returns a list E , where $E[i] = \mu(\text{ExprList}[i])$

- Use these to partition a solution sequence:

$\text{Group}(), \Omega = \{ 1 \rightarrow \Omega \}$

$\text{Group}(\text{ExprList}, \Omega) = \{ \text{ListEval}(\text{ExprList}, \mu) \rightarrow \{ \mu' \mid \mu' \text{ in } \Omega, \text{ListEval}(\text{ExprList}, \mu) = \text{ListEval}(\text{ExprList}, \mu') \} \mid \mu \text{ in } \Omega \}$

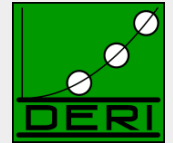
produces a *grouped solution sequence*

```
SELECT Sum(?y) AS ?Sy
WHERE { :s :p ?x; :q ?y }
GROUP BY ?x
```

Assume solution sequence $S = (\{?x \rightarrow 2, ?y \rightarrow 3\}, \{?x \rightarrow 2, ?y \rightarrow 5\}, \{?x \rightarrow 6, ?y \rightarrow 7\})$,

$\text{Group}(\{?x\}, S) = \{ (2) \rightarrow (\{?x \rightarrow 2, ?y \rightarrow 3\}, \{?x \rightarrow 2, ?y \rightarrow 5\}), (6) \rightarrow (\{?x \rightarrow 6, ?y \rightarrow 7\}) \}$

Aggregates - Semantics



Definition: Aggregation (*simplified*)

Aggregation applies set function “func” (e.g. sum, min, max, ...) to a **multiset of lists of expressions** and a **grouped solution sequence**, G as produced by the Group function. It produces a single value for each key and partition for that key (key, X).

$$\text{Aggregation}(\text{ExprList}, \text{func}, G) = \{ \text{dom}(g) \rightarrow F \mid g \text{ in } G \}$$

where $M = \text{ListEvalE}(\text{ExprList}, \text{range}(g))$

$$F = \text{func}(M), \text{ for non-DISTINCT}$$

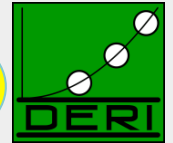
$$F = \text{func}(\text{Distinct}(M)), \text{ for DISTINCT}$$

$$G = \left\{ \begin{array}{l} (2) \rightarrow (\{?x \rightarrow 2, ?y \rightarrow 3\}, \{?x \rightarrow 2, ?y \rightarrow 3\}), \\ (6) \rightarrow (\{?x \rightarrow 6, ?y \rightarrow 7\}) \end{array} \right\}$$

$$\begin{aligned} \text{Aggregation}(?y, \text{Sum}, G) &= \{ (2) \rightarrow \text{Sum}((3), (3)), (6) \rightarrow \text{Sum}((7)) \} \\ &= \{ (2) \rightarrow 6, (6) \rightarrow 7 \} \end{aligned}$$

Aggregates - Semantics

Omitted details on error handling and scalar Parameters like "SEPERATOR" in GROUP_CPNOCAT



Definition: Aggregation (*simplified*)

Aggregation applies set function "func" (e.g. sum, min, max, ...) to a **multiset of lists of expressions** and a **grouped solution sequence**, G as produced by the Group function. It produces a single value for each key and partition for that key (key, X).

$\text{Aggregation}(\text{ExprList}, \text{func}, G) = \{ \text{dom}(g) \rightarrow F \mid g \text{ in } G \}$

where $M = \text{ListEvalE}(\text{ExprList}, \text{range}(g))$

$F = \text{func}(M)$, for non-DISTINCT

$F = \text{func}(\text{Distinct}(M))$, for DISTINCT

$G = \{ (2) \rightarrow (\{ ?x \rightarrow 2, ?y \rightarrow 3 \}, \{ ?x \rightarrow 2, ?y \rightarrow 5 \}), (6) \rightarrow (\{ ?x \rightarrow 6, ?y \rightarrow 7 \}) \}$

$\text{Aggregation}(?y, \text{Sum}, G) = \{ (2) \rightarrow \text{Sum}((3), (5)) , (6) \rightarrow \text{Sum}((7)) \}$
 $= \{ (2) \rightarrow 8 , (6) \rightarrow 7 \}$

Aggregations subsequently mapped back via **Extend(...)** to solution multisets



SELECT Sum(?y) AS ?Sy

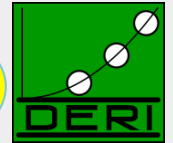
WHERE { :s :p ?x; :q ?y }

GROUP BY ?x

{ { ?x → 2 ?Sy → 8 }, { ?x → 6, ?Sy → 7 } }

Aggregates - Semantics

Omitted details on error handling and scalar Parameters like "SEPERATOR" in GROUP_CPNOCAT



Definition: Aggregation (*simplified*)

Aggregation applies set function "func" (e.g. sum, min, max, ...) to a **multiset of lists of expressions** and a **grouped solution sequence**, G as produced by the Group function. It produces a single value for each key and partition for that key (key, X).

$\text{Aggregation}(\text{ExprList}, \text{func}, G) = \{ \text{dom}(g) \rightarrow F \mid g \text{ in } G \}$

where $M = \text{ListEvalE}(\text{ExprList}, \text{range}(g))$

$F = \text{func}(M)$, for non-DISTINCT

$F = \text{func}(\text{Distinct}(M))$, for DISTINCT

$G = \{ (2) \rightarrow (\{ ?x \rightarrow 2, ?y \rightarrow 3 \}, \{ ?x \rightarrow 2, ?y \rightarrow 5 \}), (6) \rightarrow (\{ ?x \rightarrow 6, ?y \rightarrow 7 \}) \}$

$\text{Aggregation}(?y, \text{Sum}, G) = \{ (2) \rightarrow \text{Sum}((3), (5)) , (6) \rightarrow \text{Sum}((7)) \}$
 $= \{ (2) \rightarrow 8 , (6) \rightarrow 7 \}$

Aggregations subsequently mapped back via **Extend(...)** to solution multisets



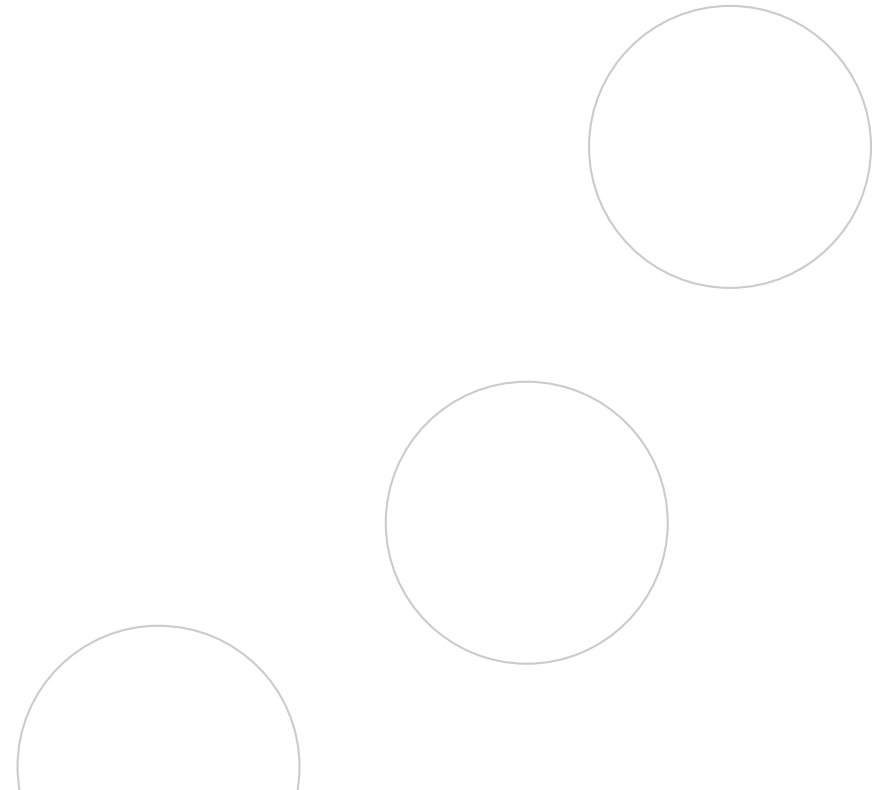
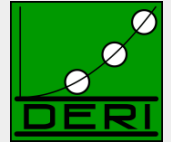
SELECT Sum(?y) AS ?Sy

WHERE { :s :p ?x; :q ?y }

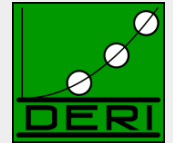
GROUP BY ?x

{ { ?x → 2 ?Sy → 8 }, { ?x → 6, ?Sy → 7 } }

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#>
```

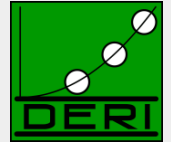
```
CONSTRUCT{ ?P foaf:name ?FullName }
```

```
WHERE {
```

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

```
}
```

Subqueries “Limit per resource”

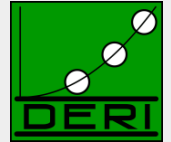


- Give me **all** titles of papers of **10 persons** who co-authored with Tim Berners-Lee

```
SELECT ?T
WHERE {
  ?D foaf:maker ?P ; rdfs:label ?T .
  {
    SELECT DISTINCT ?P
    WHERE { ?D foaf:maker <http://dblp.13s.de/.../authors/Tim_Berners-Lee>, ?P .
           FILTER ( ?P != <http://dblp.13s.de/.../authors/Tim_Berners-Lee> )
           }
    LIMIT 10
  }
}
```

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

Subqueries - Semantics



- *Note: Before Solution Modifiers are applied, SPARQL semantics converts solution multisets to solution sequences*

```
SELECT ?T
WHERE {
  ?D foaf:maker ?P ; rdfs:label ?T .
}
```

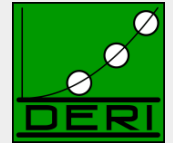
```
SELECT DISTINCT ?P
WHERE { ?D foaf:maker <http://dblp.../Tim_Berners-Lee>, ?P .
        FILTER ( ?P != <http://dblp.../Tim_Berners-Lee> )
}
ORDER BY ?P
LIMIT 10
```

Annotations for the subquery:

- eval(P,G)**: Points to the WHERE clause.
- ToList(M)**: Points to the subquery block.
- OrderBy(Ω , cond)**: Points to the ORDER BY clause.
- Slice(Ω , start, length)**: Points to the LIMIT clause.
- ToMultiSet(Ω)**: Points to the entire subquery block.

- *Subqueries require one additional algebra operator, **toMultiset**, which takes Sequences and returns Multisets*

Subqueries – Known Limitations



- Attention: Subqueries are compositional, 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

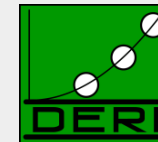
```
:jim a foaf:Person .
:tim a foaf:Person .

: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!

Subqueries – Known Limitations



- Attention: Subqueries are compositional, 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 ?P
    WHERE {
      ?D foaf:maker ?P ; dc:title ?T
    }
    LIMIT 3
  }
}
```

Doesn't work either... this is “only” a normal join, doesn't inject binding inside the query

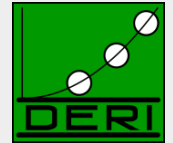
```
:jim a foaf:Person .
:tim a foaf:Person .

: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”
:tim	“Doc1”
:tim	“Doc2”

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

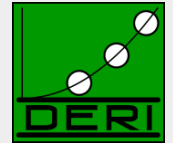
FROM in Subqueries? NO!



- 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.13s.de/.../authors/Tim_Berners-Lee>
    { [ foaf:maker <http://dblp.13s.de/.../authors/Tim_Berners-Lee>,
      [ foaf:name ?N ] ] . } }
}
```

(NOT) EXISTS: Another form of Subqueries



- **Note Subqueries in EXIST are evaluated AFTER substitution!**

Graph:

```
:a a foaf:Person ;  
  foaf:homepage <http://a1> ;  
  foaf:homepage <http://a2> .  
:b a foaf:Person .
```

?X

:a

Query:

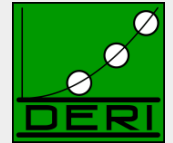
```
SELECT ?X  
WHERE{ ?X rdf:type foaf:Person  
       EXISTS { ?X foaf:homepage ?H } ) }
```

Recall, Filter semantics:
Substitute R and then check
whether ASK query
evaluates to true

Filter:

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

(NOT) EXISTS: Another form of Subqueries



- **Note Subqueries in EXIST are evaluated AFTER substitution!**

Graph:

```
:a a foaf:Person ;
   foaf:homepage <http://a1> ;
   foaf:homepage <http://a2> .
:b a foaf:Person .
```

Query:

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

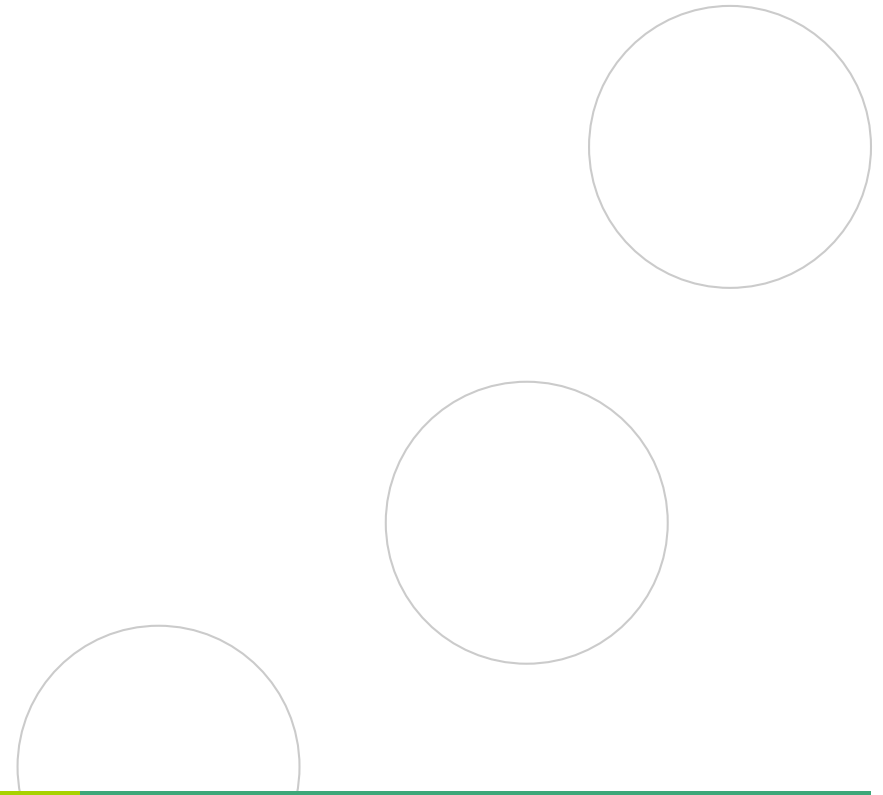
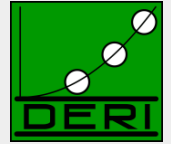
?X

:a

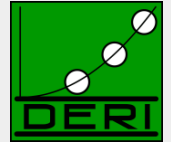
:a

As opposed to EXISTS,
joins can create duplicates

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 } ) }
```

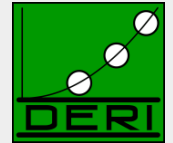
- *SPARQL1.1 has two alternatives to do the same*

- *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*

- *Semantics*



MINUS and NOT EXISTS



- May have different results, e.g.:

```
PREFIX ex: <http://example.org/>
```

```
SELECT *
```

```
WHERE{ ?S ?P ?O
```

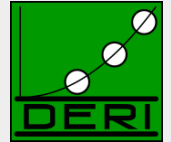
```
MINUS { ex:a ex:b ex:c } }
```

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

```
ex:a ex:b ex:c
```

?S	?P	?O
a	b	c

Current Semantics of MINUS



Definition 3:

Join:

$$M1 \bowtie M2 = \{ \mu1 \cup \mu2 \mid \mu1 \in M1, \mu2 \in M2, \text{ and } \mu1, \mu2 \text{ are compatible} \}$$

Union:

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

Minus:

$$M1 \setminus M2 = \{ \mu \in M1 \mid \text{forall } \mu' \in M2, \mu \text{ and } \mu' \text{ are not compatible} \\ \text{or } \text{dom}(\mu) \text{ and } \text{dom}(\mu') \text{ are disjoint} \}$$

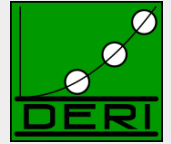
LeftJoin:

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

Filter:

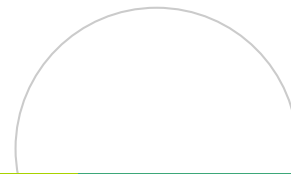
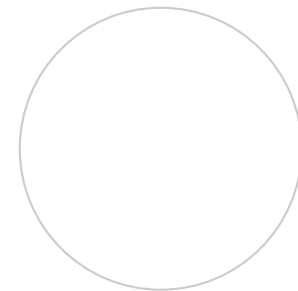
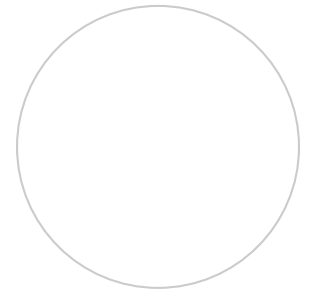
$$M|_R = \{ \mu \mid \mu \in M \text{ and } \mu(R) = \text{true} \}$$

Property Path Expressions



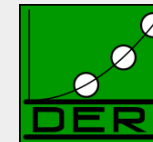
Digital Enterprise Research Institute

www.deri.ie



Enabling **networked** knowledge.

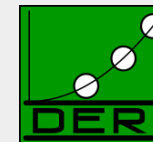
Property Path expressions



- 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



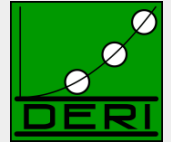
■ elt ... Path Element

Syntax Form	Matches
<i>uri</i>	A URI or a prefixed name. A path of length one.
<i>^elt</i>	Inverse path (object to subject).
<i>!uri</i> or <i>!(uri₁ ... uri_n)</i>	Negated property set. A URI which is not one of <i>uri_i</i>
<i>!^uri</i> and <i>!(uri₁ ... uri_j ^uri_{j+1} ... ^uri_n)</i>	Negated property set. A URI which is not one of <i>uri_i</i> , nor <i>uri_{j+1}...^uri_n</i> as reverse paths
<i>(elt)</i>	A group path <i>elt</i> , brackets control precedence.
<i>elt1 / elt2</i>	A sequence path of <i>elt1</i> , followed by <i>elt2</i>
<i>elt1 elt2</i>	A alternative path of <i>elt1</i> , or <i>elt2</i> (all possibilities are tried).
<i>elt*</i>	A path of zero or more occurrences of <i>elt</i> .
<i>elt+</i>	A path of one or more occurrences of <i>elt</i> .
<i>elt?</i>	A path of zero or one <i>elt</i> .
<i>elt{n,m}</i>	A path between <i>n</i> and <i>m</i> occurrences of <i>elt</i> .
<i>elt{n}</i>	Exactly <i>n</i> occurrences of <i>elt</i> .
<i>elt{n,}</i>	<i>n</i> or more occurrences of <i>elt</i> .
<i>elt{,n}</i>	Between 0 and <i>n</i> occurrences of <i>elt</i> .

■ Semantics: by translation to native SPARQL with two core property paths Operators:

- ArbitraryPath(X, path, Y)
- ZeroLengthPath(X, path, Y)

Path expressions

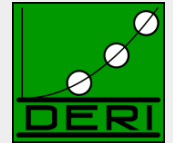


- Can be used for some ontological inference (well known since [Perez et al. 2008])
- E.g. Find all Beers in the Beer ontology

```
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:



Digital Enterprise Research Institute

www.deri.ie

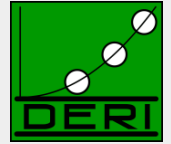
Some current (partial) SPARQL1.1 implementations:

- **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.**

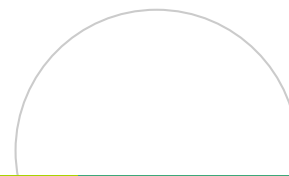
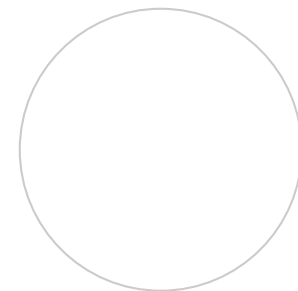
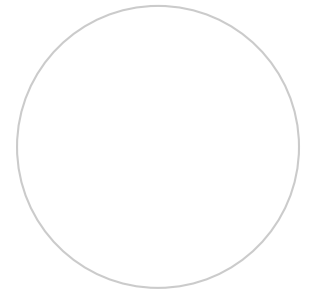
Others probably forthcoming...

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

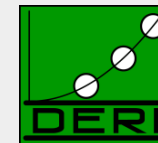
Part 2: Entailment Regimes



SPARQL 1.1 querying over OWL2 ontologies and RIF rulesets?



SPARQL1.1 Entailment Regimes



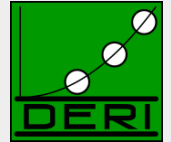
- SPARQL1.1 will define SPARQL query answering over OWL2 ontologies and RIF rule sets:

- <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

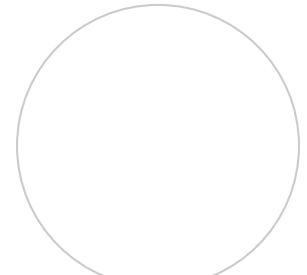
– Won't go into details of those, but sketch the main ideas!

RDFS/OWL2 and SPARQL1.1



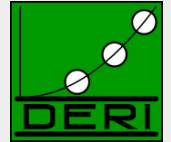
- General Idea: Answer Queries with implicit answers
- E.g. example from before:

```
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>

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

■ Challenges+Pitfalls:

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

■ 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 entirely intuitive, so
 - Good to know ;-)

■ 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 .  
...
```

■ Graph:

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

Query with RDFS Entailment in mind:

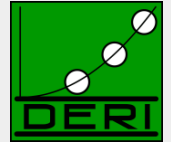
```
SELECT ?CM { ?CM a rdfs:ContainerMembershipProperty }
```

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, }
```

More on SPARQL 1.1 + RDFS



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

- ATTENTION: this also means no “surrogate blank nodes”!
- Graph

```
:alice :name "Alice" .
```

- Note: the informative RDFS inference rules at <http://www.w3.org/TR/rdf-mt/#rules> contains the following rules:

```
rdfs1: { _:L rdf:type rdfs:Literal } :- { ?S ?P ?L . } ^ literal(?L)  
where _:L is a blank node allocated to the literal bound to ?L
```

- BUT: the following query

```
SELECT ?L WHERE { ?L rdf:type rdfs:Literal }
```

has no answers by above restriction!

Stupid way to say Peter is 50:

```
ex:Peter a [ a owl:Restriction ;  
             owl:onProperty ex:age ;  
             owl:allValuesFrom [ rdf:type rdfs:Datatype ;  
             owl:oneOf ("50"^^xsd:integer) ] ] .
```

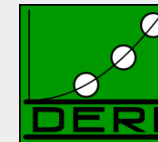
Stupid query asking What is NOT Peters age:

```
SELECT ?x WHERE {  
    ex:Peter a [ a owl:Restriction ; owl:onProperty ex:age ;  
                owl:allValuesFrom [ a rdfs:Datatype ;  
                                     owl:datatypeComplementOf [ a  
                                     rdfs:Datatype ; owl:oneOf (?x) ] ] ] }
```

Theoretical answer: all literal different from 50

No danger in SPARQL 1.1 restricts answers to `rdf:/rdfs:/owl:vocabulary` minus `rdf:_1 ... rdf:_n` plus terms occurring in the data graph

Non-distinguished variables:



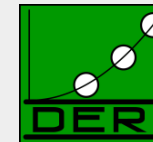
■ E.g. Graph

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

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

Non-distinguished variables:



■ E.g. Graph

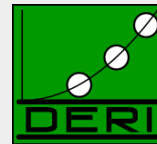
```
foaf:Person rdfs:subClassOf foaf:Agent .
foaf:Person rdfs:subClassOf
  [ a owl:Restriction ;
    owl:onProperty :hasFather ;
    owl:someValuesFrom foaf:Person ] .
foaf:knows rdfs:range foaf:Person.
:jeff a Person .
:jeff foaf:knows :aidan .
```

```
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.

Non-distinguished variables:



- **Simple Solution may seem not always intuitive, but:**
 - OWL Entailment in SPARQL based on BGP matching, i.e.
 - always only returns results with named individuals
 - Doesn't affect SELECT: takes place before projection
 - That is: **non-distinguished variables can't occur "by design"**
 - In fact, conjunctive queries with non-distinguished variable still an open research problem for OWL:
 - Decidable for SHIQ, [B. Glimm et al. 2008]
 - Decidable for OWL1 DL without transitive properties OWL1 Lite without nominals [B. Glimm, KR-10]
 - Decidability for SHOIN, SROIQ though still unknown...

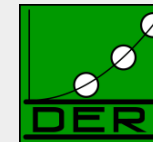
- Once again: SPARQL entailment defined only at the level of **BGP** matching
- SPARQL1.1 Algebra is layered “on top”, no interaction

```
:person1 rdf:type [ owl:unionOf (:male :female) ] .
```

```
SELECT ?X { {?X rdf:type :male }  
UNION  
{?X rdf:type :female }  
}
```

→ No result!

SPARQL 1.1 other features: Aggregates



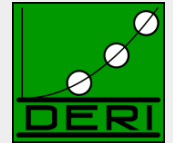
- Similar as before... aggregates are evaluated within algebra **after** BGP matching, so, no effect:

```
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 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 }
```

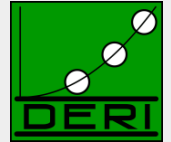
Under RDFS/OWL entailment returns :{?Y/3}

■ 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)

■ 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 }
```

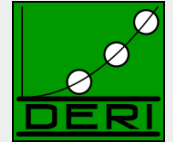
```
owl4: { ?S ?P ?Y } :- { ?X owl:SameAs ?Y . ?S ?P ?X }
```

■ 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.

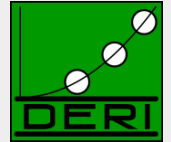
In current draft called
`rif:imports`

```
<http://ruleset1.rif> rif:usedWithProfile
  <http://www.w3.org/ns/entailment/Simple> .
<http://dblp.13s.../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

SPARQL1.1 + RIF Core Challenges



■ Semantics:

- Relatively Straightforward: **BGPs matching** defined as being RIF-RDF-entailed by RDF data graph in combination with the referenced ruleset.

■ Infinite answers possible

- (even though RIF Core has no function symbols):

```
:a :b 1 .  
{ ?S ?P (?O + 1) } :- { ?S ?P ?O } .
```

```
SELECT ?O { :a :b ?O . }
```

■ So far (as opposed to SPARQL/OWL SPARQL/RDFS) no restrictions on finiteness in SPARQL1.1/RIF

- Finite answers up to the user, or
- Restrict to strongly safe RIF Core (inspired by [Eiter et al. 2006] or
- System streams out answers (e.g. a la Prolog)

■ SPARQL 1.0

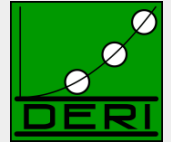
- UNIONS of Conjunctive Queries, FILTERs, GRAPH queries, OPTIONAL, (hidden) negation
- contributed largely to the current Linked Data boom
- Inspired interesting academic work

■ 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

■ Functions Library in SPARQL1.0 is insufficient:

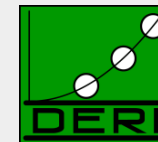
- Bound(.)
- isLiteral(.)
- isBlank(.)
- isIRI(.)
- Str(.)
- Regex(. , .)
- +, -, *, <, >, =

■ New functions to be included in standard library:

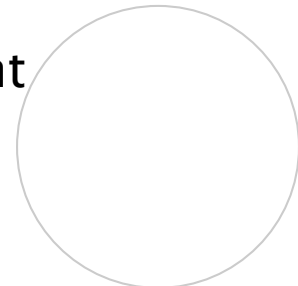
- COALESCE, IF
- Functions from the Xpath/Xquery function library
 - String manipulation, more math, etc. ... e.g. fn:concat

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 }
```

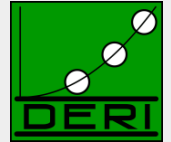
```
UNION
```

DBLP SPARQL
endpoint

```
{ [ foaf:maker <http://dblp.13s.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
```

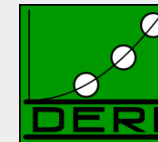
```
  UNION
```

```
  { SERVICE <http://dblp.13s.de/d2r/sparql>
```

```
    { [ foaf:maker <http://dblp.13s.de/.../authors/Tim_Berners-Lee>,
      [ foaf:name ?N ] ] . } }
```

```
}
```


Basic federated Queries (*time permitting*)



■ ISSUES:

- treatment of variables in SERVICE

```
SERVICE ?v { P }
```

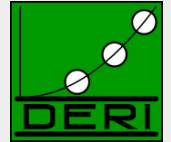
- Alternatives for semantics:

- “FILTER-Style”: evaluate SERVICE patterns last within a group, but probably needs to forbid interactions between several SERVICE PATTERNS within the same group:

E.g. { SERVICE ?X { :S :P ?Y } SERVICE ?Y { :S :P ?X } }

- OPTIONAL Style: evaluate bindings in place (non-compositional)
- GRAPH style: i.e. define an extension of RDF dataset including a “service set”

Basic Federated Queries - BINDINGS



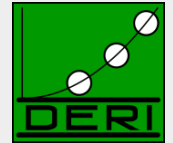
- Sometimes you want to “inject” or “fix” some bindings into the query to be sent to an external endpoint.
- Goal: reduce data to be transferred:
- Example:

```
... WHERE { ?s :p2 ?v2 } BINDINGS ?s ?v1 { ( <s1> 7 ) ( <s2> UNBOUND ) }
```

```
Join( [ { ?s⇒<s1> ?v1⇒7 }  
        { ?s⇒<s2> } ],  
      BGP (?s :p2 ?v2) )
```

- Note: does NOT really allow to inject variable bindings, only constants! → remains compositional

Basic Federated Queries - BINDINGS



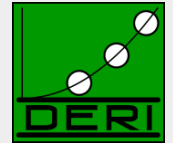
- Sometimes you want to “inject” or “fix” some bindings into the query to be sent to an external endpoint.
- Goal: reduce data to be transferred:
- Example:

```
... 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 {} } }
```

→ doesn't add expressivity...

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

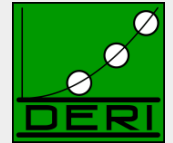
- INSERT DATA
 - DELETE DATA
 - DELETE/INSERT
 - DELETE
 - INSERT
 - DELETE WHERE
 - LOAD
 - CLEAR

- 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](#)

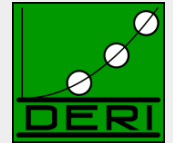
3.3 Instances

- 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](#)

Please comment!!!

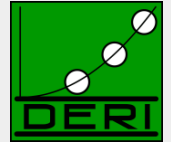


Chair-hat on: *Pleeeeeeease*

(1) Read the specs!

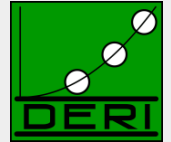
(2) Send us comments public-rdf-dawg-comments@w3.org

References



- [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. ICOT2010

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/>

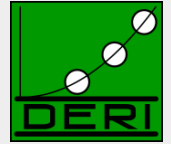
OWL Entailment:

- ❑ 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:

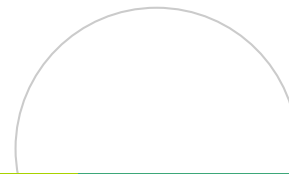
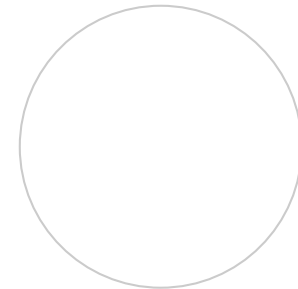
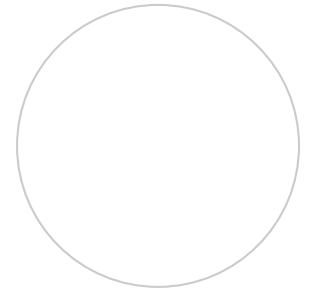
- ❑ RIF Core Dialect <http://www.w3.org/TR/rif-core/>
- ❑ RIF Basic Logic Dialect <http://www.w3.org/TR/rif-bld/>
- ❑ RIF RDF and OWL compatibility <http://www.w3.org/TR/rif-rdf-owl/>

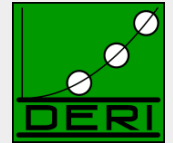
Acknowledgements



- The members of the W3C SPARQL WG, particularly Lee Feigenbaum, who I stole some examples from
- The members of the W3C RIF WG

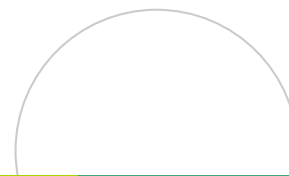
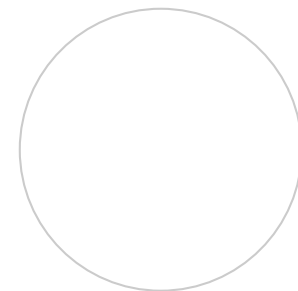
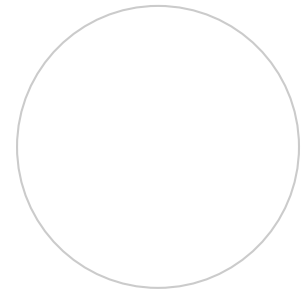
- The RR2010 chairs for inviting me 😊





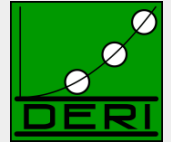
GiaBATA

Implementing SPARQL, OWL2RL, RIF on top of DLV



- A system which implements dynamic SPARQL querying, under different entailment regimes and how it can be implemented.
- Based on LP engine DLV
 - Datalog with built-ins (covers roughly RIF Core),
 - persistent Database backend (DLV-DB)
 - Optimisations (rewriting to push join processing into SQL as far as possible, magic sets,...)
 - plus a lot more features (nonmonotonicity, aggregates, ...)
- Overall idea for SPARQL+RDFS/OWL2RL over RDF graphs:
 - Translate OWL2RL to Datalog rules a la RIF, see above.
 - Translate SPARQL query to Datalog [Polleres, 2007]
 - Feed resulting program into a rules engine (DLV-DB) that runs over a Rel DB storing RDF graphs.
- Check Details at [Ianni et al. 2009]:
http://axel.deri.ie/~axepol/presentations/20091029ianni-etal-ISWC2009_GiaBATA.pptx

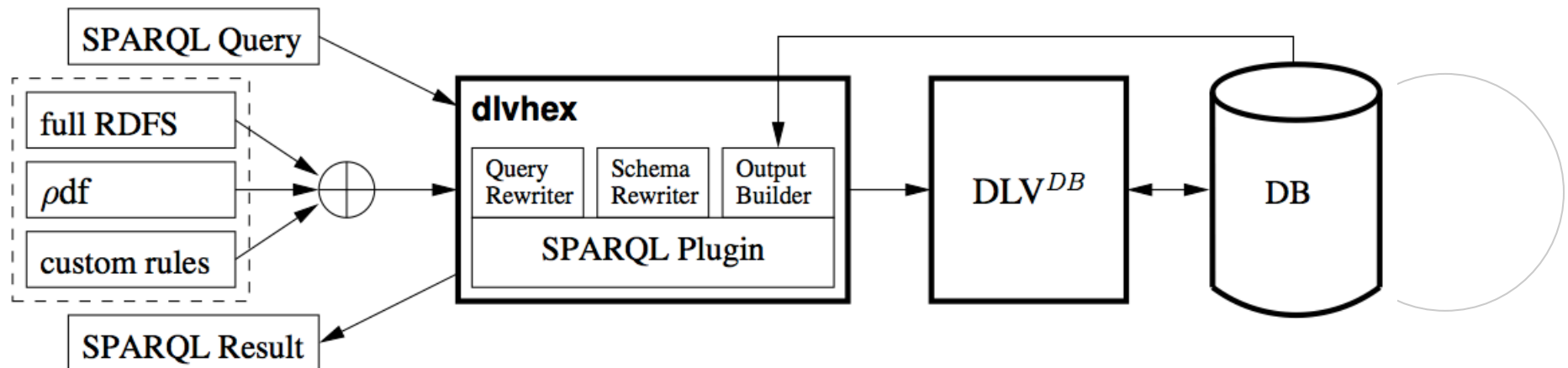
How to implement this?



■ GiaBATA system [Ianni et al., 2009]:

- SPARQL → dlhex (logic program)
- Ruleset → dlhex (logic program)

→ SQL



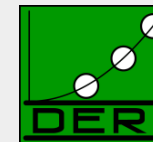
■ Deductive Database techniques:

- Datalog engine (dlhex)
- Postgres SQL Database underneath (dlv-db)
- RDF storable in different schemas in RDB
- Magic sets, storage

- Based on [Polleres 2007]

```
select * from <http://alice.org/>
where { ?X a foaf:Person. ?X foaf:name ?N.
       filter ( ?N != "Alice") optional { ?X foaf:mbox ?M } }
```

```
(r1) "triple"(S,P,0,default) :- &rdf[ "alice.org" ](S,P,0).
(r2) answer1(X_N,X_X,default) :- "triple"(X_X,"rdf:type","foaf:Person",default),
                                "triple"(X_X,"foaf:name",X_N,default),
                                &eval[" ?N != 'Alice' ", "N", X_N ](true).
(r3) answer2(X_M,X_X,default) :- "triple"(X_X,"foaf:mbox",X_M,default).
(r4) answer_b_join_1(X_M,X_N,X_X,default) :- answer1(X_N,X_X,default),
                                             answer2(X_M,X_X,default).
(r5) answer_b_join_1(null,X_N,X_X,default) :- answer1(X_N,X_X,default),
                                              not answer2_prime(X_X,default).
(r6) answer2_prime(X_X,default) :- answer1(X_N,X_X,default),
                                   answer2(X_M,X_X,default).
(r7) answer(X_M,X_N,X_X) :- answer_b_join_1(X_M,X_N,X_X,default).
```



- Straightforward, just translates rules in a way “compatible” with the SPARQL translation:

```
{?s ?q ?o } <= {?s ?p ?o . ?p rdfs:subPropertyOf ?q}
```

%FROM CLAUSES

```
triple(P, SubPropertyOf, P, G) :- triple(P, Type, Property, G), graph(G, D), data(D), defaultGraph(D),  
resource_literal(Type, "<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>", _),  
resource_literal(Property, "<http://www.w3.org/1999/02/22-rdf-syntax-ns#Property>", _),  
resource_literal(SubPropertyOf, "<http://www.w3.org/2000/01/rdf-schema#subPropertyOf>", _).
```

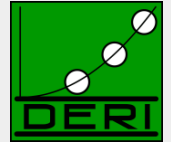
%FROM NAMED CLAUSES

```
triple(P, SubPropertyOf, P, G) :- triple(P, Type, Property, G), graph(G, D), data(D), namedGraph(D),  
resource_literal(Type, "<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>", G),  
resource_literal(Property, "<http://www.w3.org/1999/02/22-rdf-syntax-ns#Property>", G),  
resource_literal(SubPropertyOf, "<http://www.w3.org/2000/01/rdf-schema#subPropertyOf>", G).
```

%USING ONTOLOGIES

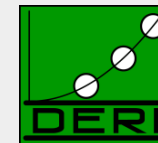
```
triple(P, SubPropertyOf, P, G) :- triple(P, Type, Property, G), graph(G, D), data(D), ontology(D),  
resource_literal(Type, "<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>", G),  
resource_literal(Property, "<http://www.w3.org/1999/02/22-rdf-syntax-ns#Property>", G),  
resource_literal(SubPropertyOf, "<http://www.w3.org/2000/01/rdf-schema#subPropertyOf>", G).
```

SPARQL+Rules → SQL



- Done by dlv-DB, cf. [Terracina, et al.,2008]
 - All non-recursive parts are pushed to the Database
 - All recursive parts handled by semi-naïve evaluation
(more efficient than WITH RECURSIVE views in SQL, where necessary, intermediate results temporarily materialized into the DB)

- Some necessary optimisations to make this *reasonably* performant:
 - FILTER expression evaluation is pushed to SQL (3-valued semantics of SPARQL Filters is handled natively in SQL)
 - No miracles... but magic: Magic set optimisations for focused fwd-chaining evaluation.
 - Join-reordering based on statistics/selectivity e.g. a la [Vidal et al. 2009], not yet implemented, but we did some manual reordering to optimize the query plan in the experiments.

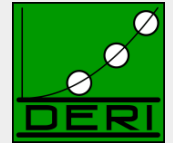


- OWL2RL and RIF Entailment, done, but doesn't yet consume RIF directly:
 - Integration with RIF-plugin [Obermeier et al. RR2010] planned

SPARQL1.1 features:

- Subqueries , doable (however: modulo solution modifiers)
- NOT EXISTS, MINUS, doable
- Property Path Expressions, doable
- Aggregates, probably doable, cf. [Polleres et al. 2007], [Faber et al. 2004]

Additional References



- [Ianni et al., 2009] G. Ianni, T. Krennwallner, A. Martello, A. Polleres. Dynamic querying of mass-storage RDF data with rule-based entailment regimes. ISWC2009
- [Obermeier et al. 2010] Philipp Obermeier, Marco Marano and Axel Polleres - Processing RIF and OWL2RL within DLVHEX. RR2010 – System Demo.
- [Terracina, et al.,2008] Terracina, G., Leone, N., Lio, V., Panetta, C.: Experimenting with recursive queries in database and logic programming systems. Theory Pract. Log. Program. 8(2) (2008) 129–165.
- [Vidal et al. 2010] M.-E. Vidal, E. Ruckhaus, T. Lampo, A. Marínez, J. Sierra, A. Polleres. On the efficiency of joining group patterns in SPARQL queries. ESWC2010

Additional Acknowledgements:

Giovambattista Ianni, Alessandra Martello, Thomas Krennwallner, Philipp Obermeier