SIEMENS

Unit 5: SPARQL1.1 …

new! W3C Recommendation since 21 March 2013

This is where SPARQL1.1 started (2009)

Various SPARQL1.0 implementations out there, various extensions.

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)**
  - **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.

New in SPARQL1.1

SPARQL1.1 Query Language

- **Project Expressions**
- **Aggregate functions**
- **Subqueries**
- **Negation**
- **Property Paths**
- **Extend the function library**

We will focus on these in this lecture

SPARQL 1.1 Federated Query

- Basic federated Queries over different SPARQL endpoints

SPARQL 1.1 Entailment

- *RDF(S), OWL, RIF*

SPARQL 1.1 Update

- Full Update language

SPARQL 1.1 Graph Store HTTP Protocol

- simple RESTful update method to modify RDF graphs (HTTP methods)

SPARQL 1.1 Service Description

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

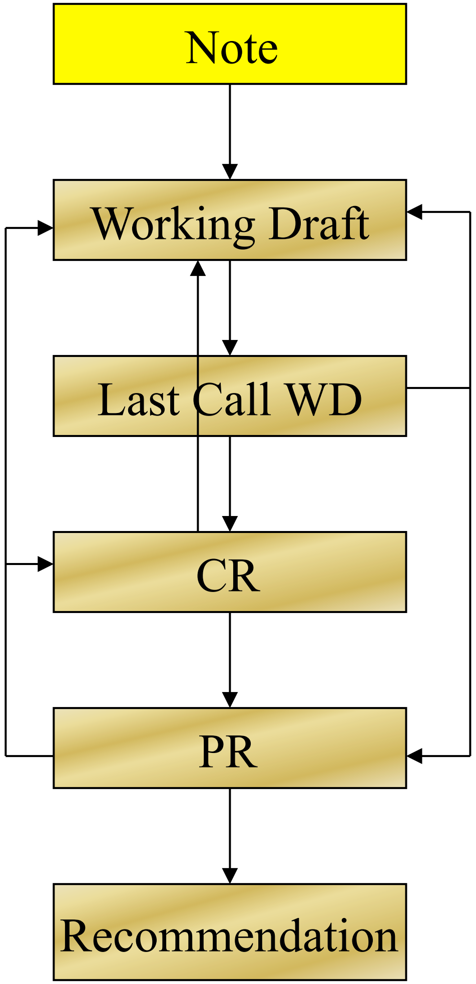
Plus several results formats (JSON, CSV/TSV, XML) and refurbished SPARQL Protocol

Where is SPARQL 1.1 in terms of becoming a standard?

SIEMENS

Standardization process: Six types of documents

- **Note**
  - Not a component in the standardization process
  - No declaration that W3C stands behind
- **Working Draft (WD)**
  - Documentation of a discussion condition
- **Last Call WD**
  - When the goals are reached
- **Candidate Recommendation (CR)**
  - Confirmation of success
- **Proposed Recommendation**
  - Extension; partial implementation
- **Recommendation**
  - official W3C standard



New query language features

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

Project Expressions

Assignments, Creating new values…

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

~~SELECT ?Item **?NewP**~~

~~WHERE { ?Item ex:price ?Pr FILTER (?NewP = ?Pr * 1.1) }~~

Data:

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

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

Results:

?Item	?NewP

Project Expressions

Assignments, Creating new values…

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

Data:

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

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

Results:

?Item	?NewP
lemonade	3.3
beer	3.3
wine	3.85

Project Expressions

Assignments, Creating new values…

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

Data:

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

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

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

Results:

?Item	?NewP
lemonade	3.3
beer	3.3
wine	3.85

Ignore entire row in result?

Project Expressions

Assignments, Creating new values…

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

Data:

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

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

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

Results:

SPARQL 1.1: Leaves “errors” unbound!

?Item	?NewP
lemonade	3.3
beer	3.3
wine	3.85
liqueur1	

Alternative to Project Expressions – BIND:

Same meaning, different syntax **BIND**…

Note: BIND is evaluated in-place

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

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
lemonade	3.3
beer	3.3
wine	3.85
liqueur1	

Alternative to Project Expressions – BIND:

Same meaning, different syntax **BIND**…

*Note: BIND is evaluated **in-place, cf.*** http://www.w3.org/TR/sparql11-query/#bind

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

?Item	?NewP
lemonade	
beer	
wine	
liqueur1	

SIEMENS

Project expressions - Semantics

Assignments, Creating new values…

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

Semantics:

extend(*μ*, var, expr) = *μ* if var not in *dom(μ)* and eval(expr) is an error

extend(μ, var, expr) = *μ* ∪ { var → value | var not in *dom(μ)* and value = eval(expr) is defined}

extend(*μ*, var, expr) undefined if var in *dom(μ)*

For sets of solutions:

extend(M , var, term) = {{ extend(*μ*, var, term) | *μ* in M }}

Project expressions - Semantics

Assignments, Creating new values…

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

Semantics:

extend(*μ*, var, expr) = *μ* if var not in *dom(μ)* and eval(expr) is an error

extend(μ, var, expr) = *μ* ∪ { var → value | var not in *dom(μ)* and value = eval(expr) is defined}

extend(*μ*, var, expr) undefined if var in *dom(μ)* *i.e., this is syntactically disallowed.*

For sets of solutions:

extend(M , var, term) = {{ extend(μ, var, term) | μ in M }}

Aggregates

Aggregates

“Count items”

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

Data:

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

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

Results:

?C
5

Aggregates

“Count categories”

```
PREFIX ex: <http://example.org/>
SELECT (Count(?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
5

Aggregates

"Count categories"

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

Data:

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

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

Results:

?C
3

Aggregates - Grouping

“Count items per categories”

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

Data:

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

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

Results:

?T	?C
Softdrink	1
Beer	1
Wine	3

Aggregates – Filtering Groups

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

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

Data:

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

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

Results:

?T	?C
Wine	3

Other Aggregates

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

Example SUM

Note:

Important to know that Sum/Avg, just delegates to numeric operations (sum uses "+", etc., so errors, unbounds, non-numerics need special handling!

"Sum Prices per categories"

```
PREFIX ex: <http://example.org/>
SELECT ?T (Sum(?Pr) 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	

Example SUM

Note:

Important to know that Sum/Avg, just delegates to numeric operations (sum uses "+", etc., so errors, unbounds, non-numerics need special handling!

"Sum Prices per categories"

```
PREFIX ex: <http://example.org/>
SELECT ?T (Sum(?Pr) AS ?P)
WHERE { ?Item rdf:type ?T; ex:price ?Pr
        FILTER( isNumeric(?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 SUM

Note:

Important to know that Sum/Avg, just delegates to numeric operations (sum uses “+”, etc., so errors, unbounds, non-numerics need special handling!

“Sum Prices per categories”

```
PREFIX ex: <http://example.org/>
SELECT ?T (Sum(COALESCE(xs:decimal(?Pr),0) AS ?C)
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 SUM

Note:

Important to know that Sum/Avg, just delegates to numeric operations (sum uses "+", etc., so errors, unbounds, non-numerics need special handling!

"Sum Prices per categories"

```
PREFIX ex: <http://example.org/>
SELECT ?T (Sum(IF(isNumeric(?Pr),?Pr,0) AS ?P) 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 "

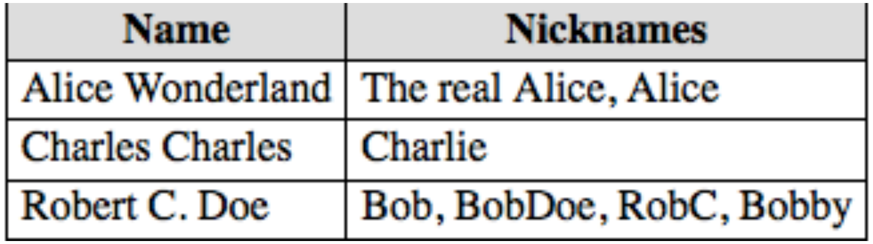
```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ( SAMPLE(?N) as ?Name)
       ( GROUP_CONCAT(?M; SEPARATOR = ", ") AS ?
Nicknames )
WHERE { ?P a foaf:Person ;
           foaf:name ?N ;
           foaf:nick ?M . }
GROUP BY ?P
```

```
@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

Details:

http://www.w3.org/TR/sparql11-query/#aggregateAlgebra

SIEMENS

Evaluate a list of (GROUP BY) expressions:

ListEval(ExprList, *μ*) returns a list E, where E[i] = μ(ExprList[i])

Use these to partition a solution sequence:

Group((), Ω) = { 1 → Ω }

Group(ExprList, Ω) = { ListEval(ExprList, μ) →
{ μ' | μ' in Ω, ListEval(ExprList, μ) = ListEval(ExprList, μ') } | μ in Ω }

produces a ***grouped solution sequence***

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

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

Group((?x), S) = { (2) → ({?x→2, ?y→3}, {?x→2, ?y→5}),
(6) → ({?x→6, ?y→7}) } }

Aggregates - Semantics

Ommitted details on error handling and scalar Parameters like “SEPERATOR” in GROUP_CPONCAT

Definition: Aggregation (*simplified*)

Aggregation applies set function “func” (e.g. sum, min, max, …) to the **multiset** obtained from applying a **list of expressions** to 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).

Aggregation(ExprList, func, G) = {g → F(Ω) | g → Ω in G }

where $\mathbf{M(\Omega)} = \{ \text{ListEval(ExprList, } \mu) \mid \mu \text{ in } \Omega))$

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

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

G = { (2) → ({?x→2, ?y→3}, {?x→2, ?y→5}),
(6) → ({?x→6, ?y→7}) } }

Aggregation(?y, Sum, G) = { (2) → Sum((3), (5)) , (6) → Sum((7)) }
= { (2) → 8 , (6) → 7 }

SIEMENS

Aggregates - Semantics

Definition: Aggregation (*simplified*)

Aggregation applies set function “func” (e.g. sum, min, max, …) to the **multiset** obtained from applying a **list of expressions** to 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).

Aggregation(ExprList, func, G) = {g → F(Ω) | g → Ω in G }

where **M(Ω)** = { ListEval(ExprList, μ) | μ in Ω))

F(Ω) = func(M(Ω)), for non-`DISTINCT`

F(Ω) = func(Distinct(M(Ω))), for `DISTINCT`

G = { (2) → ({?x→2, ?y→3}, {?x→2, ?y→3}),
(6) → ({?x→6, ?y→7}) } }

Aggregation(?y, Sum, G) = { (2) → Sum((3), (3)) , (6) → Sum((7)) }
= { (2) → 6 , (6) → 7 }

Aggregates - Semantics

Definition: Aggregation (*simplified*)

Aggregation applies set function "func" (e.g. sum, min, max, …) to the **multiset** obtained from applying a **list of expressions** to 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).

Aggregation(ExprList, func, G) = {g → F(Ω) | g → Ω in G }

where **M(Ω)** = { ListEval(ExprList, μ) | μ in Ω))

F(Ω) = func(M(Ω)), for non-`DISTINCT`

F(Ω) = func(Distinct(M(Ω))), for `DISTINCT`

G = { (2) → ({?x→2, ?y→3}, {?x→2, ?y→3}),
(6) → ({?x→6, ?y→7}) } }

Aggregation(?y, Sum(DISTINCT), G) = { (2) → Sum(DISTINCT((3), (3))) , (6) → Sum(DISTINCT(((7))) }
= { (2) → 3 , (6) → 7 }

Aggregates - Semantics

Definition: Aggregation (*simplified*)

Aggregation applies set function “func” (e.g. sum, min, max, …) to the **multiset** obtained from applying a **list of expressions** to 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).

$$\textbf{Aggregation(ExprList, func, G)} = \{g \to F(\Omega) \mid g \to \Omega \text{ in } G\}$$

where $\mathbf{M(\Omega)} = \{ \text{ListEvalE(ExprList}, \mu) \mid \mu \text{ in } \Omega))$

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

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

Aggregations are subsequently mapped back via to solution multisets in the evaluation of SELECT expressions, cf. http://www.w3.org/TR/sparql11-query/#sparqlSelectExpressions

G = { (2) → ({?x→2, ?y→3}, {?x→2, ?y→5}),
(6) → ({?x→6, ?y→7}) } }

Aggregation(?y, Sum, G) = { (2) → Sum((3), (5)) , (6) → Sum((7)) }
= { (2) → 8, (6) → 7 }

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

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

SIEMENS

Subqueries

Subqueries to realise complex mappings

- How to concatenate first name and last name?
- Wasn't possible in SPARQL 1.0 … Now possible without problems per subqueries!

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


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

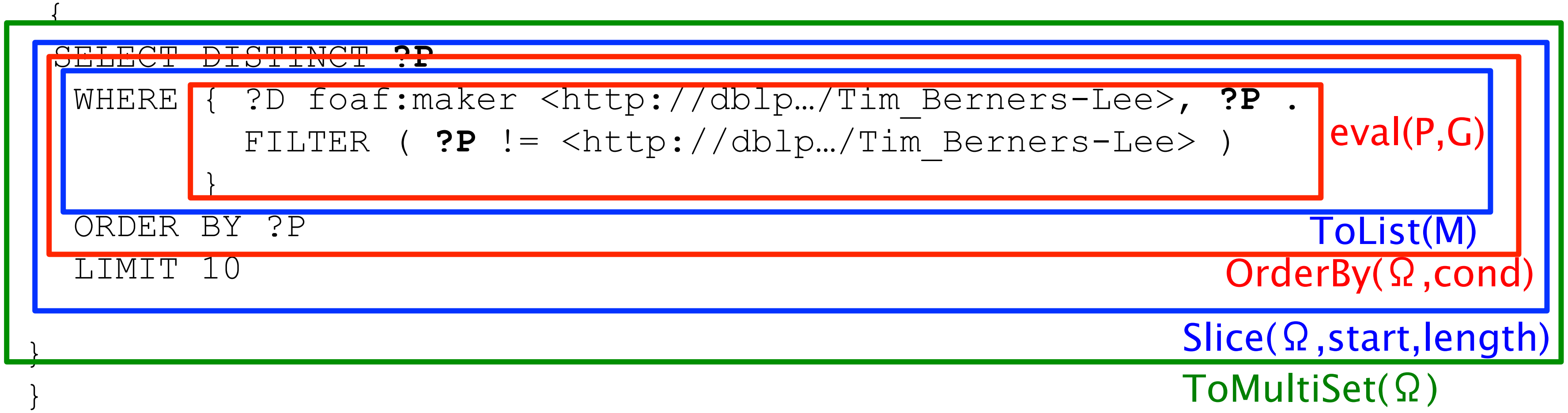
Subqueries “Limit per resource”

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

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

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
 }
}
```

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

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*

subtle difference, not relevant for most queries… but let's look into it

MINUS and NOT EXISTS

May have different results, e.g.:

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

SELECT *
WHERE{ ?S ?P ?O
                  FILTER( NOT EXISTS { ex:a ex:b ex:c } ) }
```

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


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

?S	?P	?O

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

Property Path Expressions

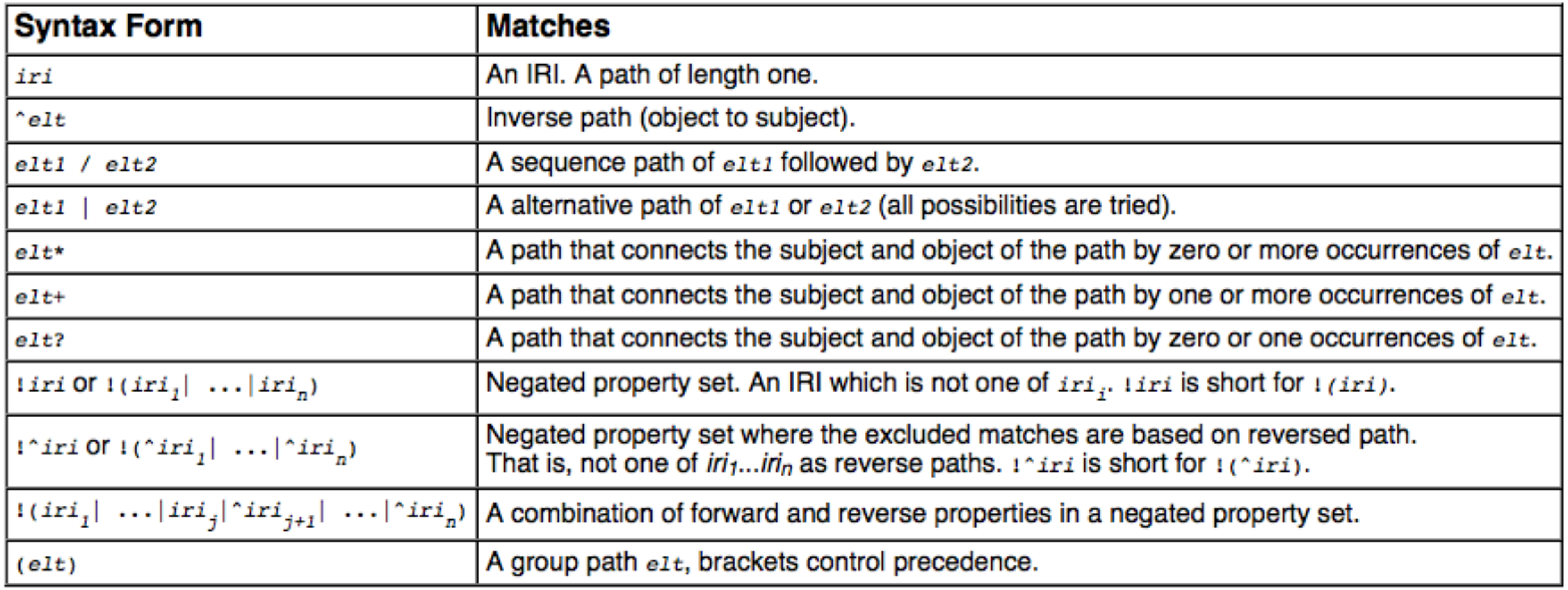
Property Path Expressions

Arbitrary Length paths, Concatenate property 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
iri	An IRI. A path of length one.
^*elt*	Inverse path (object to subject).
elt1 / *elt2*	A sequence path of *elt1* followed by *elt2*.
elt1 | *elt2*	A alternative path of *elt1* or *elt2* (all possibilities are tried).
*elt**	A path that connects the subject and object of the path by zero or more occurrences of *elt*.
elt+	A path that connects the subject and object of the path by one or more occurrences of *elt*.
elt?	A path that connects the subject and object of the path by zero or one occurrences of *elt*.
!*iri* or !(iri_1| ...|iri_n)	Negated property set. An IRI which is not one of iri_i. !*iri* is short for !(*iri*).
!^*iri* or !(^iri_1| ...|^iri_n)	Negated property set where the excluded matches are based on reversed path. That is, not one of iri_1...iri_n as reverse paths. !^*iri* is short for !(^*iri*).
!(iri_1| ...|iri_j|^iri_{j+1}| ...|^iri_n)	A combination of forward and reverse properties in a negated property set.
(*elt*)	A group path *elt*, brackets control precedence.

SIEMENS

Path Expressions: Semantics

- Semantics defined mostly in terms of rewriting:

  / … rewrites to a sequence of patterns

  | … rewrites to UNION

  ^ … rewrites to inverted path

  ? … new algebra function ZeroOrOnePath()

  * … new algebra function ZeroOrMorePath()

  + … new algebra function OneOrMorePath()

- A lot of last-minute discussion about semantics (counting vs. non-counting) see also [Arenas, Conca, Pérez, WWW2012] and [Losemann, Martens, PODS2012] → ***Detailed presentation of Property Paths and their semantis: possible topic for a student presentation!***

SPARQL 1.1 extended function library

Many new functions as opposed to SPARQL1.0:

Mentioned a few already:

- coalesce
- if
- isNumeric

Many new functions for strings, e.g. strbefore(), strafter(), …

See full list (snapshot) at:

http://www.w3.org/2009/sparql/docs/query-1.1/rq25.xml#SparqlOps

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)***

We will focus on these in this lecture

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

SPARQL Basic Federated Query

Allows you to query a remote endpoint from “WITHIN” your query…
Keyword **SERVICE**

Can be used e.g. to compute aggregates from an endpoint that doesn’t yet support them, e.g. SPARQL 1.1 for dbpedia, e.g. “*How many inhabintants do Austria’s top-3 cities have in total (sum)?”*

Using ARQ:

```
SELECT (SUM(?pop) AS ?P )
 { SERVICE <http://dbpedia.org/sparql/>
   { SELECT DISTINCT ?C ?pop
     WHERE {
     ?C <http://dbpedia.org/ontology/populationTotal> ?pop ;
     <http://dbpedia.org/ontology/country> <http://dbpedia.org/resource/Austria> .
     [] <http://dbpedia.org/property/city> ?C .
     }
     ORDER BY DESC ( ?pop )
     LIMIT 3
   }
}
```

SPARQL 1.1 Entailment

SPARQL 1.1 Entailment:
Example where Reasoning is needed

Give me all facts about Tim Berners-Lee from DBPEdia and DBLP?

```
SELECT ?P ?O
WHERE { <http://dbpedia.org/resource/Tim_Berners-Lee>  ?P ?O }
```

If I ask this query to DBPedia, I get quite some results…
… but not if I ask the same query to DBLP.

Because:
a) DBLP does not “know” that
`http://dbpedia.org/resource/Tim_Berners-Lee`
=
http://dblp.l3s.de/d2r/page/authors/Tim_Berners-Lee

b) SPARQL can’t follow links (more on that in the **one of the next** lectures)

SPARQL 1.1 Entailment: OWL

SPARQL 1.1 “understands” OWL:

```
<http://dbpedia.org/resource/Tim_Berners-Lee>
  foaf:homepage
         <http://www.w3.org/People/Berners-Lee/> .
```

dbpedia.org

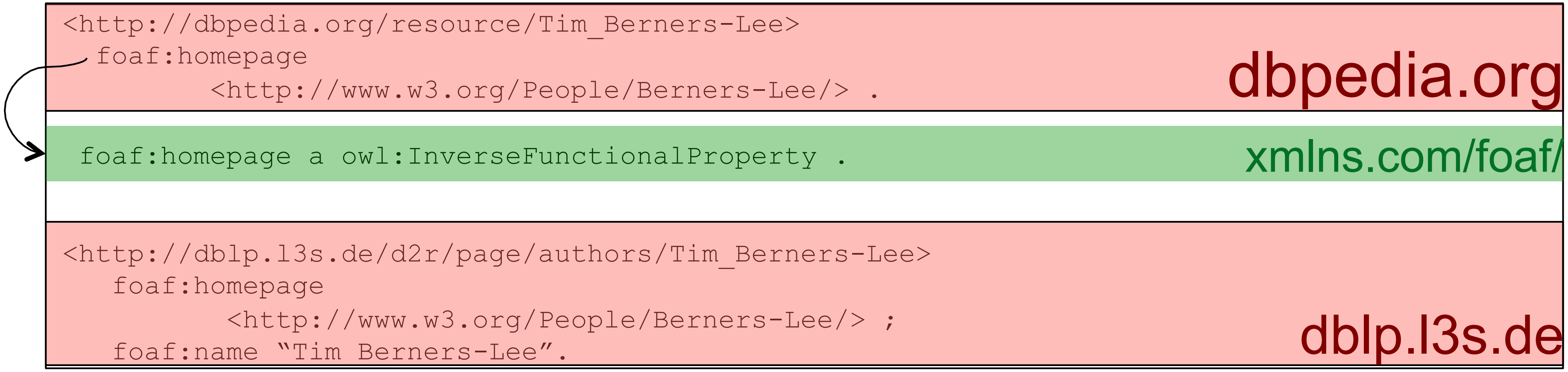
```
<http://dblp.l3s.de/d2r/page/authors/Tim_Berners-Lee>
   foaf:homepage
          <http://www.w3.org/People/Berners-Lee/> ;
   foaf:name “Tim Berners-Lee”.
```

dblp.l3s.de

```
SELECT ?O
FROM <dbpedia.org>
FROM <dblp.l3s.de>
WHERE { <http://dbpedia.org/resource/Tim_Berners-Lee>  foaf:name ?O }
```

SPARQL 1.1 Entailment: OWL

SPARQL 1.1 “understands” OWL:



```
<http://dbpedia.org/resource/Tim_Berners-Lee>
  foaf:homepage
         <http://www.w3.org/People/Berners-Lee/> .
```

dbpedia.org

```
 foaf:homepage a owl:InverseFunctionalProperty .
```

xmlns.com/foaf/

```
<http://dblp.l3s.de/d2r/page/authors/Tim_Berners-Lee>
   foaf:homepage
          <http://www.w3.org/People/Berners-Lee/> ;
   foaf:name “Tim Berners-Lee”.
```

dblp.l3s.de

```
SELECT ?O
FROM <dbpedia.org>
FROM <dblp.l3s.de>
WHERE { <http://dbpedia.org/resource/Tim_Berners-Lee>  foaf:name ?O }
```

?O
“Tim Berners-Lee”

SPARQL 1.1 Entailment: OWL

SPARQL 1.1 “understands” OWL:

```
<http://dbpedia.org/resource/Tim_Berners-Lee>
  foaf:homepage
         <http://www.w3.org/People/Berners-Lee/> .

 foaf:homepage a owl:InverseFunctionalProperty .
```

$\top \sqsubseteq \leq 1.homepage$

=

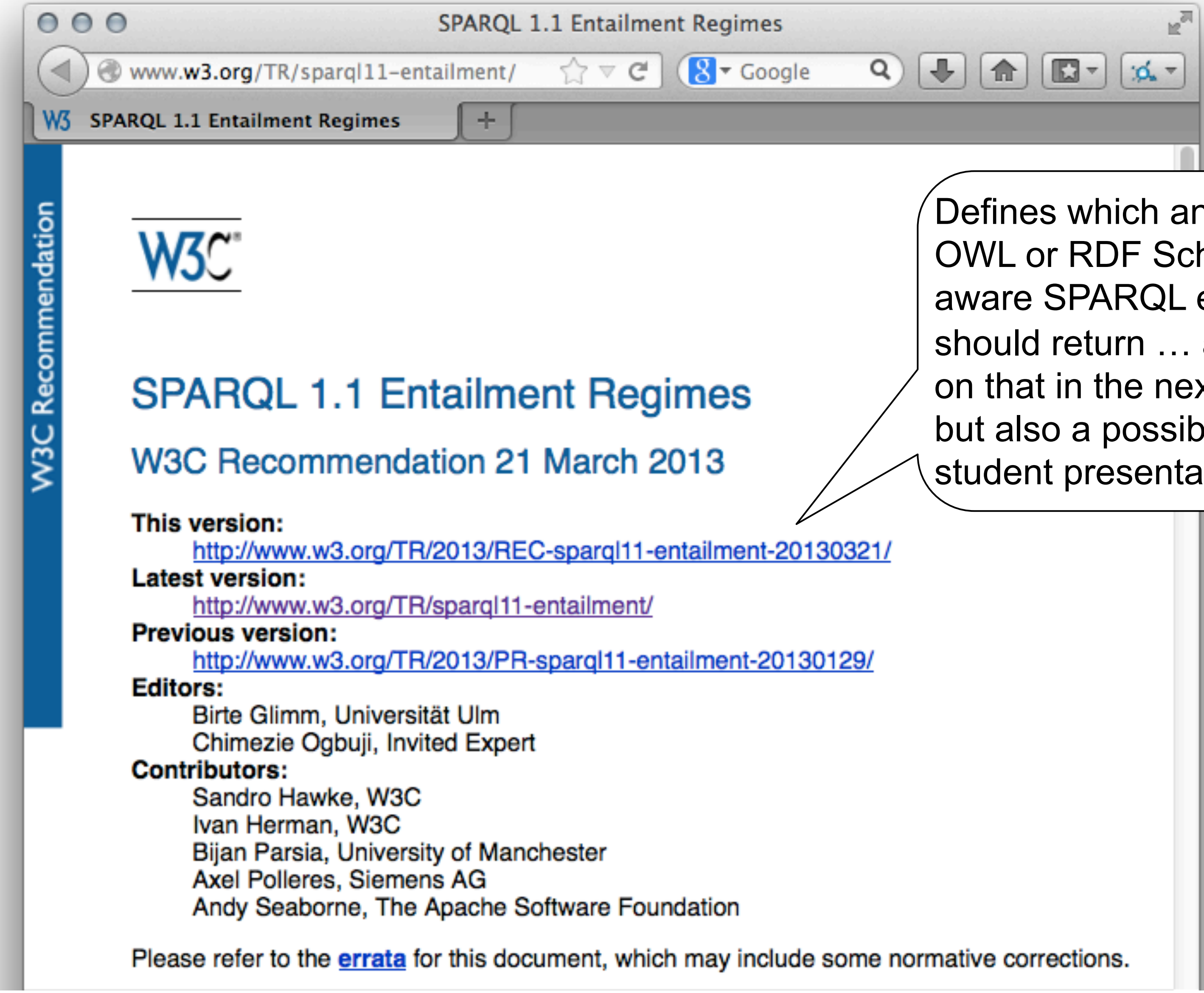
```
<http://dbpedia.org/resource/Tim_Berners-Lee>      >
   foaf:homepage
          <http://www.w3.org/People/Berners-Lee/> ;
   foaf:name “Tim Berners-Lee”.
```

```
SELECT ?O
FROM <dbpedia.org>
FROM <dblp.l3s.de>
WHERE { <http://dbpedia.org/resource/Tim_Berners-Lee>  foaf:name ?O }
```

?O
“Tim Berners-Lee”

SIEMENS

SPARQL 1.1 Entailment: OWL


SPARQL 1.1 Entailment Regimes
www.w3.org/TR/sparql11-entailment/
Google
SPARQL 1.1 Entailment Regimes
W3C Recommendation
W3C

SPARQL 1.1 Entailment Regimes

W3C Recommendation 21 March 2013

This version:
http://www.w3.org/TR/2013/REC-sparql11-entailment-20130321/
Latest version:
http://www.w3.org/TR/sparql11-entailment/
Previous version:
http://www.w3.org/TR/2013/PR-sparql11-entailment-20130129/
Editors:
Birte Glimm, Universität Ulm
Chimezie Ogbuji, Invited Expert
Contributors:
Sandro Hawke, W3C
Ivan Herman, W3C
Bijan Parsia, University of Manchester
Axel Polleres, Siemens AG
Andy Seaborne, The Apache Software Foundation

Please refer to the errata for this document, which may include some normative corrections.

Defines which answers an OWL or RDF Schema-aware SPARQL engine should return … a bit more on that in the next lecture, but also a possible topic for student presentation!

SPARQL 1.1 Update

SQL has not only a query language, but also a Data manipulation language.
→SPARQL Update to fill this gap:

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

→ Allows to change/update an RDF Store from outside, again via standard HTTP protocol.

Note: security issues are a separate issue, not prescribed yet by the standard!

Some implementations of SPARQL 1.1 :

Some current (partial) SPARQL1.1 implementations:

Jena 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… Virtuoso (e.g. dbpedia) seems to support most of SPARQL1.1 already.

References

Find all SPARQL 1.1 Drafts here: http://www.w3.org/2009/sparql/wiki/Main_Page

Papers:

[Losemann, Martens, PODS2012] Katja Losemann, Wim Martens: The complexity of evaluating path expressions in SPARQL. PODS 2012: 101-112

[Arenas, Conca, Pérez, WWW2012] Marcelo Arenas, Sebastián Conca, Jorge Pérez: Counting beyond a Yottabyte, or how SPARQL 1.1 property paths will prevent adoption of the standard. WWW 2012: 629-638

[Gutierrez et al. 2011, ESWC] Claudio Gutierrez, Carlos A. Hurtado, Alejandro A. Vaisman: RDFS Update: From Theory to Practice. ESWC (2) 2011: 93-107

[Angles, Gutierrez, AMW2011] Renzo Angles, Claudio Gutierrez: Subqueries in SPARQL. AMW 2011

[Hartig, Bizer ,Freytag 2009] Olaf Hartig, Christian Bizer, Johann Christoph Freytag: Executing SPARQL Queries over the Web of Linked Data. International Semantic Web Conference 2009: 293-309

[Hartig 2012] Olaf Hartig: SPARQL for a Web of Linked Data: Semantics and Computability. ESWC 2012: 8-23

[Fionda et al., WWW2012] Valeria Fionda, Claudio Gutierrez, Giuseppe Pirrò: Semantic navigation on the web of data: specification of routes, web fragments and actions. WWW 2012: 281-290

Assignment 2:

ATTENTION: If you have NOT received an email with feedback, I have NOT received your assignment!

Discuss Assignment 2 and common problems:

What is RDF Entailment (as opposed to Simple Entailment)?

Don’t interpret things into the OWL ontology that aren’t said there!

ill-typed literals alone don’t cause D-inconsistency

provide some examples of OWL inconsistencies

Time allowed:

If you have your solutions here, or have read my feedback already, we can also go through it.

Assignment 3:

Check http://www.polleres.net/teaching/SemWebTech_2013/#assignment3

Deadline: 31 May 2013

ATTENTION: Pick a topic for the final presentation until next time!

Student presentations:

I have time to discuss your proposals still, if you have some already, otherwise, more topics by next time.

Who has sent me a topic suggestion already?

- SPARQL GUIs (F. J. Ekaputra)
- Good Relations Ontology and use (B.Ege)

Who plans to still do?

Some suggested topics (which we can assign now already – first come, first serve:

- W3C RDF1.1 WG – status semantics, changes, semantics for named graphs, etc.
- W3C Linked Data Platform WG – http://www.w3.org/2012/ldp/wiki/Main_Page
- SPARQL1.1 spec parts which we don't cover in detail (e.g. Entailment Regimes, Update, Semantics of Property Paths, etc.) - Jonas
- OWL2 and meta-modeling
- SKOS
- RDFa & schema.org
- Berlin SPARQL Benchmark, latest edition, see http://lists.w3.org/Archives/Public/semantic-web/2013Apr/0237.html

More own topics suggestions welcome!

Presentations

First slot:	24/06/2013
Second slot:	25/06/2013

Send me the slides at least 1 week in advance per email!
→ You should start to work on the topic soon!