Introduction to the Semantic Web Technologies

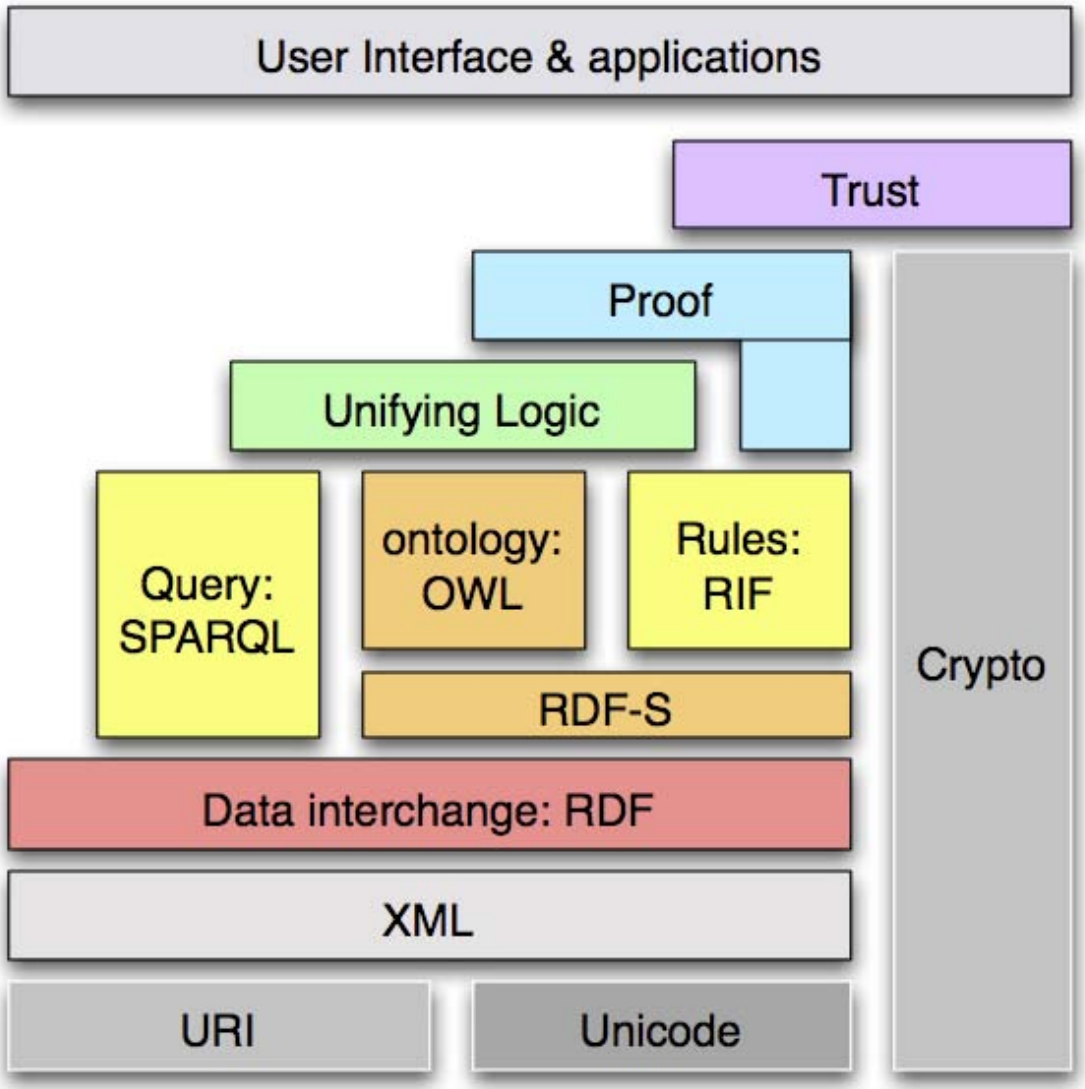
Konstantin Schekotihin

Alpen-Adria-Universität
Klagenfurt, Austria

Agenda

1. SPARQL
   - Introduction and syntax
   - Querying RDF with SPARQL

Semantic Web Technology Stack



Introduction I

SPARQL is an RDF query language that allows us to:

- Retrieve values from RDF documents
- Explore triple stores by querying for classes and properties
- Execute complex SQL-like queries on single/multiple triple stores
- Return results in form of RDF (transformations)

SPARQL uses Turtle syntax:

- namespaces `@prefix ns:<URI> .`
- literals `"01"^^xsd:decimal, "string"@de`
- triple `ns:s ns:p ns:o .`

SPARQL Recommendations

- SPARQL 1.0 (January 2008) includes
  - SPARQL 1.0 Query Language
  - SPARQL 1.0 Protocol
  - SPARQL Results XML Format
- SPARQL 1.1 (March 2013)
  - SPARQL 1.1 Query Language
  - SPARQL 1.1 Update Language
  - SPARQL 1.1 Protocol for RDF
  - SPARQL 1.1 Graph Store HTTP Protocol
  - SPARQL 1.1 Entailment Regimes
  - SPARQL 1.1 Service Description
  - SPARQL 1.1 Federated Query
  - SPARQL 1.1 Conformance Tests
  - SPARQL 1.1 Query Results JSON Format
  - SPARQL 1.1 Query Results CSV and TSV Formats
  - SPARQL Query Results XML Format

SPARQL Query Structure I

```
# declarations of namespaces
PREFIX ex: <http://example.com/version/1.0> .
# definition of result clause
SELECT ?var1 ?var2
# definition of datasets
FROM <http://example.com/DigitalCamera.rdf>
# conditions
WHERE {
    ?var1 ex:madeBy ?var2 .
}
# modifiers
ORDER BY ?var1
```

SPARQL Query Structure II

- Variable names start with ?, which can match any node of a given RDF graph
- SELECT returns a table of substitutions (resources/literals) for variables in a query
- FROM indicates the input graph
- WHERE specifies patterns to be found in the graph
- Conditions on matched subgraphs are applied by means of filters (later)
- ORDER BY specifies the order of substitutions in the resulting table

SPARQL Endpoints

- SPARQL queries work on subgraphs of a given RDF graph
- Endpoints are service providers: accept queries and return results
- One can differentiate between generic and specific endpoints
  - Generic use HTTP to retrieve the data from the Web
  - Specific are connected with a particular RDF dataset
- All communication with endpoints is done over HTTP
- A variety of libraries/formats can be used for communication:
  - XML – results are returned in XML format using a specific predefined vocabulary
  - RDF – "construction" queries require SPARQL engine to respond in RDF format (RDF/XML, Turtle, N-Triples, etc.)
  - JSON – adapter for XML vocabulary which is useful in Web application
- XSLT allows to transform XML results directly to HTML

SPARQL Engines

- OpenJena's ARQ http://sparql.org/sparql.html
- Stardog http://stardog.com/
- OpenLink's Virtuoso http://demo.openlinksw.com/sparql
- Redland's Rasqal http://librdf.org/query/

Sample dataset: http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
# use * to select all variables
SELECT ?name ?email
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE {
    ?person foaf:name ?name .
    ?person foaf:mbox ?email .
}
```

Complex constraints

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?homepage
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE {
    <http://www.w3.org/People/Berners-Lee/card#i> foaf:knows
    ?known .
    ?known foaf:homepage ?homepage .
}
```

<http://www.w3.org/People/Berners-Lee/card#i> is a URI defining Tim Berners-Lee himself.

Exploring DBPedia I

- DBPedia is an RDF dataset created from Wikipedia's infoboxes, categories, external links, etc.
- DBPedia 2015-04 describes "5.9M things out of which 4.3M resources have abstracts, 452K geo coordinates and 1.45M depictions. In total, 4 million resources are classified in a consistent ontology and consists of 2,06M persons, 682K places, 376K creative works, 188K organizations, 278K species and 5K diseases."[1]

Exploring DBPedia II

Use this query and DBPedia endpoint[2] to retrieve only 50 top concept names ordered alphabetically starting from 10^{th}

```
SELECT DISTINCT ?Concept
WHERE {
    [] a ?Concept .
}
# enabling ordering will considerably increase processing time
# ORDER BY ?Concept
LIMIT   50
OFFSET  10
```

[1] http://blog.dbpedia.org/?p=148
[2] DBPedia SPARQL endpoint `http://dbpedia.org/sparql`

Filtering Query Results

Use boolean conditions to remove unwanted results

- Logical: `!`, `&&`, `||`
- Math: `+`, `-`, `*`, `/`
- Comparison: `=`, `!=`, `>`, `<`, `...`
- Build-in functions: `isURI`, `isBlank`, `isLiteral`, `bound`, `str`, `lang`, `datatype`, `sameTerm`, `langMatches`, `regex`
- SPARQL 1.1 extends this list!

```
SELECT ?place ?population ?comment
WHERE {
    ?place a dbo:Place ;
        dbo:country dbr:Austria ;
        rdfs:comment ?comment;
        dbo:populationTotal ?population .
    FILTER ((?population > 500000) &&
        langMatches(lang(?comment), "ru")) .
}
```

Optional results

SPARQL allows to define optional constraints.

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX tbl:  <http://www.w3.org/People/Berners-Lee/card#>
SELECT *
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE {
    tbl:i foaf:knows ?known .
    OPTIONAL { ?known foaf:homepage ?homepage }
}
```

Test the query with OpenJena's ARQ available at
`http://sparql.org/sparql.html`

Unite alternative results

Form a disjunction of two graph patterns

```
SELECT ?place ?population
WHERE {{
    ?place a dbo:Place ;
    dbo:country dbr:Austria ;
    dbo:populationTotal ?population .
    FILTER (?population > 100000) .
} UNION {
    ?place a dbo:City ;
    dbo:country dbr:Germany ;
    dbo:populationTotal ?population .
    FILTER (?population > 1000000) .
}}
```

- The query selects all Austrian regions and German cities with more than 10^5 and 10^6 inhabitants respectively.
- Test the query with DBPedia SPARQL endpoint http://dbpedia.org/sparql

Special queries

- CONSTRUCT returns an RDF graph

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT{
 ?person vCard:FN ?name .
 ?person vCard:EMAIL ?email .
}
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE {
    ?person foaf:name ?name   .
    ?person foaf:mbox  ?email .}
```

- ASK returns true if query pattern has any matches in the RDF graph, otherwise it returns false
- DESCRIBE returns some RDF that from the server's point of view describes a resource