Introduction to the Semantic Web Technologies

Kostyantyn Shchekotykhin

Alpen-Adria-Universität
Klagenfurt, Austria

Agenda

Introduction I

- SPARQL 1.1 is proposed as a W3C Recommendation, i.e. almost a standard
- Extends the original recommendation with a number of useful extensions
  - Aggregation functions: ability to group results and compute aggregate values such as sum, count, min, max, avg or sum
  - Subqueries: one query is embedded into another query
  - Projected expressions: query results might contain values given as constants or returned by function or some expressions in the SELECT statement
  - Negation: improved syntax for negation
  - Property paths: query length paths in a graph specifying means of "regular expressions"
  - Support of basic federated queries: split single query among endpoints and combine the results

Introduction II

- Improved compatibility with other Semantic Web standards by support of different entailment regimes: RDF(S), OWL, RIF
- SPARQL UPDATE allows graphs manipulations such as insert, delete, create, drop and others
- Upwards compatible with SPARQL 1.0
- A number of implementations with different levels of support are available, e.g. Jena ARQ

Project expressions

- The query computes length of homepage URLs found in the graph

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX tbl:<http://www.w3.org/People/Berners-Lee/card#>

SELECT ?known ?homepage (STRLEN(str(?homepage)) AS ?len)
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE {
    tbl:i foaf:knows ?known .
    OPTIONAL { ?known foaf:homepage ?homepage }}
```

- Use BIND to assign values to variables
- Order of declarations is important!

```
SELECT ?known ?homepage ?len
WHERE {
    ...
    OPTIONAL { ?known foaf:homepage ?homepage }
    BIND(STRLEN(str(?homepage)) AS ?len)}
```

ALPEN-ADRIA UNIVERSITÄT KLAGENFURT I WIEN GRAZ

Aggregate queries I

- Aggregate queries postprocess the results returned by a query by partitioning them into groups and applying aggregation functions to them
- GROUP BY specifies variables which values should used to partition the result set
- Common aggregation functions COUNT, MIN, MAX, SUM, AVG
- SAMPLE randomly selects one of the groups values
- GROUP_CONCAT concatenates values of a group using a given string as a separator
- Use HAVING to filter results after application of aggregates

Aggregate queries II

- The following query selects all persons known by TBL and counts the number of their emails

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX tbl:<http://www.w3.org/People/Berners-Lee/card#>
SELECT ?known (COUNT(DISTINCT ?mail) AS ?mails)
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE {
    tbl:i foaf:knows ?known .
    ?known foaf:mbox ?mail .
}
GROUP BY ?known
```

Subqueries I

- We want to select names of all persons who have more than 1 mailbox
- First we find the target persons and then get their names

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mboxcount
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE {{
    SELECT  ?person (COUNT(?mbox) AS ?mboxcount)
    WHERE { ?person foaf:mbox ?mbox }
    GROUP BY ?person
    HAVING (COUNT(?mbox) > 1)}
  ?person foaf:name ?name .
}
```

Subqueries II

- We can also select all persons who do not have a mail box

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX tbl:<http://www.w3.org/People/Berners-Lee/card#>
SELECT ?name ("n/a" AS ?mail)
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE {{
    SELECT  ?person (COUNT(?mbox) AS ?mb)
    WHERE { tbl:i foaf:knows ?person .
        OPTIONAL {?person foaf:mbox ?mbox}}
    GROUP BY ?person
    HAVING (COUNT(?mbox) = 0)}
  ?person foaf:name ?name .
}
```

Negation I

- In SPARQL 1.0 one can simulate negation by a combination of OPTIONAL with BOUND filter and logical NOT operator
- OPTIONAL selects values of variables we want to exclude and BOUND removes them
- Select all persons who do not have emails

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?name
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE {
   ?person foaf:name ?name .
   OPTIONAL { ?person foaf:mbox ?mbox }
   FILTER(!bound(?mbox))
}
```

Negation II

- Use MINUS to remove all bindings that match the right-hand side of the expression

```
MINUS { ?person foaf:mbox ?mbox }
```

- NOT EXISTS filter that given bindings from the result tests whether or not a given pattern exists in a graph

```
FILTER(NOT EXISTS { ?person foaf:mbox ?mbox })
```

Property paths

- Allow querying for arbitrary length path through the graph
- Reuse syntax of regular expressions

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

Federated queries

- Allow querying remote endpoints for data
- Use SERVICE keyword to denote the part of a query that should be processed by an endpoint
- Is not widely deployed yet

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX tbl:<http://www.w3.org/People/Berners-Lee/card#>

SELECT ?person ?pic
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE{
    tbl:i foaf:knows  ?person .
    SERVICE <http://dbpedia.org/sparql>
            {?person foaf:depiction ?pic .}
    FILTER (!isBlank(?person))
}
```

SPARQL 1.1 Update

- SPARQL 1.1 Update is a language for managing and updating RDF graphs.
  - INSERT DATA { triples }
  - DELETE DATA { triples }
  - [DELETE { template }] [INSERT { template }] WHERE { pattern }
  - LOAD uri [INTO GRAPH uri]
  - CLEAR GRAPH uri
  - CREATE GRAPH uri
  - DROP GRAPH uri

Entailment regimes

- How can we get that mozzarellaDiBufala is a Cheese?
- SPARQL 1.1 answer the query in an OWL entailment regime
- It requires OWL reasoner to classify the ontology

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix pizzas: <http://ainf.at/pizzas#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
pizzas:Cheese a owl:Class .
pizzas:Mozzarella a owl:Class .
pizzas:onTop a owl:ObjectProperty .
pizzas:Pizza a owl:Class; rdfs:subClassOf _:b1 .
_:b1 a owl:Restriction; owl:allValuesFrom pizzas:Cheese;
                        owl:onProperty pizzas:onTop .
pizzas:Margherita pizzas:onTop pizzas:mozzarellaDiBufala;
                  a pizzas:Pizza, owl:NamedIndividual.
pizzas:mozzarellaDiBufala a pizzas:Mozzarella,
                            owl:NamedIndividual .
```