Universität Freiburg
Institut für Informatik
Prof. Dr. G. Lausen
Alexander Schätzle
Martin Przyjaciel-Zablocki

Georges-Köhler Allee, Geb. 51
D-79110 Freiburg
lausen@informatik.uni-freiburg.de
schaetzle@informatik.uni-freiburg.de
zablocki@informatik.uni-freiburg.de

Data Models and Query Languages
Summerterm 2013

6. Exercise Sheet: RDF & SPARQL

Discussion: 28.06.2013

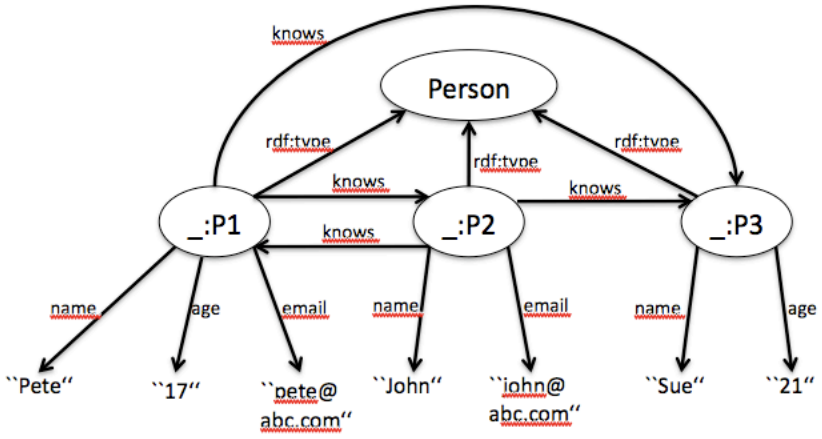
Exercise 1 (Evaluating SPARQL Queries)
Consider the RDF database

D := { (_:P1,rdf:type,Person), (_:P1,name,"Pete"), (_:P1,age,"17"), (_:P1,email,"pete@abc.com"),
(_:P2,rdf:type,Person), (_:P2,name,"John"), (_:P2,email,"john@abc.com"),
(_:P3,rdf:type,Person), (_:P3,name,"Sue"), (_:P3,age,"21"),
(_:P1,knows,_:P2), (_:P1,knows,_P3), (_:P2,knows,_:P1), (_:P2,knows,_:P3) }.

Draw the RDF graph. Evaluate the following SPARQL graph patterns step by step according to the semantics introduced in the lecture and phrase their semantics in plain English.

Assume that every of the following queries is preceded by the necessary namespace definitions and the Select * solution format.

RDF Graph:



a) `{ ?p rdf:type Person. ?p age ?age. FILTER (?age>20) }`

In Worten: URI und Alter aller Personen, die älter als 20 Jahre sind.

Zwischenrechnung (AND-Ausdrucks separat):

$[\![((?p, \textit{rdf:type}, \textit{Person}) \text{ AND } (?p, \textit{age}, ?\textit{age}))]\!]_D$
$= [\![((?p, \textit{rdf:type}, \textit{Person})]\!]_D \bowtie [\![(?p, \textit{age}, ?\textit{age}))]\!]_D$
$= \{\{?p \mapsto \_:P1\}, \{?p \mapsto \_:P2\}, \{?p \mapsto \_:P3\}\} \bowtie \{\{?p \mapsto \_:P1, ?\textit{age} \mapsto \text{“17”}\}, \{?p \mapsto \_:P3, ?\textit{age} \mapsto \text{“21”}\}\}$
$= \{\{?p \mapsto \_:P1, ?\textit{age} \mapsto \text{“17”}\}, \{?p \mapsto \_:P3, ?\textit{age} \mapsto \text{“21”}\}\}$

Als Gesamtergebnis ergibt sich also:

$[\![((?p, \textit{rdf:type}, \textit{Person}) \text{ AND } (?p, \textit{age}, ?\textit{age})) \text{ FILTER } (?\textit{age} > 20)]\!]_D$
$= \{\mu \in [\![((?p, \textit{rdf:type}, \textit{Person}) \text{ AND } (?p, \textit{age}, ?\textit{age}))]\!]_D \mid \mu \models (?\textit{age} > 20)\}$
$= \{\{?p \mapsto \_:P3, ?\textit{age} \mapsto \text{“21”}\}\}$

b) `{ { ?p rdf:type Person. ?p name ?name. } OPTIONAL { ?p age ?age . } }`

In Worten: Alle Personen mit Name und, falls vorhanden, deren Alter.

Lösung:

$[\![((?p, \textit{rdf:type}, \textit{Person}) \text{ AND } (?p, \textit{name}, ?\textit{name})) \text{ OPT } (?p, \textit{age}, ?\textit{age})]\!]_D$
$= [\![(?p, \textit{rdf:type}, \textit{Person}) \text{ AND } (?p, \textit{name}, ?\textit{name})]\!]_D =\bowtie [\![(?p, \textit{age}, ?\textit{age})]\!]_D$
$= ([\![(?p, \textit{rdf:type}, \textit{Person})]\!]_D \bowtie [\![(?p, \textit{name}, ?\textit{name})]\!]_D) =\bowtie [\![(?p, \textit{age}, ?\textit{age})]\!]_D$
$= (\{\{?p \mapsto \_:P1\}, \{?p \mapsto \_:P2\}, \{?p \mapsto \_:P3\}\}$
$\quad \bowtie \{\{?p \mapsto \_:P1, ?\textit{name} \mapsto \text{“Pete”}\}, \{?p \mapsto \_:P2, ?\textit{name} \mapsto \text{“John”}\}, \{?p \mapsto \_:P3, ?\textit{name} \mapsto \text{“Sue”}\}\})$
$\quad =\bowtie \{\{?p \mapsto \_:P1, ?\textit{age} \mapsto \text{“17”}\}, \{?p \mapsto \_:P3, ?\textit{age} \mapsto \text{“21”}\}\}$
$= \{\{?p \mapsto \_:P1, ?\textit{name} \mapsto \text{“Pete”}\}, \{?p \mapsto \_:P2, ?\textit{name} \mapsto \text{“John”}\}, \{?p \mapsto \_:P3, ?\textit{name} \mapsto \text{“Sue”}\}\}$
$\quad =\bowtie \{\{?p \mapsto \_:P1, ?\textit{age} \mapsto \text{“17”}\}, \{?p \mapsto \_:P3, ?\textit{age} \mapsto \text{“21”}\}\}$
$= (\{\{?p \mapsto \_:P1, ?\textit{name} \mapsto \text{“Pete”}\}, \{?p \mapsto \_:P2, ?\textit{name} \mapsto \text{“John”}\}, \{?p \mapsto \_:P3, ?\textit{name} \mapsto \text{“Sue”}\}\}$
$\quad \bowtie \{\{?p \mapsto \_:P1, ?\textit{age} \mapsto \text{“17”}\}, \{?p \mapsto \_:P3, ?\textit{age} \mapsto \text{“21”}\}\}) \cup$
$\quad (\{\{?p \mapsto \_:P1, ?\textit{name} \mapsto \text{“Pete”}\}, \{?p \mapsto \_:P2, ?\textit{name} \mapsto \text{“John”}\}, \{?p \mapsto \_:P3, ?\textit{name} \mapsto \text{“Sue”}\}\}$
$\quad \setminus \{\{?p \mapsto \_:P1, ?\textit{age} \mapsto \text{“17”}\}, \{?p \mapsto \_:P3, ?\textit{age} \mapsto \text{“21”}\}\})$
$= \{\{?p \mapsto \_:P1, ?\textit{name} \mapsto \text{“Pete”}, ?\textit{age} \mapsto \text{“17”}\}, \{?p \mapsto \_:P3, ?\textit{name} \mapsto \text{“Sue”}, ?\textit{age} \mapsto \text{“21”}\}\} \cup$
$\quad \{\{?p \mapsto \_:P2, ?\textit{name} \mapsto \text{“John”}\}\}$
$= \{\{?p \mapsto \_:P1, ?\textit{name} \mapsto \text{“Pete”}, ?\textit{age} \mapsto \text{“17”}\}, \{?p \mapsto \_:P2, ?\textit{name} \mapsto \text{“John”}\},$
$\quad \{?p \mapsto \_:P3, ?\textit{name} \mapsto \text{“Sue”}, ?\textit{age} \mapsto \text{“21”}\}\}$

c) `{ { ?p rdf:type Person. ?p age ?age. } UNION { ?p rdf:type Person. ?p email ?email. }`

In Worten: Liste aller Personen die entweder ein Alter oder eine Emailadresse spezifiziert haben (Personen die beides haben tauchen doppelt im Endergebnis auf).

$[\![((?p, \mathit{rdf{:}type}, \mathit{Person})\ \textsc{And}\ (?p, \mathit{age}, ?\mathit{age}))\ \textsc{Union}\ ((?p, \mathit{rdf{:}type}, \mathit{Person})\ \textsc{And}\ (?p, \mathit{email}, ?\mathit{email}))]\!]_D$
$= ([\![(?p, \mathit{rdf{:}type}, \mathit{Person})]\!]_D \bowtie [\![(?p, \mathit{age}, ?\mathit{age})]\!]_D) \cup ([\![(?p, \mathit{rdf{:}type}, \mathit{Person})]\!]_D \bowtie [\![(?p, \mathit{email}, ?\mathit{email})]\!]_D)$
$= (\{\{?p \mapsto \_{:}P1\}, \{?p \mapsto \_{:}P2\}, \{?p \mapsto \_{:}P3\}\} \bowtie \{\{?p \mapsto \_{:}P1, ?\mathit{age} \mapsto \text{“17”}\}, \{?p \mapsto \_{:}P3, ?\mathit{age} \mapsto \text{“21”}\}\}) \cup$
$(\{\{?p \mapsto \_{:}P1\}, \{?p \mapsto \_{:}P2\}, \{?p \mapsto \_{:}P3\}\}$
$\bowtie \{\{?p \mapsto \_{:}P1, ?\mathit{email} \mapsto \text{“pete@abc.com”}\}, \{?p \mapsto \_{:}P2, ?\mathit{email} \mapsto \text{“john@abc.com”}\}\})$
$= \{\{?p \mapsto \_{:}P1, ?\mathit{age} \mapsto \text{“17”}\}, \{?p \mapsto \_{:}P3, ?\mathit{age} \mapsto \text{“21”}\}\} \cup$
$\{\{?p \mapsto \_{:}P1, ?\mathit{email} \mapsto \text{“pete@abc.com”}\}, \{?p \mapsto \_{:}P2, ?\mathit{email} \mapsto \text{“john@abc.com”}\}\}$
$= \{\{?p \mapsto \_{:}P1, ?\mathit{age} \mapsto \text{“17”}\}, \{?p \mapsto \_{:}P3, ?\mathit{age} \mapsto \text{“21”}\},$
$\{?p \mapsto \_{:}P1, ?\mathit{email} \mapsto \text{“pete@abc.com”}\}, \{?p \mapsto \_{:}P2, ?\mathit{email} \mapsto \text{“john@abc.com”}\}\}$

d) `{ { ?p rdf:type Person. OPTIONAL ?p email ?email. } FILTER (!bound(?email)) }`

In Worten: Alle Personen für die keine Email angegeben ist.

Zwischenrechnung (Opt-Ausdruck separat):

$[\![(?p, \mathit{rdf{:}type}, \mathit{Person})\ \textsc{Opt}\ (?p, \mathit{email}, ?\mathit{email})]\!]_D$
$= [\![(?p, \mathit{rdf{:}type}, \mathit{Person})]\!]_D =\!\bowtie [\![(?p, \mathit{email}, ?\mathit{email})]\!]_D$
$= \{\{?p \mapsto \_{:}P1\}, \{?p \mapsto \_{:}P2\}, \{?p \mapsto \_{:}P3\}\}$
$=\!\bowtie \{\{?p \mapsto \_{:}P1, ?\mathit{email} \mapsto \text{“pete@abc.com”}\}, \{?p \mapsto \_{:}P2, ?\mathit{email} \mapsto \text{“john@abc.com”}\}\}$
$= \ldots$
$= \{\{?p \mapsto \_{:}P1, ?\mathit{email} \mapsto \text{“pete@abc.com”}\}, \{?p \mapsto \_{:}P2, ?\mathit{email} \mapsto \text{“john@abc.com”}\}, \{?p \mapsto \_{:}P3\}\}$

Als Gesamtergebnis ergibt sich also:

$[\![((?p, \mathit{rdf{:}type}, \mathit{Person})\ \textsc{Opt}\ (?p, \mathit{email}, ?\mathit{email}))\ \textsc{Filter}\ (!\mathit{bound}(?\mathit{email}))]\!]_D$
$= \{\mu \in [\![((?p, \mathit{rdf{:}type}, \mathit{Person})\ \textsc{Opt}\ (?p, \mathit{email}, ?\mathit{email}))]\!]_D \mid \mu \models (!\mathit{bound}(?\mathit{email}))\}$
$= \{\{?p \mapsto \_{:}P3\}\}$

Verify your results using the ARQ SPARQL engine. An installation instruction, the above RDF document D, and example query a) are provided at the exercise page of the lecture homepage.

Exercise 2 (SPARQL Queries)
Consider the RDF database D from the previous exercise. Specify the following requests as SPARQL queries and indicate the final results obtained when evaluating them on document D.

a) All pairs of distinct persons that have a common friend (i.e., it must hold that the intersection of persons they know is non-empty).

```
SELECT ?p1 ?p2
WHERE {
  ?p1 rdf:type Person .
  ?p2 rdf:type Person .
  ?p1 knows ?p3 .
  ?p2 knows ?p3
```

```
  FILTER (?p1!=?p2)
}
```

Ergebnis auf Datenbank D:

$S = \{\{?p1 \mapsto \_:P1, ?p2 \mapsto \_:P2\}, \{?p1 \mapsto \_:P2, ?p2 \mapsto \_:P1\}$

b) The names of all persons that know at least one person or are younger than 20 years. If present, the email address and, also if present, the age of this person should be included in the result.

```
SELECT DISTINCT ?name ?email ?age
WHERE {
  ?p rdf:type Person .
  ?p name ?name.
  { { ?p knows ?p2 } UNION { ?p age ?age2 FILTER (?age2<20) } }
  OPTIONAL { ?p email ?email }
  OPTIONAL { ?p age ?age }
}
```

Ergebnis auf Datenbank D:

$S = \{\{?name \mapsto$ *"Pete"*$, ?email \mapsto$ *"pete@abc.com"*$, ?age \mapsto$ *"17"*$\}, \{?name \mapsto$ *"John"*$, ?email \mapsto$ *"john@ab*

c) Construct a new graph using the `CONSTRUCT` form that contains all persons (including their names) that know at least two persons.

```
CONSTRUCT { ?p rdf:type Person. ?p name ?name }
WHERE {
  ?p rdf:type Person .
  ?p name ?name .
  ?p knows ?p2 .
  ?p2 rdf:type Person .
  ?p knows ?p3 .
  ?p3 rdf:type Person
  FILTER (?p2!=?p3)
}
```

Ergebnis auf Datenbank D:

D' = { (_:P1,rdf:type,Person), (_:P1,name,"Pete"), (_:P2,rdf:type,Person), (_:P2,name,"John") }

d) All Persons, which are directly or indirectly connected via the knows-predicate.

```
SELECT ?p1 ?p2
WHERE {
  ?p1 knows+  ?p2   FILTER (?p1!=?p2)
}
```

Ergebnis auf Datenbank D:

e) All cyclic knows-relationships.

```
SELECT ?p1 ?p2
WHERE {
```

```
  ?p1 knows+  ?p2   FILTER (?p1!=?p2).
  ?p2 knows+  ?p1   FILTER (?p1!=?p2).
}
```

Ergebnis auf Datenbank D:

Verify your results using the ARQ engine.

Exercise 3 (Modelling in RDF)
Encode the following scenario in RDF. Modellieren Sie das folgende Szenario als RDF Datenbank.

Die Grundschule Freiburg hat drei MitarbeiterInnen: Lehrer Herr Maier, Lehrerin Frau Schmidt, sowie die Direktorin Frau Koster, die – zusätzlich zu ihren administrativen Aufgaben als Rektorin – auch Schüler unterrichtet. Herr Maier ist den Erstklässlern zugeteilt, während Frau Schmidt und Frau Koster gemeinsam die Klassen 2-4 *unterrichten. Herr Maier hat eine Spezialausbildung als Sportlehrer und unterrichtet deshalb alle Klassen im Fach Sport. Jede Klasse hat einen Klassensprecher und mindestens einen Schüler. Marie ist Klassensprecherin der vierten Klasse. Ihre Lieblingsfächer sind Sport, Malen und Mathematik.*

The elementary school of Freiburg has three employees: the two teachers Mr. Maier and Mrs. Schmidt, and the schoolmaster Mrs. Koster. In addition to their administrative duties, Mrs. Koster also does some teaching. In particular, Mr. Maier is assigned to the first-graders, while Mrs. Schmidt and Mrs. Koster together teach the second-, third-, and fourth-graders. Mr. Maier has specialized in sports and therefore is assigned to physical education for all four grades of school. Each grade has a class representative and at least one pupil. Actually, Marie is a fourth-grader. Her favourite subjects in school are physical education, painting, and mathematics.

Use URIs, Blank Nodes, Literals, and RDF containers in your RDF graph. Whenever it makes sense, also use the `rdfs` vocabulary, in particular `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain`, and `rdfs:range`. Finally list the facts that can be derived from your graph according to the RDFS semantics.

Im Folgenden eine Beispielmodellierung der Datenbank in nicht-graphischer Darstellung. Beim Vorrechnen empfiehlt es sich, Teile der Datenbank als Graph zu zeichnen, insbesondere D_i. Der Einfachheit halber verzichten wir auf die explizite Angabe von Namespaces für neu eingeführtes Vokabular.

D_m = { (ElementarySchool,rdf:type,rdfs:Class), (Employer,rdf:type,rdfs:Class),
(ElementarySchool,rdfs:subClassOf,Employer), (Teacher,rdf:type,rdfs:Class),
(Schoolmaster,rdf:type,rdfs:Class), (Pupil,rdf:type,rdfs:Class),
(Schoolmaster,rdfs:subClassOf,Teacher), (Teacher,rdfs:subClassOf,Person),
(Pupil,rdfs:subClassOf,Person), (SchoolSubject,rdf:type,rdfs:Class),
(Grade,rdf:type,rdfs:Class),
(name,rdf:type,rdf:Property), (name,rdfs:domain,Person), (name,rdfs:range,xsd:string),
(gender,rdf:type,rdf:Property), (name,rdfs:domain,Person), (gender,rdfs:range,xsd:string),
(works,rdf:type,rdf:Property), (works,rdfs:domain,Person), (works,rdfs:range,Employer),
(gradenr,rdf:type,rdf:Property), (gradenr,rdfs:domain,Grade), (gradenr,rdfs:range,xsd:integer),

(teaches,rdf:type,rdf:Property), (teaches,rdfs:domain,Teacher), (teaches,rdfs:range,Grade),
(teachesPhysEdu,rdf:type,rdf:Property), (teachesPhysEdu,rdfs:subPropertyOf,teaches),
(teachesPhysEdu,rdfs:domain,Teacher), (teachesPhysicalEdu,rdfs:range,Grade),
(visits,rdf:type,rdf:Property), (visits,rdfs:domain,Pupil), (visits,rdfs:range,Grade),
(represents,rdf:type,rdf:Property), (represents,rdfs:domain,Pupil), (represents,rdfs:range,Grade),
(represents,rdfs:subPropertyOf,visits), (favoriteSubjects,rdf:type,rdf:Property),
(favoriteSubjects,rdfs:domain,Pupil), (favoriteSubjects,rdfs:range,rdf:Bag) }

D_i = { (ESFr,rdf:type,ElementarySchool), (ESFr,rdfs:label,"Elementary School Freiburg"),
(T1,rdf:type,Teacher), (T1,name,"Maier"), (T1,gender,"male"), (T1,works,ESFr),
(T2,rdf:type,Teacher), (T2,name,"Schmidt"), (T2,gender,"female"), (T2,works,ESFr),
(T3,rdf:type,Schoolmaster), (T3,name,"Koster"), (T3,gender,"female"), (T3,works,ESFr),
(G1,rdf:type,Grade), (G1,gradenr,"1"), (G1,rdfs:label,"First-grader"),
(G2,rdf:type,Grade), (G2,gradenr,"2"), (G2,rdfs:label,"Second-grader"),
(G3,rdf:type,Grade), (G3,gradenr,"3"), (G3,rdfs:label,"Third-grader"),
(G4,rdf:type,Grade), (G4,gradenr,"4"), (G4,rdfs:label,"Fourth-grader"),
(T1,teaches,G1),
(T2,teaches,G2), (T2,teaches,G3), (T2,teaches,G4),
(T3,teaches,G2), (T3,teaches,G3), (T3,teaches,G4),
(T1,teachesPhysEdu,G1), (T1,teachesPhysEdu,G2), (T1,teachesPhysEdu,G3), (T1,teachesPhysEdu
(_:P1a,rdf:type,Pupil), (_:P1a,represents,G1), (_:P1b,rdf:type,Pupil), (_:P1b,visits,G1),
(_:P2a,rdf:type,Pupil), (_:P2a,represents,G2), (_:P2b,rdf:type,Pupil), (_:P2b,visits,G2),
(_:P3a,rdf:type,Pupil), (_:P3a,represents,G3), (_:P3b,rdf:type,Pupil), (_:P3b,visits,G3),
(_:P4a,rdf:type,Pupil), (_:P4a,represents,G4), (_:P4b,rdf:type,Pupil), (_:P4b,visits,G4),
(Marie,represents,G4),
(Marie,favoriteSubjects,_:SL), (_:SL,rdf:type,rdf:Bag),
(_:SL,rdf:_1,PhysEdu), (_:SL,rdf:_2,Painting), (_:SL,rdf:_3,Math),
(PhysEdu,rdf:type,SchoolSubject), (Painting,rdf:type,SchoolSubject), (Math,rdf:type,SchoolSubje

D_m definiert das benutzte Vokabular, während D_i die eigentliche Instanz definiert. In RDF sind diese "Schema"- und Instanz-Layers nicht getrennt. Die Gesamtmodellierung ergibt sich somit also als $D := D_m \cup D_i$. Selbstverständlich gibt es Spielraum bei der Modellierung. Einige Beispiele für neue Fakten, die mittels RDFS Semantik aus D hergeleitet werden können:

- (Schoolmaster,rdfs:subClassOf,Person)
- (_:P1a,visits,G1), (_:P2a,visits,G2), ...
- (Marie,visits,G4)
- (T1,teaches,G2), (T1,teaches,G3), (T1,teaches,G4)
- ...