Universität Freiburg
Institut für Informatik

Georges-Köhler Allee, Geb. 51
D-79110 Freiburg

Fang Wei-Kleiner

fwei@informatik.uni-freiburg.de

Advanced Databases and Information Systems
Summerterm 2019
Discussion on 23/05/2019

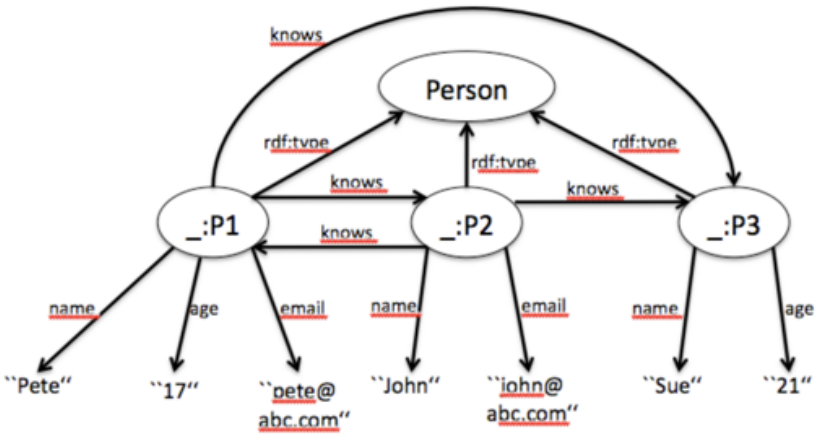
4. Sheet: Graph Databases

Exercise 1 (Graphs & 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 and 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.

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

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

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).

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

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

Exercise 2 (Graph Queries)
Consider the RDF & property graph databases from the previous exercise. Specify the following requests as SPARQL and Cypher queries, and indicate the final results obtained when evaluating them on the mentioned databases. You can, for example, test your queries with rdflib [1] by using `persons.n3` (available on ILIAS).

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@abc.com"*$\}\}$

c) 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:

d) All cyclic knows-relationships.

[1] `https://rdflib.readthedocs.io/en/stable/`

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

Ergebnis auf Datenbank D: