Apache Jena

- Home
- Download
- Learn
    - Tutorials
    - Overview
    - RDF core API tutorial
    - SPARQL tutorial
    - Manipulating SPARQL using ARQ
    - Using Jena with Eclipse
    - How-To's
    -
    - References
    - Overview
    - Javadoc
    - RDF API
    - RDF I/O
    - ARQ (SPARQL)
    - RDF Connection - SPARQL API
    - Elephas - tools for RDF on Hadoop
    - Text Search
    - TDB
    - SDB
    - SPARQL over JDBC
    - Fuseki
    - Permissions
    - Assembler
    - Ontology API
    - Inference API
    - Command-line tools
    - Extras
- Javadoc
    - Jena Core
    - ARQ
    - TDB
    - Fuseki
    - Elephas
    - Text Search
    - Spatial Search
    - Permissions
    - JDBC
    - All Javadoc
- Ask
- Get involved
    - Contribute
    - Report a bug
    -
    - Project
    - About Jena
    - Roadmap
    - Architecture
    - Project team

  - Related projects
  -
  - ASF
  - Apache Software Foundation
  - License
  - Thanks
  - Become a Sponsor
  - Security
- Improve this Page

1. TUTORIALS
2. SPARQL FILTERS

SPARQL Tutorial - Filters

Graph matching allows patterns in the graph to be found. This section describes how the values in a solution can be restricted. There are many comparisons available - we just cover two cases here.

String Matching

SPARQL provides an operation to test strings, based on regular expressions. This includes the ability to ask SQL "LIKE" style tests, although the syntax of the regular expression is different from SQL.

The syntax is:

```
FILTER regex(?x, "pattern" [, "flags"])
```

The flags argument is optional. The flag "i" means a case-insensitive pattern match is done.

The example query (q-f1.rq) finds given names with an "r" or "R" in them.

```
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>

SELECT ?g
WHERE
{ ?y vcard:Given ?g .
  FILTER regex(?g, "r", "i") }
```

with the results:

```
-------------
| g         |
=============
| "Rebecca" |
| "Sarah"   |
-------------
```

The regular expression language is the same as the XQuery regular expression language which is codified version of that found in Perl.

Testing Values

There are times when the application wants to filter on the value of a variable. In the data file vc-db-2.rdf, we have added an extra field for age. Age is not defined by the vCard schema so we have created a new property for the purpose of this tutorial. RDF allows such mixing of different definitions of information because URIs are unique. Note also that the `info:age` property value is typed.

In this extract of the data, we show the typed value. It can also be written plain 23.

```
<http://somewhere/RebeccaSmith/>
    info:age "23"^^xsd:integer ;
    vCard:FN "Becky Smith" ;
    vCard:N [ vCard:Family "Smith" ;
              vCard:Given  "Rebecca" ] .
```

So, a query (q-f2.rq) to find the names of people who are older than 24 is:

```
PREFIX info: <http://somewhere/peopleInfo#>

SELECT ?resource
WHERE
  {
    ?resource info:age ?age .
    FILTER (?age >= 24)
  }
```

The arithmetic expression must be in parentheses (round brackets). The only solution is:

```
---------------------------------
| resource                      |
=================================
| <http://somewhere/JohnSmith/> |
---------------------------------
```

Just one match, resulting in the resource URI for John Smith. Turning this round to ask for those less than 24 also yields one match for Rebecca Smith. Nothing about the Jones's.

The database contains no age information about the Jones: there are no info:age properties on these vCards so the variable `age` did not get a value and so was not tested by the filter.

Next: Optionals