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

SPARQL Tutorial - Optional Information

RDF is semi-structured data so SPARQL has a the ability to query for data but not to fail query when that data does not exist. The query is using an optional part to extend the information found in a query solution but to return the non-optional information anyway.

OPTIONALs

This query (q-opt1.rq) gets the name of a person and also their age if that piece of information is available.

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

SELECT ?name ?age
WHERE
{
    ?person vcard:FN  ?name .
    OPTIONAL { ?person info:age ?age }
}
```

Two of the four people in the data (vc-db-2.rdf)have age properties so two of the query solutions have that information. However, because the triple pattern for the age is optional, there is a pattern solution for the people who don't have age information.

```
-----------------------
| name          | age |
=======================
| "Becky Smith" | 23  |
| "Sarah Jones" |     |
| "John Smith"  | 25  |
| "Matt Jones"  |     |
-----------------------
```

If the optional clause had not been there, no age information would have been retrieved. If the triple pattern had been included but not optional then we would have the query (q-opt2.rq):

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

SELECT ?name ?age
WHERE
{
    ?person vcard:FN  ?name .
```

```
    ?person info:age ?age .
}
```

with only two solutions:

```
----------------------
| name          | age |
======================
| "Becky Smith" | 23  |
| "John Smith"  | 25  |
----------------------
```

because the `info:age` property must now be present in a solution.

OPTIONALs with FILTERs

`OPTIONAL` is a binary operator that combines two graph patterns. The optional pattern is any group pattern and may involve any SPARQL pattern types. If the group matches, the solution is extended, if not, the original solution is given (q-opt3.rq).

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

SELECT ?name ?age
WHERE
{
    ?person vcard:FN  ?name .
    OPTIONAL { ?person info:age ?age . FILTER ( ?age > 24 ) }
}
```

So, if we filter for ages greater than 24 in the optional part, we will still get 4 solutions (from the `vcard:FN` pattern) but only get ages if they pass the test.

```
----------------------
| name          | age |
======================
| "Becky Smith" |     |
| "Sarah Jones" |     |
| "John Smith"  | 25  |
| "Matt Jones"  |     |
----------------------
```

No age included for "Becky Smith" because it is less than 24.

If the filter condition is moved out of the optional part, then it can influence the number of solutions but it may be necessary to make the filter more complicated to allow for variable `age` being unbound (q-opt4.rq).

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

SELECT ?name ?age
WHERE
{
    ?person vcard:FN  ?name .
    OPTIONAL { ?person info:age ?age . }
    FILTER ( !bound(?age) || ?age > 24 )
}
```

If a solution has an `age` variable, then it must be greater than 24. It can also be unbound. There are now three solutions:

```
----------------------
| name          | age |
======================
| "Sarah Jones" |     |
| "John Smith"  | 25  |
| "Matt Jones"  |     |
----------------------
```

Evaluating an expression which has an unbound variables where a bound one was expected causes an evaluation exception and the whole expression fails.

OPTIONALs and Order Dependent Queries

One thing to be careful of is using the same variable in two or more optional clauses (and not in some basic pattern as well):

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>

SELECT ?name
WHERE
{
  ?x a foaf:Person .
  OPTIONAL { ?x foaf:name ?name }
  OPTIONAL { ?x vCard:FN  ?name }
}
```

If the first optional binds `?name` and `?x` to some values, the second `OPTIONAL` is an attempt to match the ground triples (`?x` and `<kbd>?name</kbd>` have values). If the first optional did not match the optional part, then the second one is an attempt to match its triple with two variables.

Next: union queries