SPARQL Tutorial

From https://jena.apache.org/tutorials/sparql.html

More detailed SPARQL Specification with examples:
https://www.w3.org/2001/sw/DataAccess/rq23/

Sample RDF Data

The file `vc-db-1.rdf` contains RDF for a number of vCard descriptions of people; The graph notation is shown below:

```
@prefix vCard:   <http://www.w3.org/2001/vcard-rdf/3.0#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix :        <#> .

<http://somewhere/MattJones/>
    vCard:FN    "Matt Jones" ;
    vCard:N     [ vCard:Family
                              "Jones" ;
                  vCard:Given
                              "Matthew"
                ] .

<http://somewhere/RebeccaSmith/>
    vCard:FN    "Becky Smith" ;
    vCard:N     [ vCard:Family
                              "Smith" ;
                  vCard:Given
                              "Rebecca"
                ] .

<http://somewhere/JohnSmith/>
    vCard:FN    "John Smith" ;
    vCard:N     [ vCard:Family
                              "Smith" ;
                  vCard:Given
                              "John"
                ] .

<http://somewhere/SarahJones/>
    vCard:FN    "Sarah Jones" ;
    vCard:N     [ vCard:Family
                              "Jones" ;
                  vCard:Given
                              "Sarah"
                ] .
```

or even more explicitly as triples:

```
@prefix vCard:   <http://www.w3.org/2001/vcard-rdf/3.0#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<http://somewhere/MattJones/>  vCard:FN   "Matt Jones" .
<http://somewhere/MattJones/>  vCard:N    _:b0 .
_:b0  vCard:Family "Jones" .
_:b0  vCard:Given  "Matthew" .


<http://somewhere/RebeccaSmith/> vCard:FN    "Becky Smith" .
<http://somewhere/RebeccaSmith/> vCard:N     _:b1 .
_:b1 vCard:Family "Smith" .
_:b1 vCard:Given  "Rebecca" .

<http://somewhere/JohnSmith/>    vCard:FN    "John Smith" .
<http://somewhere/JohnSmith/>    vCard:N     _:b2 .
_:b2 vCard:Family "Smith" .
_:b2 vCard:Given  "John"  .

<http://somewhere/SarahJones/>   vCard:FN    "Sarah Jones" .
<http://somewhere/SarahJones/>   vCard:N     _:b3 .
_:b3 vCard:Family  "Jones" .
_:b3 vCard:Given   "Sarah" .
```

Now on to some queries:

Query 1 `q1.rq`

```
SELECT ?x
WHERE { ?x  <http://www.w3.org/2001/vcard-rdf/3.0#FN>  "John Smith" }
```

```
[raj@tinman sparql]$ sparql --data=vc-db-1.rdf --query=q1.rq
---------------------------------
| x                             |
=================================
| <http://somewhere/JohnSmith/> |
---------------------------------
```

Query 2 `q-bp1.rq`

```
SELECT ?x ?fname
WHERE {?x  <http://www.w3.org/2001/vcard-rdf/3.0#FN>  ?fname}
```

```
[raj@tinman sparql]$ sparql --data=vc-db-1.rdf --query=q-bp1.rq
-------------------------------------------------
| x                                | fname        |
=================================================
| <http://somewhere/JohnSmith/>    | "John Smith"  |
| <http://somewhere/SarahJones/>   | "Sarah Jones" |
| <http://somewhere/MattJones/>    | "Matt Jones"  |
| <http://somewhere/RebeccaSmith/> | "Becky Smith" |
-------------------------------------------------
```

Query 3: Basic Patterns `q-bp2.rq`

```
SELECT ?givenName
WHERE
  { ?y  <http://www.w3.org/2001/vcard-rdf/3.0#Family>  "Smith" .
    ?y  <http://www.w3.org/2001/vcard-rdf/3.0#Given>  ?givenName .
  }
```

```
[raj@tinman sparql]$ sparql --data=vc-db-1.rdf --query=q-bp2.rq
-------------
| givenName |
=============
| "John"    |
| "Rebecca" |
-------------
```

QNames (short hand mechanism) `q-bp3.rq`

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

SELECT ?givenName
WHERE
 { ?y vcard:Family "Smith" .
   ?y vcard:Given  ?givenName .
 }
```

```
[raj@tinman sparql]$ sparql --data=vc-db-1.rdf --query=q-bp2.rq
-------------
| givenName |
=============
| "John"    |
| "Rebecca" |
-------------
```

Blank Nodes `q-bp4.rq`

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

SELECT ?y ?givenName
WHERE
 { ?y vcard:Family "Smith" .
   ?y vcard:Given  ?givenName .
 }
```

```
 [raj@tinman sparql]$ sparql --data=vc-db-1.rdf --query=q-bp4.rq
--------------------
| y    | givenName |
====================
| _:b0 | "Rebecca" |
| _:b1 | "John"    |
--------------------
```

Filters (String Pattern Matching) `q-f1.rq`

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

Here, flags is optional (if present it can be "i" for case insensitive match)

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

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

find given names that have a "r" or "R" in them.

```
[raj@tinman sparql]$ sparql --data=vc-db-1.rdf --query=q-f1.rq
-------------
| g         |
=============
| "Rebecca" |
| "Sarah"   |
-------------
```

Testing Values `q-f2.rq`

Let us add a new property, `age` to the data. RDF file is `vc-db-2.rdf`. A part of the new data is shown below.

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

The following SPARQL query:

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

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

produces the following answer:

```
[raj@tinman sparql]$ sparql --data=vc-db-2.rdf --query=q-f2.rq
---------------------------------
| resource                      |
=================================
| <http://somewhere/JohnSmith/> |
---------------------------------
```

Optional Information

RDF is semi-structured data; So, SPARQL has the ability to query for data but not to fail the

query if data does not exist.

Gets the name of a person and also their age if that piece of information is available. `q-opt1.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 }
}
```

```
[raj@tinman sparql]$ sparql --data=vc-db-2.rdf --query=q-opt1.rq
-----------------------
| name          | age |
=======================
| "John Smith"  | 25  |
| "Sarah Jones" |     |
| "Matt Jones"  |     |
| "Becky Smith" | 23  |
-----------------------
```

without the OPTIONAL keyword `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 .
}
```

```
[raj@tinman sparql]$ sparql --data=vc-db-2.rdf --query=q-opt2.rq
-----------------------
| name          | age |
=======================
| "John Smith"  | 25  |
| "Becky Smith" | 23  |
-----------------------
```

OPTIONAL with FILTERs `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 ) }
}
```

```
[raj@tinman sparql]$ sparql --data=vc-db-2.rdf --query=q-opt3.rq
-----------------------
| name          | age |
=======================
| "John Smith"  | 25  |
| "Sarah Jones" |     |
| "Matt Jones"  |     |
| "Becky Smith" |     |
-----------------------
```

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

Move FILTER out! `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 )
}
```

```
[raj@tinman sparql]$ sparql --data=vc-db-2.rdf --query=q-opt4.rq
-----------------------
| name          | age |
=======================
| "John Smith"  | 25  |
| "Sarah Jones" |     |
| "Matt Jones"  |     |
-----------------------
```

without !bound condition

will produce

```
[raj@tinman sparql]$ sparql --data=vc-db-2.rdf --query=q-opt4.rq
----------------------
| name         | age |
======================
| "John Smith" | 25  |
----------------------
```

ALTERNATIVES in a Pattern (UNION)

Consider vCard and FOAF (Friend of a Friend) ontologies. Both capture similar information. If you wanted to query combined data, you can use UNION.

Consider the following data that combines vCard and foaf entries (`vc-db-3.rdf`):

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

_:a rdf:type foaf:Person .

_:b rdf:type foaf:Person .

_:c rdf:type foaf:Person .

_:d rdf:type foaf:Person .

_:a foaf:name   "Matt Jones" .

_:b foaf:name   "Sarah Jones" .

_:c vcard:FN    "Becky Smith" .

_:d vcard:FN    "John Smith" .
```

The following SPARQL query (`q-union1.rq`) will retrieve names from both namespaces:

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

SELECT ?name
WHERE
{
   { [] foaf:name ?name } UNION { [] vCard:FN ?name }
}
```

Note: [] is abbreviation for a blank node that is used exactly once.

```
[raj@tinman sparql]$ sparql --data=vc-db-3.rdf --query=q-union1.rq
-----------------
| name          |
=================
| "Sarah Jones" |
| "Matt Jones"  |
| "John Smith"  |
| "Becky Smith" |
-----------------
```

alternative way (`q-unionalt.rq`)

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

SELECT ?name
WHERE
{
  [] ?p ?name
  FILTER ( ?p = foaf:name || ?p = vCard:FN )
}
```

Union - remembering where data came from (`q-union2.rq`)

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

SELECT ?name1 ?name2
WHERE
{
   { [] foaf:name ?name1 } UNION { [] vCard:FN ?name2 }
}
```

```
---------------------------------
| name1         | name2         |
=================================
| "Matt Jones"  |               |
| "Sarah Jones" |               |
|               | "Becky Smith" |
|               | "John Smith"  |
---------------------------------
```

OPTIONAL vs UNION

In practice, OPTIONAL is more common than UNION but they both have their uses.

- OPTIONAL are useful for augmenting the solutions found.
- UNION is useful for concatenating the solutions from two possibilities.

`q-union3.rq`

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

SELECT ?name1 ?name2
WHERE
{
  ?x a foaf:Person
  OPTIONAL { ?x  foaf:name  ?name1 }
  OPTIONAL { ?x  vCard:FN   ?name2 }
}
```

```
[raj@tinman sparql]$ sparql --data=vc-db-3.rdf --query=q-union3.rq
---------------------------------
| name1         | name2         |
=================================
|               | "Becky Smith" |
| "Sarah Jones" |               |
| "Matt Jones"  |               |
|               | "John Smith"  |
---------------------------------
```

NAMED GRAPHS

skip

RESULTS

skip