THE UNIVERSITY of EDINBURGH
informatics

Semantic Web Systems

Querying

Jacques Fleuriot

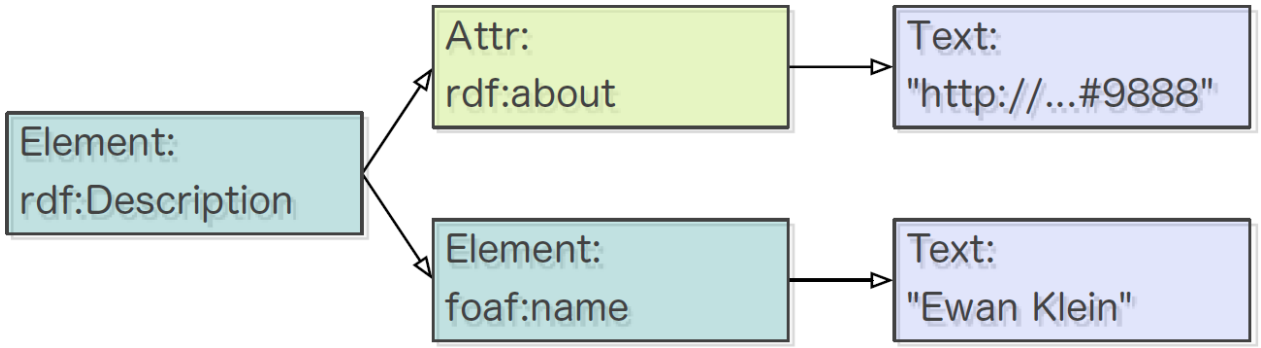
School of Informatics

In the previous lecture

- Serialising RDF in XML

RDF Triples with literal Object

edstaff:9888 foaf:name 'Ewan Klein' .



Linear version

```
<rdf:Description rdf:about="http://...#9888">
  <foaf:name>Ewan Klein</foaf:name>
</rdf:Description>
```

In the previous lecture

- RDFS & inference

Subclasses

terms:Giraffe rdfs:subClassOf terms:Herbivore .

Domain

terms:eats rdfs:domain terms:Animal .

Range

terms:eats rdfs:range terms:Plant .

In this lecture

- Querying with XML
- Querying with SPARQL

Querying with XML

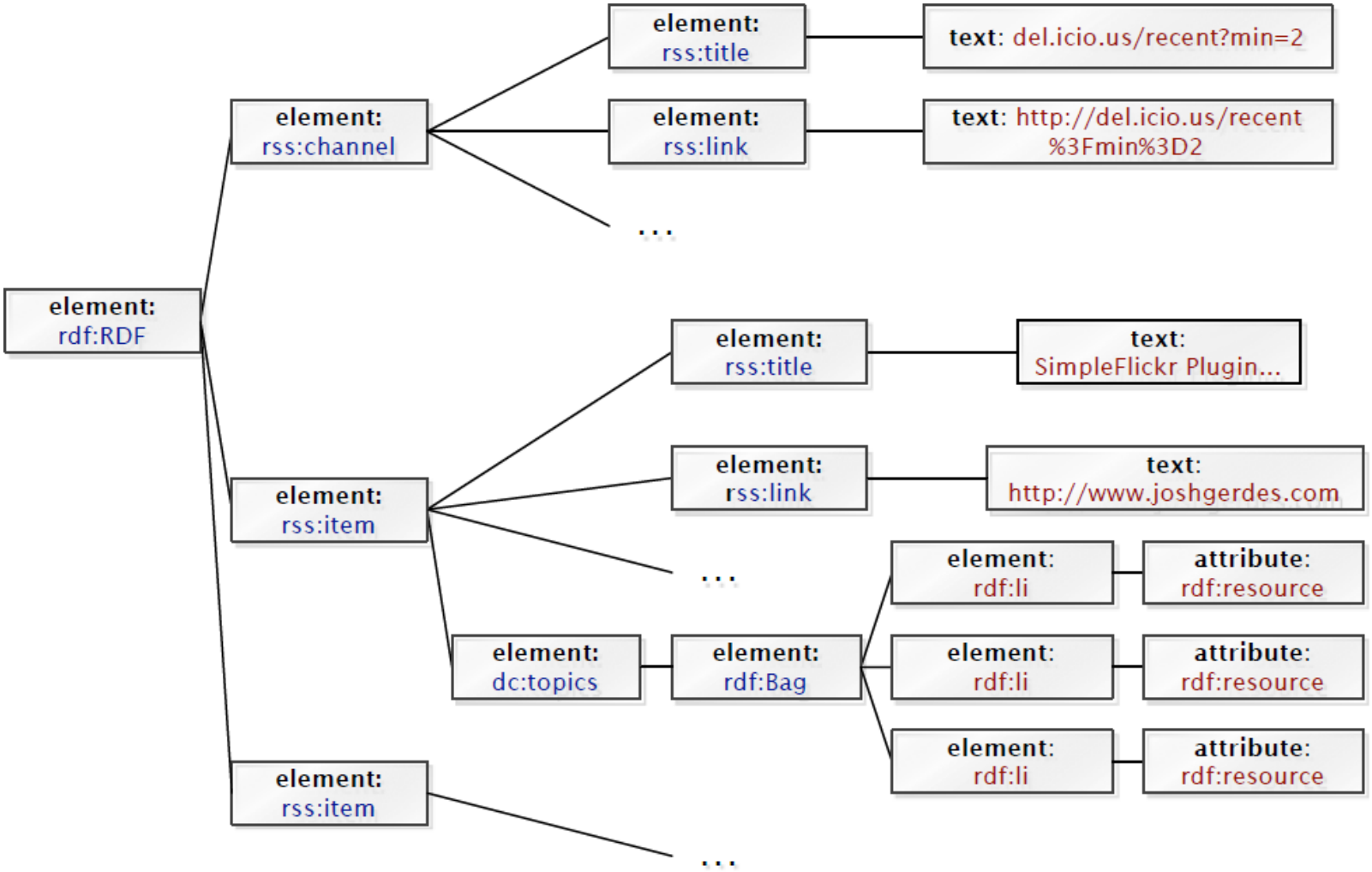
Storing RDF

- flat file – easy, but doesn't scale.
- **Triplestore** - database that is customised for storing triples:
  - Large triplestores measured in terms of billions.
  - See http://en.wikipedia.org/wiki/Triplestore for list of current technologies.
  - We used to have a local data repository http://data.inf.ed.ac.uk based on 4store (http://www.4store.org).

Querying RDF Data

- Querying is crucial to being able to use RDF data.
- Allows data across different data repositories to be combined.
- Allows selected data to be re-used and re-presented.
- Compare XML and SPARQL.

Node Tree for a del.icio.us RSS Feed



Pick a path through XML tree

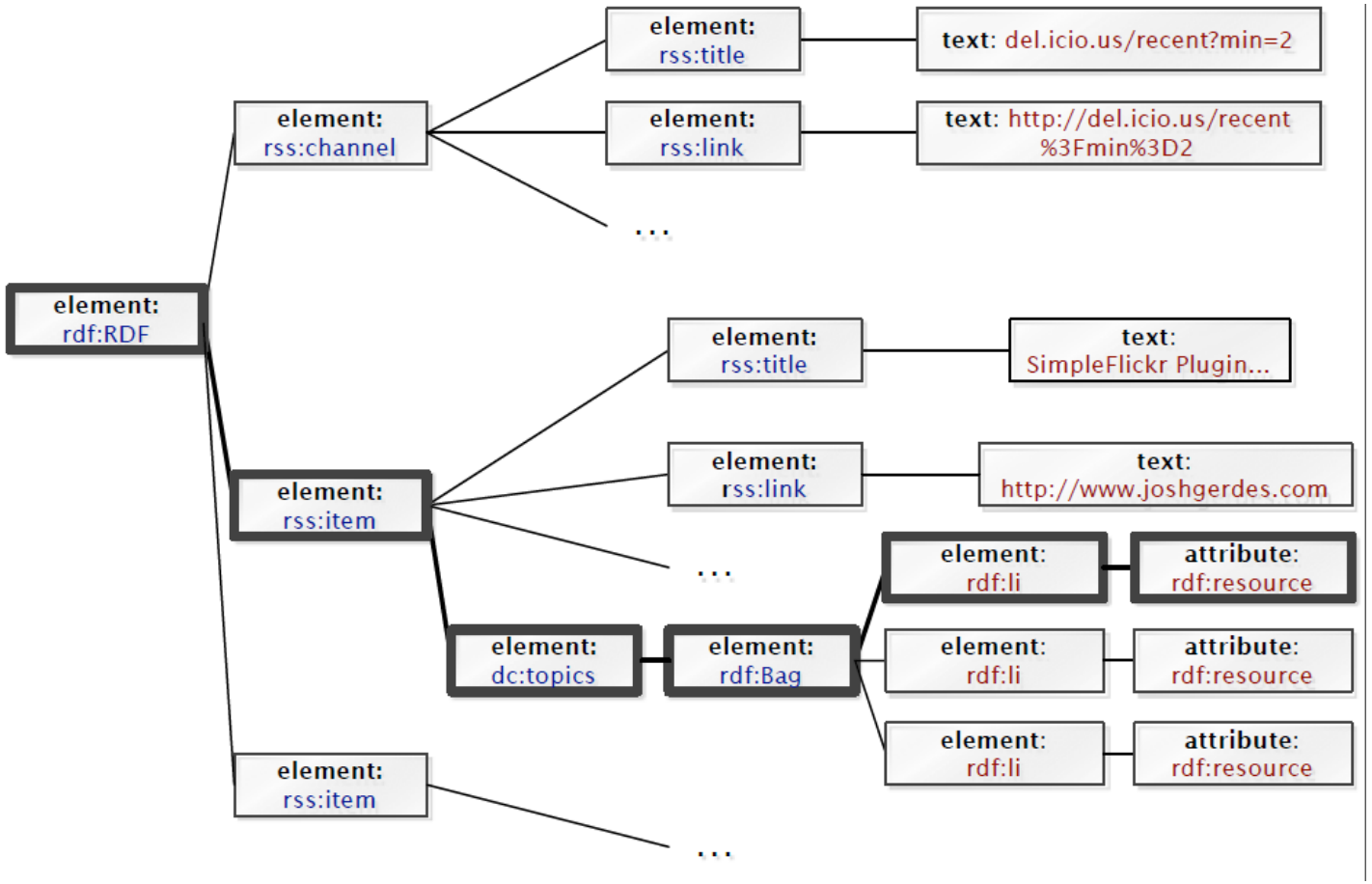
A path expression:

/rdf:RDF/rss:item/dc:topics/rdf:Bag/rdf:li

Reading from right to left:

- Select every node with tag rdf:li
- that's the direct child of an rdf:Bag element
- that's the direct child of a dc:topics element
- that's the direct child of an rss:item element
- that's the direct child of the rdf:RDF element
- at the root of the document ('/').

The path through the node tree



Remarks on XML query

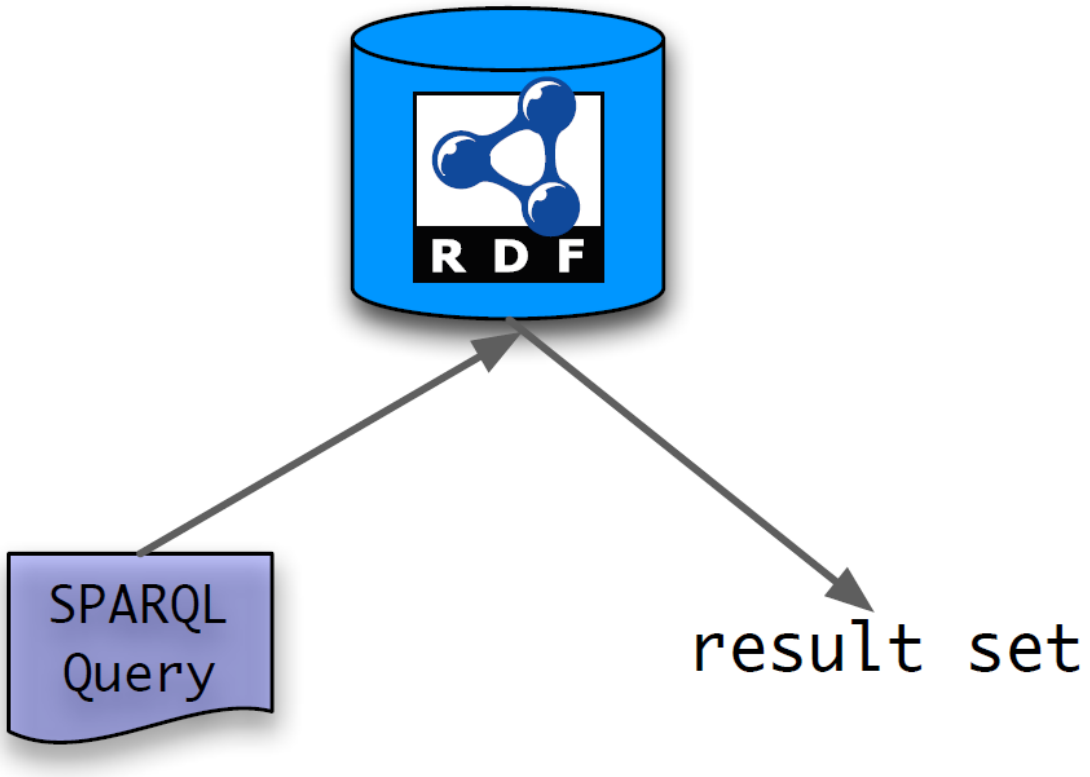
- Basic idea: search along paths in an XML tree.
- XPath provides an exact syntax for this.
- Most modern programming languages provide rich support for XML parsing and querying.
- Requires you know how data is encoded in the XML document.

Querying with SPARQL

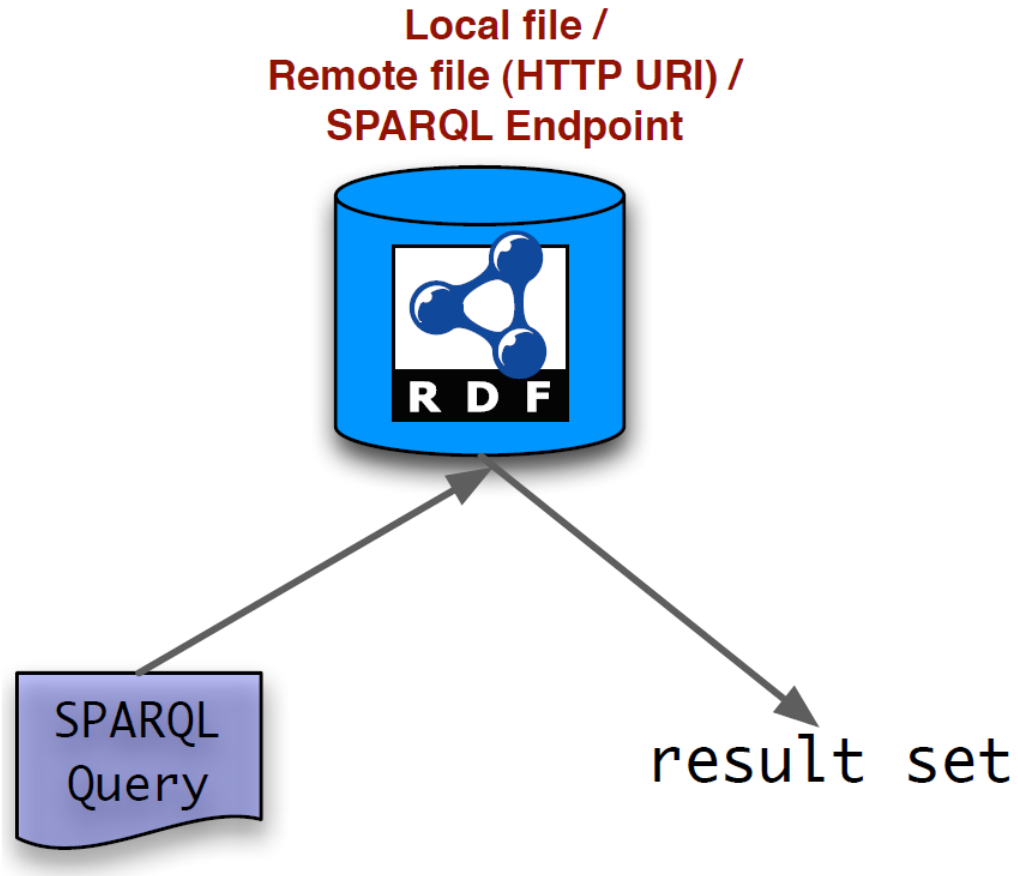
RDF Query

- Make use of the **graph model**, not the (XML) **serialisation**.
- At least half a dozen competing proposals over approx 10 year period:
  - Path-based query.
  - SQL-like pattern matching.
- SPARQL is now the standard approach, and is 2nd type of language.

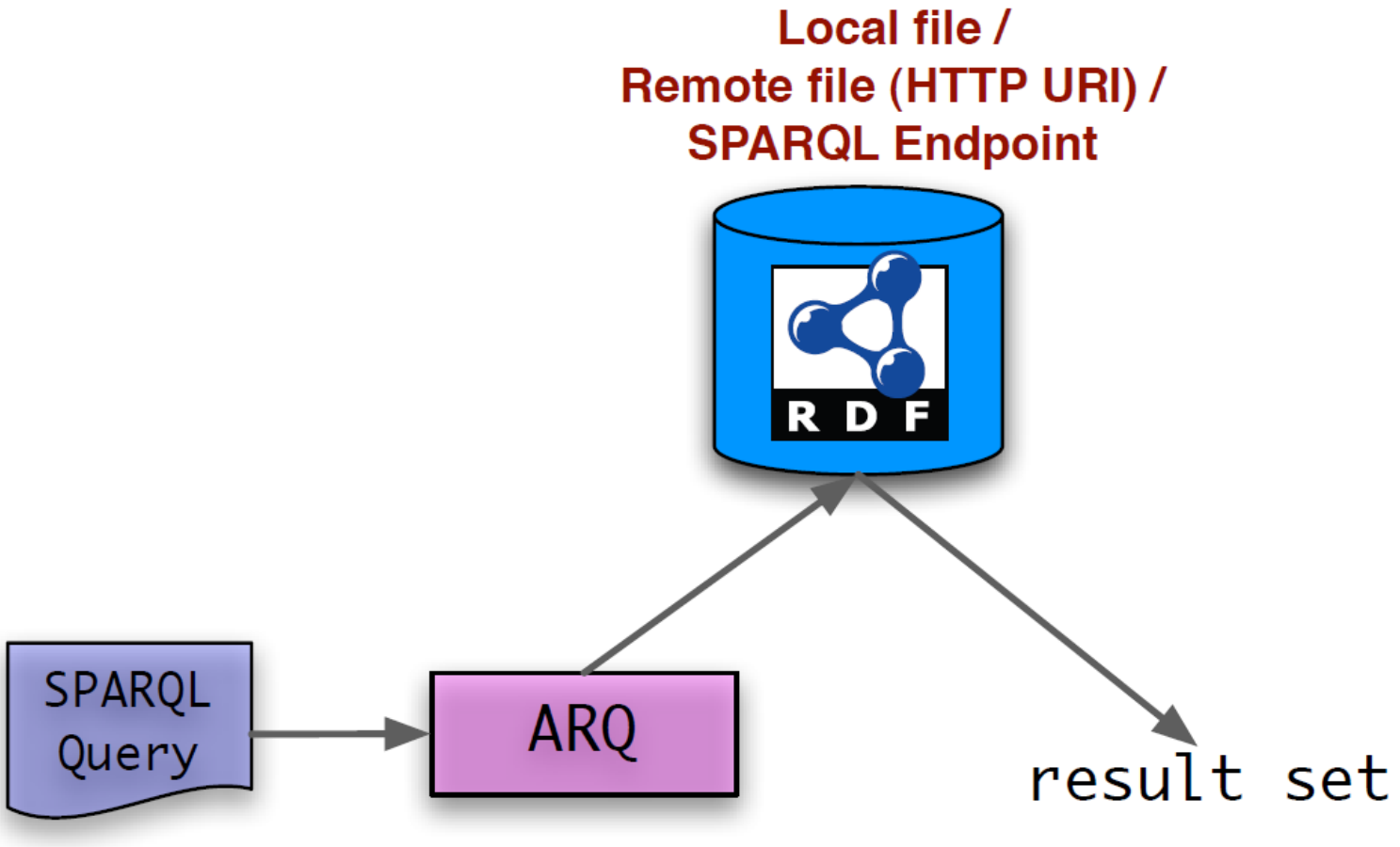
SPARQL overview



SPARQL overview



SPARQL overview



SPARQL query example

Query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE { ?x   foaf:name   ?name . }
```

SPARQL components

1. Basic graph pattern matching
2. Graph expressions
3. Solution modifiers

Basic graph patterns (BGPs)

- Basic graph patterns are a block of adjacent triple patterns that match, or don't match, all together.
- Every named variable must receive some value for the pattern to have been matched.
- Filters add restrictions on the values variables can take.

Graph expressions

- Graph expressions:
  - OPTIONAL, UNION, GRAPH
- **Combine** BGPs in various ways to give more complicated patterns.
- Graph expressions are **recursive** so you can have patterns within patterns.
- A SPARQL query has one graph expression in the WHERE clause.

Solution modifiers

- Solution modifiers:
  - DISTINCT, ORDER BY, LIMIT/OFFSET
- Apply to output of matching the query graph pattern;
- Process it in various ways to yield the result set.

Patterns

- Based on triples.
- Combination of URIs/QNames, literals and variables.
- Variable names:
  - ?var, $var
  - cannot start with an integer, or contain ':', '-' or '.'.

Small FOAF file

FOAF

```
@prefix  :  <http://inf.ed.ac.uk/ont#> .
@prefix  foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix  rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

:ehk  a  foaf:Person ;
        foaf:mbox_sha1sum  "e9403..." ;
        foaf:name  "Ewan Klein" ;
        foaf:knows  [  a  foaf:Person ;
           rdfs:seeAlso  <http://www.ibiblio.org/hhalpin/foaf.rdf>;
           foaf:mbox_sha1sum  "c5e75..." ;
           foaf:name  "Harry Halpin" ] .
```

SELECT example

Query example-00.rq

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE { ?x   foaf:name   ?name . }
```

Running the query

```
arq -- query=example-00.rq \
      -- data=http://homepages.inf.ed.ac.uk/ewan/foaf.n3
```

- Run query against the RDF data to be found at URI.
- URI has to be addressable via HTTP when the query is executed.

Result set

```
--------------------
|  name             |
==============
| ”Ewan Klein”     |

| ”Harry Halpin”   |

----------------------
```

Matching

Query matches the graph:

- find a set of variable ↦ value bindings, such that
- result of replacing variables by values is a triple in the graph.

Solution1:

variable ?x has value blank node _:a and variable ?name has value ”Ewan”
Triple (_:a foaf:name ”Ewan Klein”) is in the graph.

Solution 2:

variable ?x has value blank node _:b and variable ?name has value ”Harry”
Triple (_:b foaf:name ”Harry Halpin”) is in the graph.

Multiple patterns

Query example-01.rq

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name1  ?name2
FROM <http://homepages.inf.ed.ac.uk/ewan/foaf.n3>
WHERE {
   ?person1   foaf:knows  ?person2 .
   ?person1   foaf:name   ?name1 .
   ?person2   foaf:name   ?name2 .
  }
```

- Get name1 and name2 where person1 knows person2, and person1 has name1 and person2 has name2
- Dots '.' separate patterns in the query.

Running the query

```
arq -- query=example-01.rq
```

Result set

```
-----------------------------------------
| name1          | name2                |
=========================================

| ”Ewan Klein”   | ”Harry Halpin”       |

-----------------------------------------
```

More on patterns

- Can use N3 abbreviated syntax in Basic Graph Patterns

Abbreviated Version of example-01.rq

```
…
WHERE {
        [ foaf:knows  [ foaf:name  ?name2 ] ;
          foaf:name  ?name1 ] .
 }
```

Multiple data sources

Query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name1  ?name2
FROM <http://homepages.inf.ed.ac.uk/ewan/foaf.n3>
FROM <http://www.ibiblio.org/hhalpin/foaf.rdf>
WHERE {
   ?person1   foaf:name   ?name1 ;
              foaf:knows  [  foaf:name   ?name2 ];
  }
```

NB: Multiples FROM clauses allowed in a query.

Multiple data sources

name1	name2
”Harry Halpin”	”Daniel Weitzner”
”Harry Halpin”	”Tim Berners-Lee”
”Harry Halpin”	”Dan Connolly”
”Harry Halpin”	”Ian Davis”
”Harry Halpin”	”Paolo Bouquet”
”Ewan Klein”	”Harry Halpin”

Multiple data sources

Query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name1  ?name2
WHERE {
   ?person1   foaf:name   ?name1 ;
              foaf:knows  [  foaf:name   ?name2 ];
  }
```

Running the query

```
arq -- query=example-00.rq \
     -- data=http://homepages.inf.ed.ac.uk/ewan/foaf.n3 \
     -- data=http://www.ibiblio.org/hhalpin/foaf.rdf
```

- Create models from the data sets;
- merge the models;
- run query against model.

SPARQL query forms

SELECT: Like SQL; returns a tuple of values.

Also:

CONSTRUCT: Builds a new graph by inserting values into a triple pattern.

ASK: Asks whether a query has a solution in a graph.

Conclusions

- XML-based query depends on paths through the node tree.
- SPARQL matches triple patterns in the RDF graph.
- XML-based query depends on knowing the syntactic structure of the serialisation.
  - There are different but equivalent serialisations of RDF in XML;
  - these would require different XML queries.
- SPARQL query depends on knowing the graph structure of the RDF store.
  - There are different but equivalent serialisations of RDF (in XML, N3, ...);
  - these can all be matched using the same SPARQL query.

Reading

- SWWO Ch5

Task

Using your RDF dataset, formulate some SPAQRL queries which could be run on it. What would the outcome be?