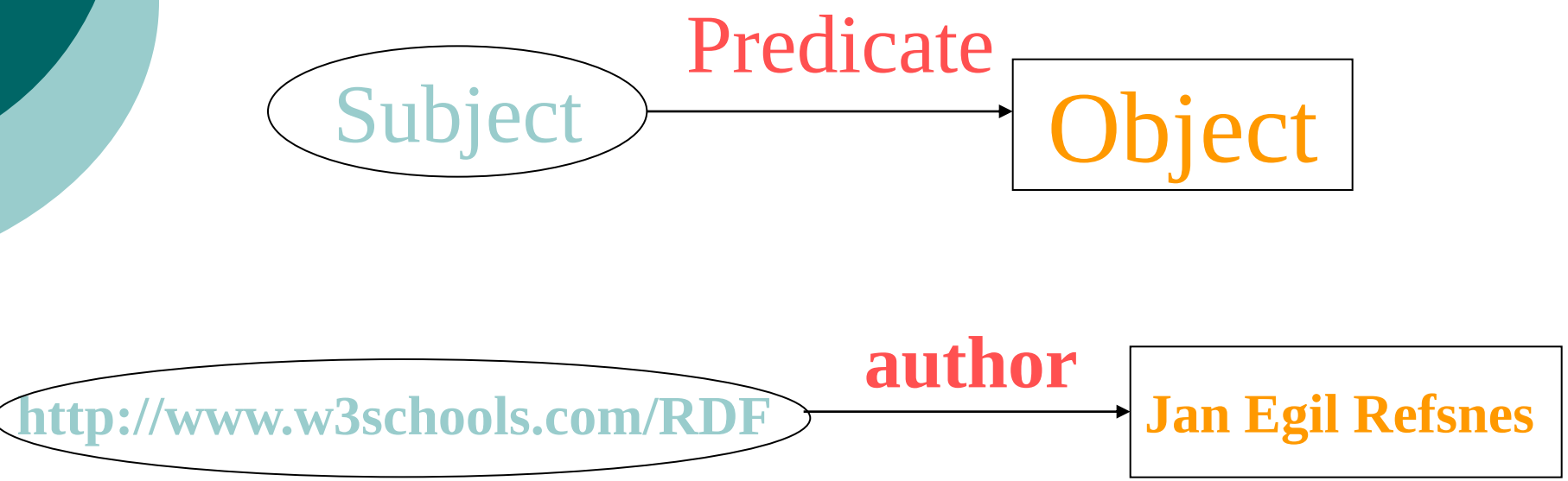
SPARQL:
A query language for RDF

Matthew Yau
C.Y.Yau@warwick.ac.uk

What is SPARQL

- RDF is a format for representing general data about resources. RDF is based on a graph, where subject and object nodes are related by predicate arcs. RDF can be written in XML, or as triples.
- RDF schema is a method for defining structures for RDF files. It allows RDF resources to be grouped into classes, and allows subclass, subproperty and domain/range descriptions to be specified.
- SPARQL is a query language for RDF. It provides a standard format for writing queries that target RDF data and a set of standard rules for processing those queries and returning the results.

RDF Statements



?subject ?predicate ?object

SPARQL searches for all subgraphs that match the graph described by the triples in the query.

A sample of SPARQL

```
SELECT ?student
WHERE { ?student b:studies bmod:CS328 }
```

Prefixes & namespaces

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:cd="http://www.recshop.fake/cd#">

b:studies
bmod:CS328
PREFIX b: http://...

PREFIX bmod:
http://www2.warwick.ac.uk/fac/sci/dcs/teaching/material/

- The PREFIX keyword is SPARQL's version of an xmlns:namespace declaration and works in basically the same way.

SPARQL basics

- SPARQL is not based on XML, so it does not follow the syntax conventions seen before.
- Names beginning with a ? or a $ are variables.
- Graph patterns are given as a list of *triple* patterns enclosed within braces {}
- The variables named after the SELECT keyword are the variables that will be returned as results. (~SQL)

Combining conditions

```
PREFIX b: http://www2.warwick.ac.uk/rdf/
PREFIX bmod:
   http://www2.warwick.ac.uk/fac/sci/dcs/te
   aching/material/
PREFIX foaf:
http://xmlns.com/foaf/0.1/

SELECT ?name

WHERE { ?student b:studiesbmod:CS328
?student foaf:name ?name }
```

FOAF

- FOAF (Friend Of A Friend) is an experimental project using RDF, which also defines a standardised vocabulary.
- The goal of FOAF is to make personal home pages machine-readable, and machine-understandable, thereby creating an internet-wide connected database of people.

Friend-of-a-friend:
http://xmlns.com/foaf/0.1/

FOAF

- FOAF is based on the idea that most personal home pages contain similar sets of information.
- For example, the name of a person, the place they live, the place they work, details on what they are working on at the moment, and links to their friends.
- FOAF defines RDF predicates that can be used to represent these things. These pages can then be understood by a computer and manipulated.
- In this way, a database can be created to answer questions such as “what projects are my friends working on?”, “do any of my friends know the director of BigCorp?” and similar.

A sample FOAF document

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:foaf="http://xmlns.com/foaf/0.1">
<foaf:Person>
<foaf:name>Matthew Yau</foaf:name>
<foaf:mbox
rdf:resource="mailto:C.Y.Yau@warwick.ac.uk " />
</foaf:Person>
</rdf:RDF>
```

FOAF

- You can see that FOAF stores details about a person's name, workplace, school, and people they know.
- Note that the meaning of “knows” is deliberately ambiguous. In spite of the “friend of a friend” title, the presence of a “knows” relation does not mean that the people are friends.
- Likewise, it does not mean that the relationship is reciprocal. Equally, the lack of a “knows” relation does not mean the people do not know each other Thus, unless extra rules are applied, it is not possible to check for reciprocation by looking for a knows relation from the other person.

Multiple results

```
PREFIX b: <
   http://www2.warwick.ac.uk/rdf/>
PREFIXbmod: <
   http://www2.warwick.ac.uk/fac/sci/dcs/teaching/material/ >
PREFIX foaf:
http://xmlns.com/foaf/0.1/

SELECT ?module ?name
WHERE { ?student b:studies ?module .
?student foaf:name ?name }
```

Abbreviating the same subject

```
PREFIX b: <
   http://www2.warwick.ac.uk/rdf/>
PREFIXbmod: <
   http://www2.warwick.ac.uk/fac/sci/dcs/teaching/material/ >
PREFIX foaf:
http://xmlns.com/foaf/0.1/

SELECT ?module ?name
WHERE { ?student b:studies ?module ;
foaf:name ?name }
```

Abbreviating multiple objects

```
SELECT ?module ?name
WHERE { ?student b:studies ?module .
?student b:studies bmod:CS328 ;
foaf:name ?name }
```

is identical to

```
SELECT ?module ?name
WHERE { ?student b:studies ?module ,
                           bmod:CS328 ;
foaf:name ?name }
```

Optional graph components

```
SELECT ?student ?email
WHERE { ?student b:studies mod:CS328 .
?student foaf:mbox ?email }
```

- The above query returns the names and e-mail addresses of students studying CS328. However, if a student does not have an e-mail address registered, with a foaf:mbox predicate, then the query will not match and that student will not be included in the list. This may be undesirable.

Optional graph components

```
SELECT ?student ?email
WHERE { ?student b:studies mod:CS328 .
OPTIONAL { ?student foaf:mbox ?email } }
```

- Because the email match is declared as OPTIONAL in the query above, SPARQL will match it if it can, but if it can't, it will not reject the overall pattern because of it. So if a student has no registered e-mail address, they will still appear in the result list with a blank (unbound) value for the ?email result.

Optional graph components

```
SELECT ?module ?name ?phone
WHERE { ?student b:studies ?module .
?student foaf:name ?name .
OPTIONAL {
?student b:contactpermission true .
?student b:phone ?phone
} }

SELECT ?module ?name ?age
WHERE { ?student b:studies ?module .
?student foaf:name ?name .
OPTIONAL { ?student b:age ?age .
FILTER (?age > 25) } }
```

Further optional components

```
SELECT ?student ?email ?home
WHERE { ?student b:studies mod:CS328 .
OPTIONAL { ?student foaf:mbox ?email .
?student foaf:homepage ?home } }
```

```
SELECT ?student ?email ?home
WHERE { ?student b:studies mod:CS328 .
OPTIONAL { ?student foaf:mbox ?email } .
OPTIONAL { ?student
   foaf:homepage ?home }
}
```

Combining matches

```
SELECT ?student ?email
WHERE { ?student foaf:mbox ?email .
{ ?student b:studies mod:CS328 }
UNION { ?student b:studies mod:CS909 }
}
```

- When patterns are combined using the UNION keyword, the resulting combined pattern will match if either of the subpatterns is matched.

Multiple graphs and the dataset

- All the queries we have seen so far have operated on single RDF graphs.
- A SPARQL query actually runs on an RDF dataset which may include multiple RDF graphs.
- RDF graphs are identified, like everything else, by URI. As with other resources, the URI that represents the graph does not have to be the actual URI of the graph file, although the program processing the query will need to somehow relate the URI to an actual RDF graph stored somewhere.

Stating the dataset

```
SELECT ?student ?email ?home
FROM <http://www2.warwick.ac.uk/rdf/student>
WHERE { ?student b:studies mod:CS909 .
OPTIONAL { ?student foaf:mbox?email .
?student foaf:homepage ?home } }
```

By using several FROM declarations, you can combine several graphs in the dataset:

```
SELECT ?student ?email ?home
FROM <http://www2.warwick.ac.uk/rdf/student>
FROM <http://www2.warwick.ac.uk/rdf/foaf>
WHERE { ?student b:studies mod:CS909 .
OPTIONAL { ?student foaf:mbox?email .
?student foaf:homepage ?home } }
```

Multiple graphs

- The dataset can be composed of one (optional) default graph and any number of named graphs.
- The FROM keyword specifies the default graph. If you use several FROM keywords, the specified graphs are merged to create the default graph.
- You can also add graphs to the dataset as named graphs, using the FROM NAMED keyword.
- However, to match patterns in a named graph you must use the GRAPH keyword to explicitly state which graph they must match in.

Named graphs

```
SELECT ?student ?email ?home
FROM NAMED
<http://www2.warwick.ac.uk/rdf/student>
FROM NAMED
http://www2.warwick.ac.uk/rdf/foaf

WHERE {
GRAPH <http://www2.warwick.ac.uk/rdf/student>
   { ?student b:studies mod:CS909 } .
GRAPH <http://www2.warwick.ac.uk/rdf/foaf>
{
OPTIONAL { ?student foaf:mbox ?email .
?student foaf:homepage ?home }
}
}
```

Abbreviation using prefixes

```
PREFIX brdf: <http://www2.warwick.ac.uk/rdf/>
SELECT ?student ?email ?home
FROM NAMED
<http://www2.warwick.ac.uk/rdf/student>
FROM NAMED
<http://www2.warwick.ac.uk/rdf/foaf>

WHERE {
GRAPH brdf:student { ?student b:studies
   mod:CS909 } .
GRAPH brdf:foaf{
OPTIONAL { ?student foaf:mbox ?email .
?student foaf:homepage ?home }
}
}
```

Named and default graph

```
PREFIX brdf: <http://www2.warwickac.uk/rdf/>
SELECT ?student ?email ?home
FROM <http://www2.warwickac.uk/rdf/student>
FROM NAMED
<http://www2.warwickac.uk/rdf/foaf>

WHERE {
?student b:studies mod:CS909 .
GRAPH brdf:foaf{
OPTIONAL { ?studentfoaf:mbox?email .
?studentfoaf:homepage ?home }
}
}
```

Graph as a query

- As well as being a bound resource, the parameter of the GRAPH keyword can also be a variable.
- By making use of this, it is possible to query which graph in the dataset holds a particular relationship, or to determine which graph to search based on data in another graph.
- It is not mandatory that all graphs referenced by a SPARQL query be declared using FROM and FROM NAMED. It need not specify any at all, and even if specified, the dataset can be overridden on a per-query basis.

Which graph is it in?

```
PREFIX brdf: <http://www2.warwickac.uk/rdf/>
SELECT ?student ?graph
WHERE {
?student b:studies mod:CS909 .
GRAPH ?graph {
?student foaf:mbox ?email
}
}
```

The output variable graph will hold the URL of the graph which matches the student to an e-mail address. Note that we presume that the query processor will have existing knowledge of some finite set of graphs and their locations, through which it will search.

Re-using the graph reference

```
PREFIXbrdf: <http://www2.warwickac.uk/rdf/>
SELECT ?student ?email
WHERE {
?student b:studies mod:CS909 .
?student rdfs:seeAlso ?graph .
GRAPH ?graph {
?student foaf:mbox ?email
}
}
```

- In this case we collect the graph URL from the rdfs:seeAlso property of the student, and then look in that graph for their e-mail address. Note that if the student does not have a rdfs:seeAlso property which points to a graph holding their e-mail address, they will not appear in the result at all.

Sorting results of a query

```
SELECT ?name ?module
WHERE {
?student b:studies ?module .
?student foaf:name ?name .
}
ORDER BY ?name

SELECT ?name ?age
WHERE {
?student b:age ?age .
?student foaf:name ?name .
}
ORDER BY DESC (?age) ASC (?name)
```

Limiting the number of results

```
SELECT ?name ?module
WHERE {
?student b:studies ?module .
?student foaf:name ?name .
}
LIMIT 20
```

Extracting subsets of the results

```
SELECT ?name ?module
WHERE {
?student b:studies ?module .
?student foaf:name ?name .
}
ORDER BY ?name
OFFSET 20
LIMIT 20
```

- Note that if no ORDER BY is specified, the order of results is random, and may vary through multiple executions of the same query. Thus, extracting a subset with OFFSET and LIMIT is only useful if an ORDER BY is also used.

Obtaining a Boolean result

- *Is any student studying any module?*
  ASK { ?student b:studies ?module }
- *Is any student studying CS909?*
  ASK { ?student b:studies bmod: CS909 }
- *Is student 029389 studying CS909?*
  ASK { bstu:029389 b:studies bmod: CS909 }
- *Is anyone whom 029389 knows, studying CS909?*
  ASK {bstu:029389 foaf:knows ?x . ?x b:studies bmod: CS909 }
- *Is any student aged over 30 studying CS909?*

Obtaining unique results

```
SELECT ?student
WHERE { ?student b:studies ?module }
```

The above query would return each student several times, because the pattern above will match once for each module a student is taking. To avoid this:

```
SELECT DISTINCT ?student
WHERE { ?student b:studies ?module }
```

Constructing an RDF result

```
CONSTRUCT
{ ?student b:studyFriend ?friend }
WHERE {
?student b:studies ?module .
?student foaf:knows ?friend .
?friend b:studies ?module }
}
```

- The section after the CONSTRUCT keyword is a specification, in triples, of an RDF graph that is constructed to hold the search result. If there is more than one search result, the triples from each result are combined.