University of Dublin
Trinity College

Information Graphs

RDF, Linked Data, SPARQL

Representing Information: Data and Relationships

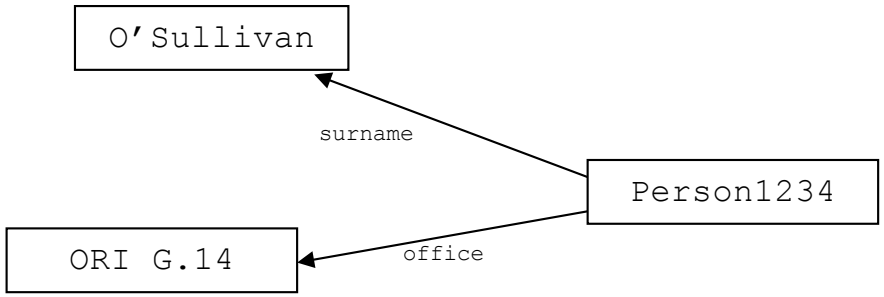
In your modules to date, for representation of data and their relationships, you have already covered :

- As *objects*, using the well-accepted techniques of object-oriented analysis and design to capture a model
- As *a tree-like* representation (e.g. Binary, AVL, XML etc.)
- As *clauses* of facts, with AI languages like Lisp or rule languages
- As *UML*, but not easily machine processable

This week we are going to investigate representing data and their relationships as graphs

- Assert advantages: infinitely extensible, mergeable and scalable

Graphs

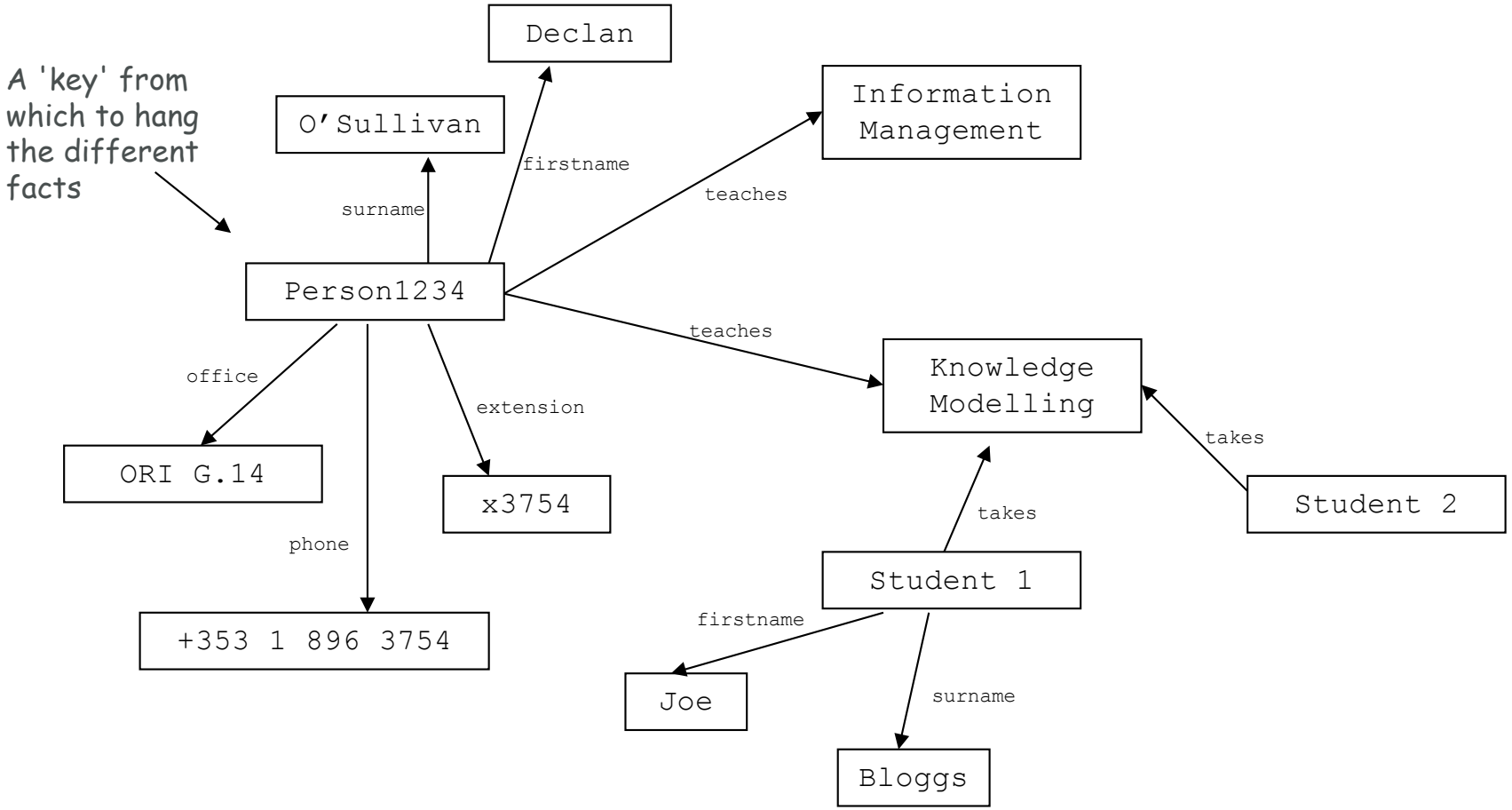


We can use the nodes of a graph for facts and the arcs as (binary) directed relationships between them

- Arcs are typically called predicates or relationships in this view
- The set of arcs intersecting a node tells us the information we know about that fact or entity

Graphs of information– 1

How do we use graphs to represent information?



Graphs of information– 2

Things to note

- Scaling – the same graph can represent a load of different information simultaneously
- Agreement – need to know what the various predicates 'mean'
- Structure – you need to know what nodes are related by a predicate
- Plurality – the same relationship may appear several times
- Symmetry – the same predicates can be used for common information, despite minor changes
- Asymmetry – relationships are inherently directed, which sometimes makes things awkward

…and this can get very tricky

…and this can be difficult to keep straight

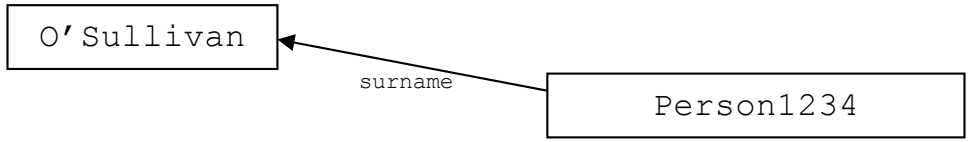
For example both lecturers and students have names

So an information graph is inherently directed

Two ways to view a graph

As nodes and arcs

- Nodes store facts, edges store relationships between them



As triples

- A three-place relationship of 'subject, predicate, object'
- The node and edge structure is induced by the triples – **each triple defines an edge**, the different subjects/objects are the population of nodes, one node per individual string

```
Person1234 surname O'Sullivan
```

Resource Definition Framework (RDF)

RDF is an XML schema that defines how to describe **information** graphs in XML in a standard way

- Standardised by the World Wide Web Consortium (W3C)
- A way of defining information graphs
- No standard predicates
- Tool support – editors, parsers, displays, …

RDF is **not** a knowledge standard *but as we will see later* it is a way of *defining* knowledge standards in a way that maximises the potential for re-use across the web

- Standard predicates for representing knowledge (e.g. OWL)
- Reasoning Tools (e.g. Protégé)

Lassila and Swick, Resource Definition Framework model and specification. W3C report, 1999

Basic structure – triples

RDF represent information using a triple structure

- Subject
- Predicate
- Object

Remember, a triple structure is one way of viewing a graph, so RDF essentially defines an information graph

Information is built up as a collection of these triples, contained within an XML file

Namespace

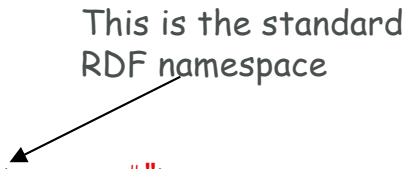
RDF uses XML namespaces to identify predicates

- Define a namespace representing a family of predicates we want to use, for example a model of location
- Define a set of elements and attributes within the namespace to model the phenomenon we're interested in
- Use the fully-qualified name of the elements to specify the predicate we want uniquely

Basic RDF structure

RDF data can live inside other XML documents

Identified by a top-level element from the well-known RDF namespace




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

Within the `RDF` element we can define triples

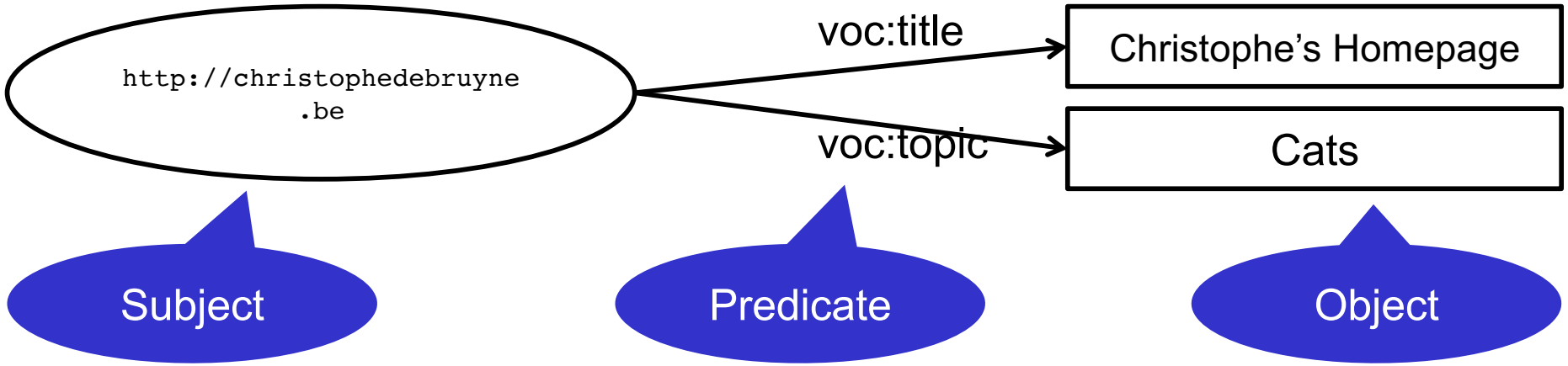
- Co-exists well with the rest of XML – if you don't understand RDF you can elide everything within this element
- Readable enough for people to build by hand

In addition to what follows you may want to check out Manola and Miller, RDF Primer. W3C draft technical note, 2002.

Simple Example and visualisation

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:voc="http://example.org/voc#">

  <rdf:Description rdf:about="http://christophedebruyne.be">
    <voc:title>Christophe's Page</voc:title>
    <voc:topic>Cats</voc:topic>
  </rdf:Description>

</rdf:RDF>
```

RDF Namespace



Another example

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:s="http://www.kdeg.cs.tcd.ie/xml/demo.html#">
<rdf:Description rdf:about="http://www.kdeg.cs.tcd.ie/">
<s:describes>Knowledge and Data Engineering Group</s:describes>
</rdf:Description>
</rdf:RDF>
```

The RDF namespace

The namespace for the predicates we want to use

A standard RDF element meaning 'the subject of the triples contained in this element'

A triple whose predicate is taken from the `s` namespace and whose object is the given text

See how the predicate gets created from the namespace?

Subject: http://www.kdeg.cs.tcd.ie

Predicate: http://www.kdeg.cs.tcd.ie/xml/demo/demo.html#describes

Object: Knowledge and Data Engineering Group

A cluster of facts

Given a common subject we can build a cluster of facts using nested predicate elements

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://www.kdeg.cs.tcd.ie/xml/demo.html#">
  <rdf:Description rdf:about="http://www.kdeg.cs.tcd.ie/">
    <s:describes>Knowledge and Data Engineering Group</s:describes>
    <s:author>S. Punter</s:author>
    <s:phone>+353 1 123 4567</s:phone>
  </rdf:Description>
</rdf:RDF>
```

Each of these gives rise to a triple with the same subject (inherited from the containing `Description` element)

As long as we agree what the predicates mean, we can use whichever we want

An alternative syntax

There's an alternative syntax that compresses all this information into a single element

- Good, because a browser will typically ignore the elements it doesn't understand, and this way there's no spurious element content to mess up the display
- Bad, because it's a bit *too* compact and less flexible than the larger form (see later)

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://www.kdeg.cs.tcd.ie/xml/demo.html#">
  <rdf:Description rdf:about="http://www.kdeg.cs.tcd.ie/"
                   s:describes="Knowledge and Data Engineering Group"
                   s:author="S. Punter"
                   s:phone="+353 1 123 4567"/>
</rdf:RDF>
```

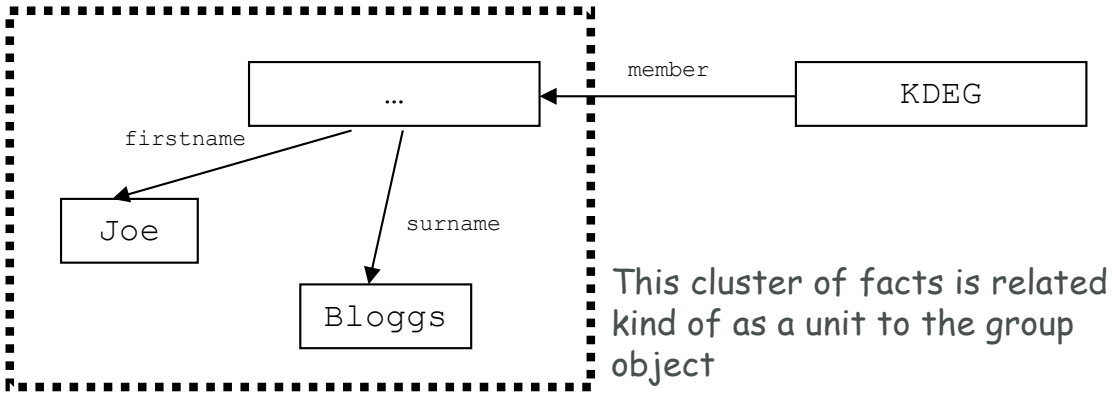
Same predicates, but as attributes rather than an nested elements

Structured values – 1

Suppose we want to talk about a member of KDEG and capture their information too

- Not information about KDEG *per se*, so shouldn't be held as child elements of the `Description` element
- Plus it'd get very confusing when there were several people…

The solution is to define another cluster of facts and relate this *cluster* to KDEG node



Structured values – 2

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://www.kdeg.cs.tcd.ie/xml/demo.html#">
  <rdf:Description rdf:about="http://www.kdeg.cs.tcd.ie/">
    <s:describes>Knowledge and Data Engineering Group</s:describes>
    <s:author>S. Punter</s:author>
    <s:phone>+353 1 123 4567</s:phone>


    <s:member>
      <rdf:Description rdf:about="http://www.cs.tcd.ie/Declan.O"Sullivan'>
        <s:surname>O'Sullivan</s:surname>
        <s:firstname>Declan</s:firstname>
        …
      </rdf:RDF>
    </s:member>
    …


  </rdf:Description>
</rdf:RDF>
```

Rather than having an `about` attribute, the object of this triple is assumed to be the subject of the nested `Description` element

These predicates are now 'about' the subject that immediately contains them

Alternative Representation

The about attribute is used for subjects -- the left-hand side of a triple and the resource attribute is used for objects (targets) -- the right-hand side of the triple.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://www.kdeg.cs.tcd.ie/xml/demo.html#">
  <rdf:Description rdf:about="http://www.kdeg.cs.tcd.ie/">
    <s:describes>Knowledge and Data Engineering Group</s:describes>
    <s:author>S. Punter</s:author>
    <s:phone>+353 1 123 4567</s:phone>
    <s:member rdf:resource="http: //people/DeclanOSullivan'/>
…
  </rdf:Description>
</rdf:RDF>
```

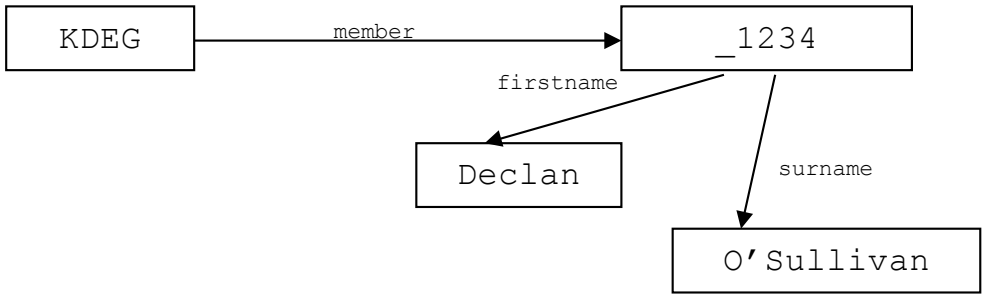
Synthetic structured values

Sometimes the structured value doesn't have an obvious identity to put in an `about` attribute

If this is the case, RDF will synthesise one

```
<s:member>
  <rdf:Description>
    <s:surname>O'Sullivan</s:surname>
    <s:firstname>Declan</s:firstname>
    …
  </rdf:RDF>
</s:person>
```

Description doesn't have an about attribute, so RDF synthesises a unique one for use in the graph



Terse RDF Triple Language – TURTLE

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:voc="http://example.org/voc#">
  <rdf:Description rdf:about="http://christophedebruyne.be">
    <voc:title>Christophe's Page</voc:title>
    <voc:topic>Cats</voc:topic>
  </rdf:Description>
</rdf:RDF>
```

RDF/XML

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix voc: <http://example.org/example#> .
<http://christophedebruyne.be> voc:title "Christophe's Page" .
<http://christophedebruyne.be> voc:topic "Cats" .
```

TURTLE

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix voc: <http://example.org/example#> .
<http://christophedebruyne.be> voc:title "Christophe's Page" ;
                               voc:topic "Cats" .
```

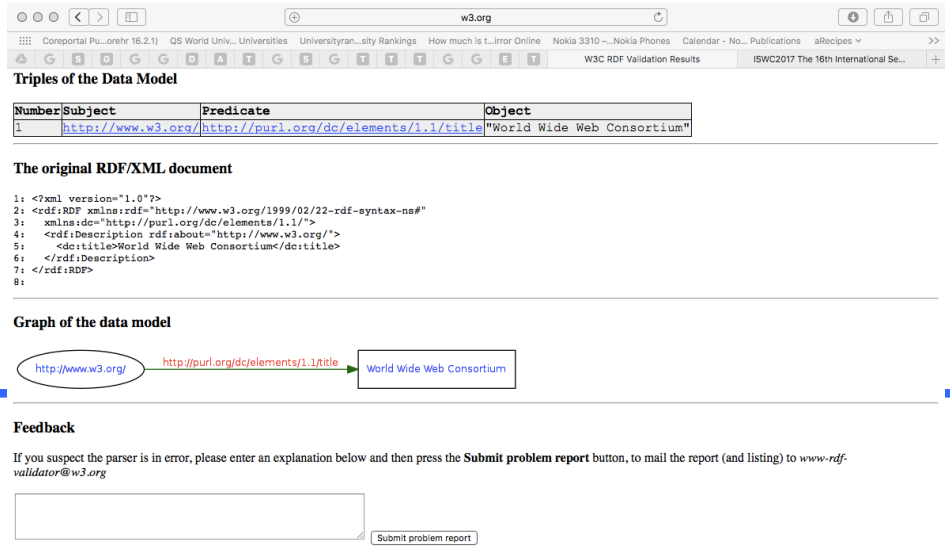
Summary – basic RDF

RDF is a web standard that lets us build information graphs using XML files

- Use URLs for predicates, with namespaces to ensure uniqueness
- Graph built from triples, but using a nested notation
- Multiple facts can be specified with minimal repetition
- Fairly straightforward for humans to write by hand

Validating and Graphing

- http://www.w3.org/RDF/Validator/



Task

1. **Physically Draw an Information Graph describing yourself**
   - **E.g. hobbies, artists_you_like, etc.**
2. **Create the corresponding RDF**
3. Use the RDF validator to validate and visualise the graph
   - http://www.w3.org/RDF/Validator/
4. Email me screenshot of visualised graph by start of next semester
   - *Steps 3 and 4 Not mandatory but remember that tutorials also help prepare for potential exam questions*

Example Syntax

```
<?xml version="1.0"?><rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-
rdf-syntax-ns#"
xmlns:s="http://www.kdeg.cs.tcd.ie/xml/demo
.html#">

<rdf:Description
rdf:about="http://www.kdeg.cs.tcd.ie/">

<s:describes>Knowledge and Data Engineering
Group</s:describes>

<s:author>S. Punter</s:author>

<s:phone>+353 1 123 4567</s:phone>
</rdf:Description>

</rdf:RDF>
```

SPARQL

SPARQL = SPARQL Protocol And RDF Query Language

Background – SPARQL

Stands for *SPARQL Protocol and RDF Query Language*

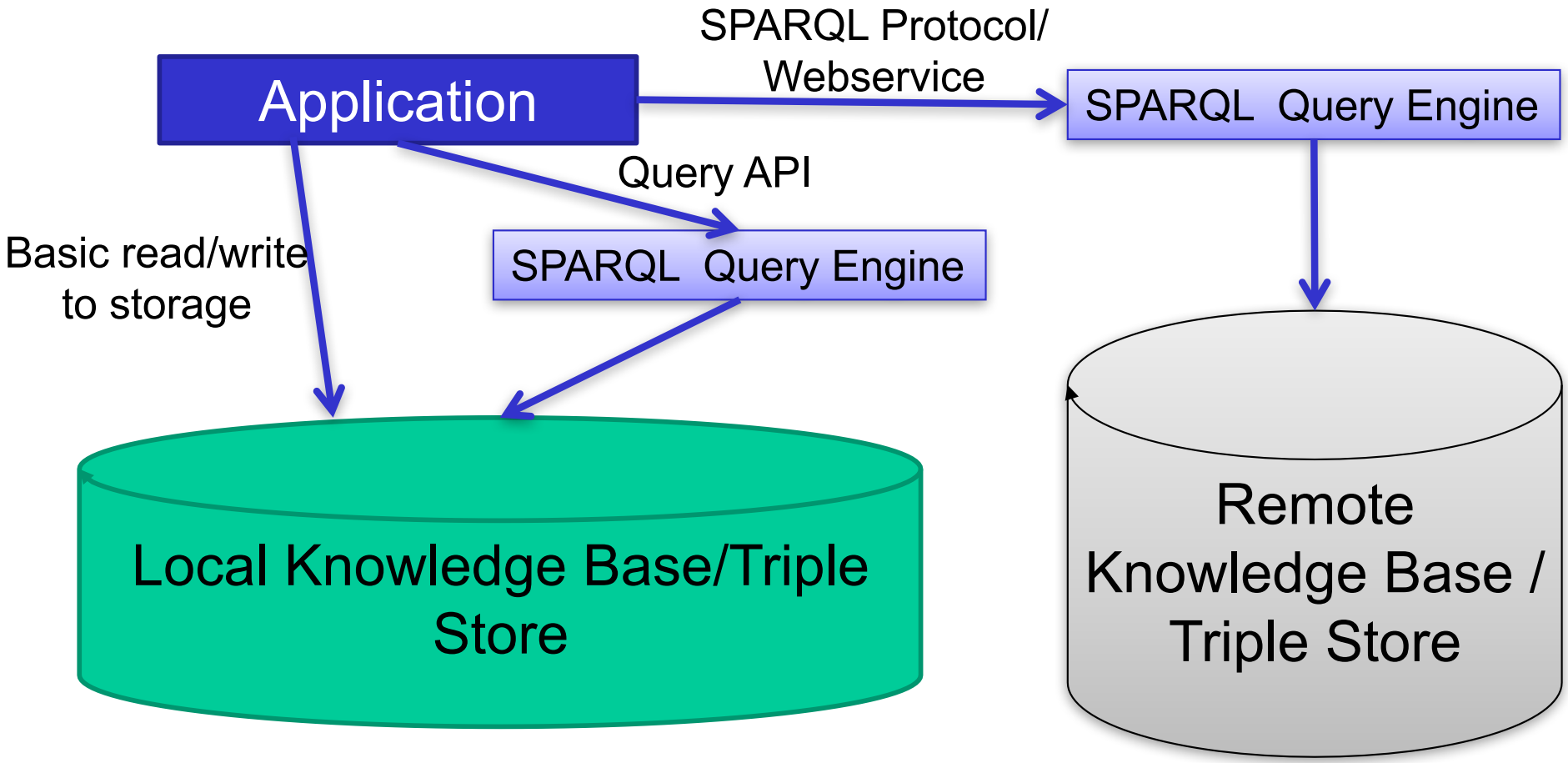
- Recursive acronym

SPARQL 1.1 is a W3C Recommendation since March 2013 (http://www.w3.org/TR/sparql11-overview/)

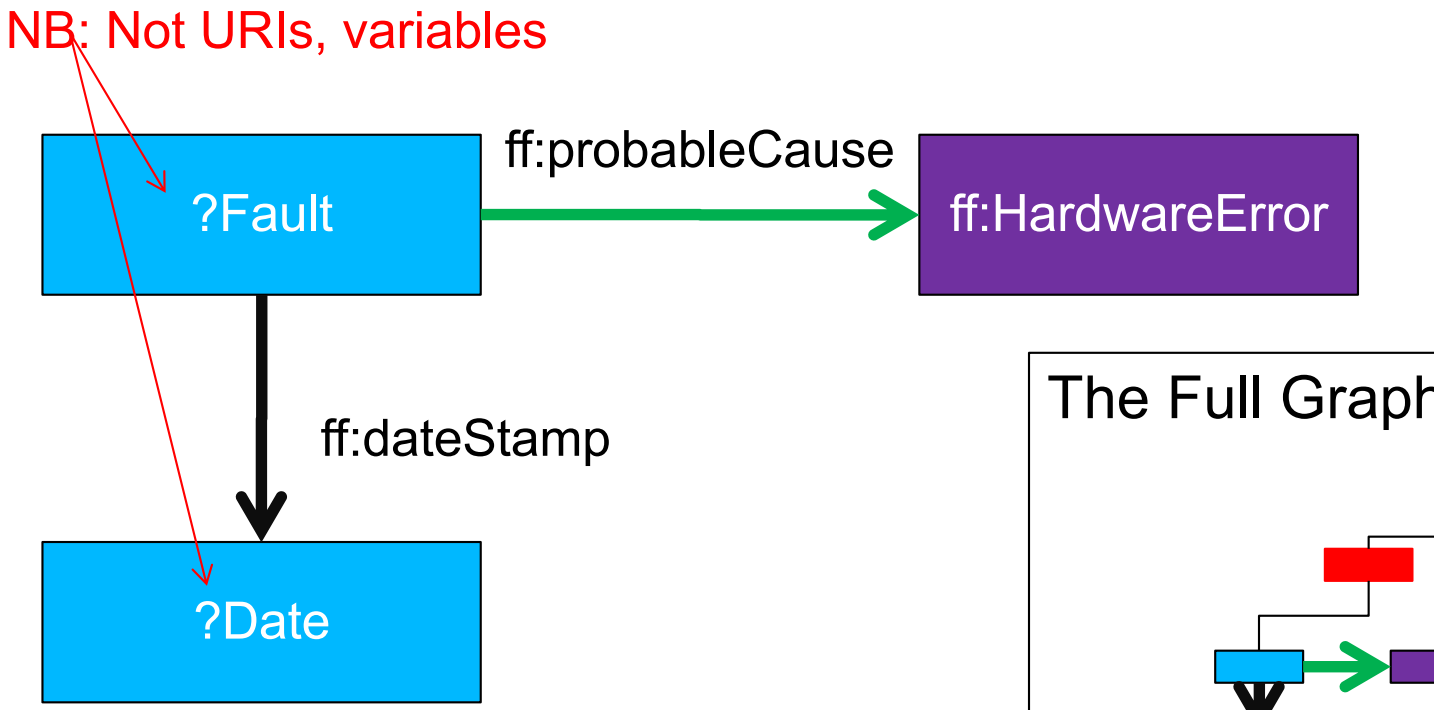
SPARQL allows us to:

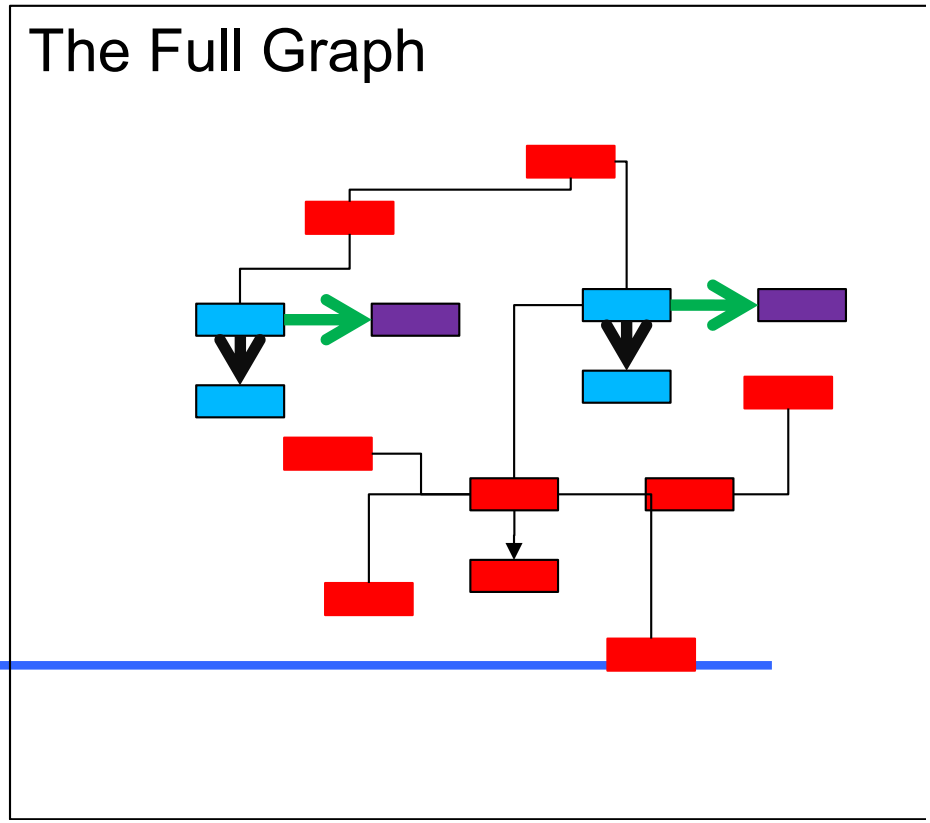
- Pull values from structured and semi-structured data
- Explore data by querying unknown relationships
- Perform complex joins of disparate databases in a single, simple query
- Transform RDF data from one vocabulary to another
- …

Where it fits



SPARQL based on Graph Matching





Background – SPARQL

A SPARQL query comprises, in order:

Prefix declarations for URI shorthands

A **result clause**, identifying what information to return from the query

Dataset definition, stating what RDF graph(s) are being queried

The **query pattern**, specifying what to query for in the underlying dataset

Query modifiers, slicing, ordering, and otherwise rearranging query results

Namespace shortened for brevity

```
# prefix declarations
PREFIX foaf: <http://.../0.1/>
...
# result clause
SELECT ...
# dataset definition
FROM ...
# query pattern
WHERE {
    ...
}
# query modifiers
ORDER BY ...
```

Adapted from “SPARQL by Example” by Feigenbaum and Prud’hommeaux

SPARQL Syntax Overview

SPARQL has 4 result forms:

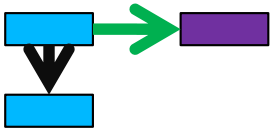
- SELECT – Return a table of results.
- CONSTRUCT – Return an RDF graph, based on a template in the query.
- DESCRIBE – Return an RDF graph, based on what the query processor is configured to return.
- ASK – Ask a boolean query.

The SELECT form directly returns a table

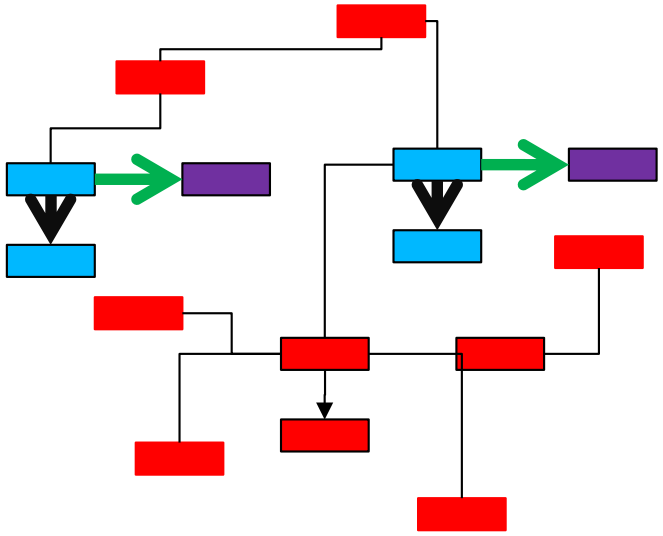
DESCRIBE and CONSTRUCT use the outcome of matching to build RDF graphs

Query Results

1. Triple Pattern



2. Graph Matching

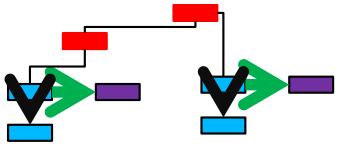


3. Query Results

A. Table

Fault	Date
1001	29 May
1002	30 May

B. New Graph



C. XML

< < ../> .../>

Background – Simple SPARQL Queries

```
PREFIX ex: <http://foo.bar/#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?cat ?name
WHERE {       ?cat a ex:Cat.
                    ?cat foaf:name ?name.      }
```

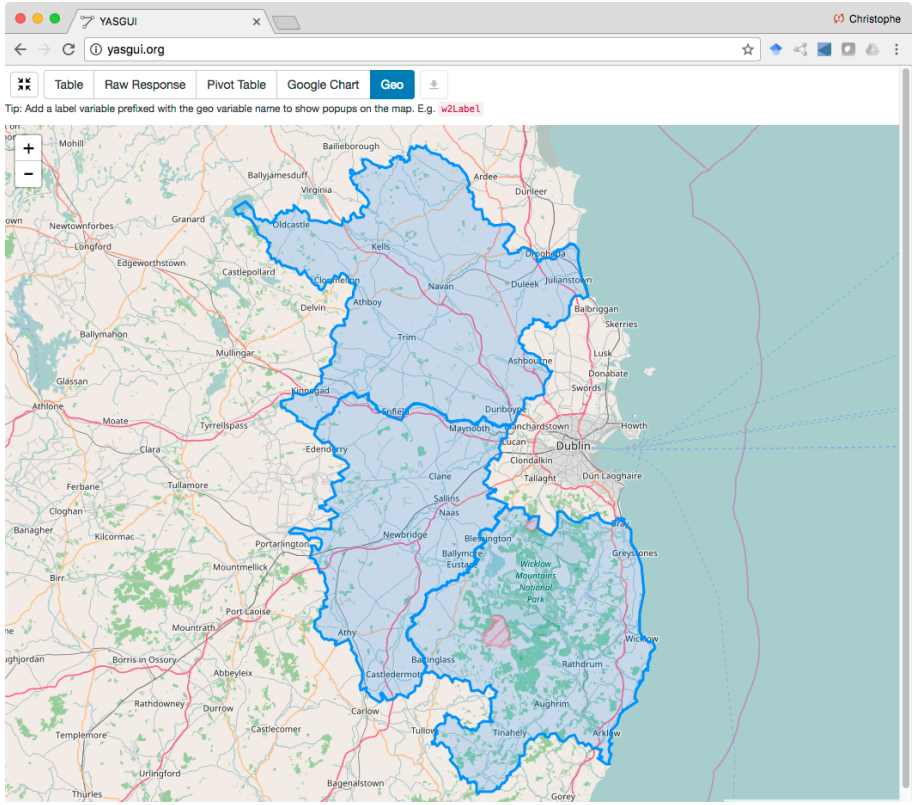
```
-----------------------------------------
| cat                           | name      |
=========================================
| <http://foo.bar/#bettina> | "Bettina" |
| <http://foo.bar/#louis>   | "Louis"   |
| <http://foo.bar/#victor>  | "Victor"  |
-----------------------------------------
```

Background – GeoSPARQL

Give a me a list of counties whose borders touch County Dublin

```
SELECT ?county ?w2 WHERE {
  ?dublin a osi:County .
  ?dublin rdfs:label "DUBLIN" .
  ?county a osi:County .
  ?dublin geo:hasGeometry ?g1 .
  ?g1 geo:asWKT ?w1 .
  ?county geo:hasGeometry ?g2 .
  ?g2 geo:asWKT ?w2
  FILTER (geof:sfTouches(?w1, ?w2)) .
}
```

RDF allows one to engage with data in a declarative manner, not bound to particular structure.



University of Dublin
Trinity College

Linked Data

What is Linked Data?

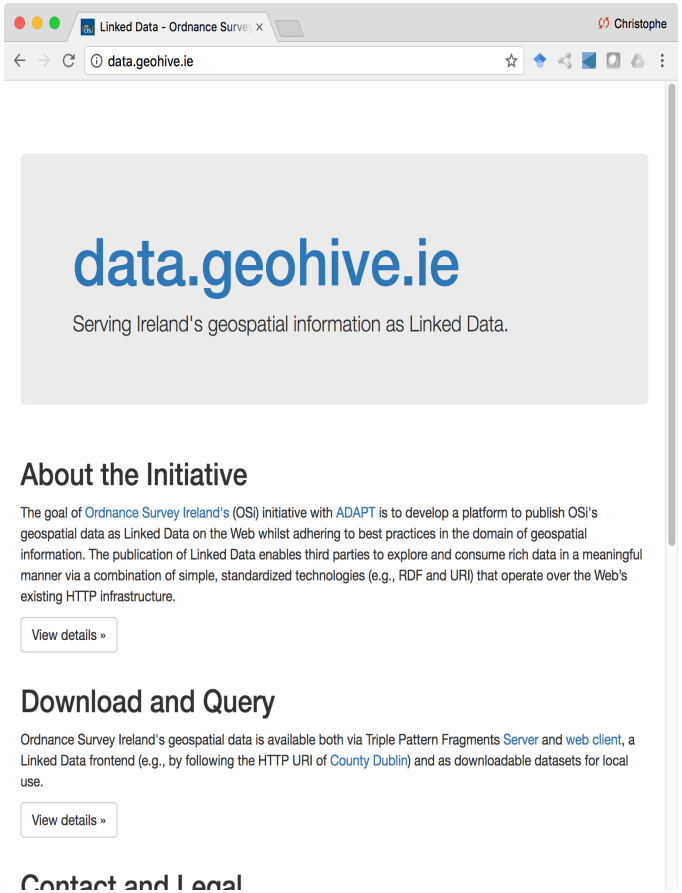
Linked Data started off as a initiative called the Linking Open Data (LOD) project.

Linked Data is a global initiative to publish and interlink structured data on the Web using a clever combination of simple, standardized technologies.

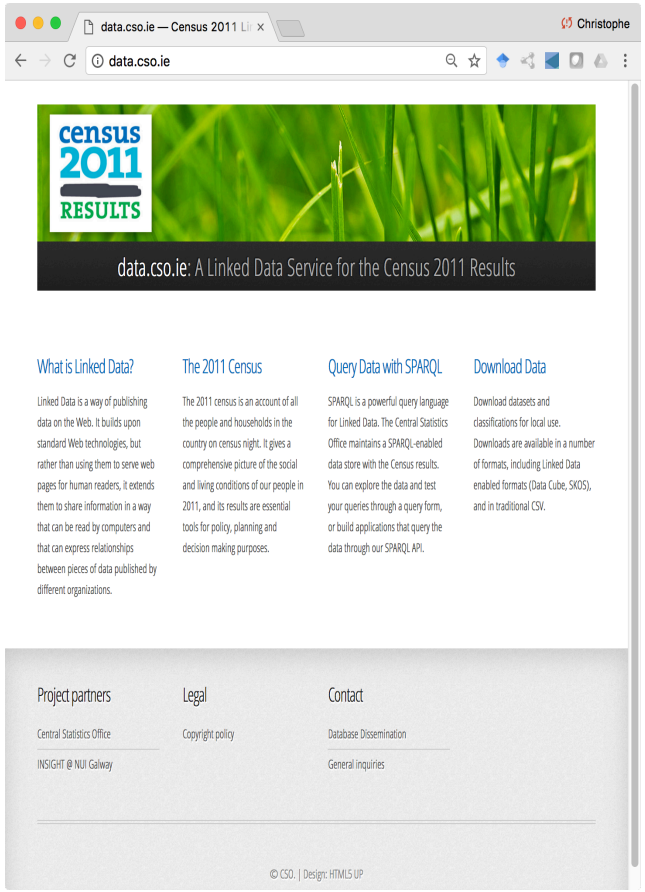
- Uniform Resource Identifiers – to name things;
- Resource Description Framework – to represent things;
- HTTP infrastructure – to obtain those representations.

Examples

http://data.geohive.ie/

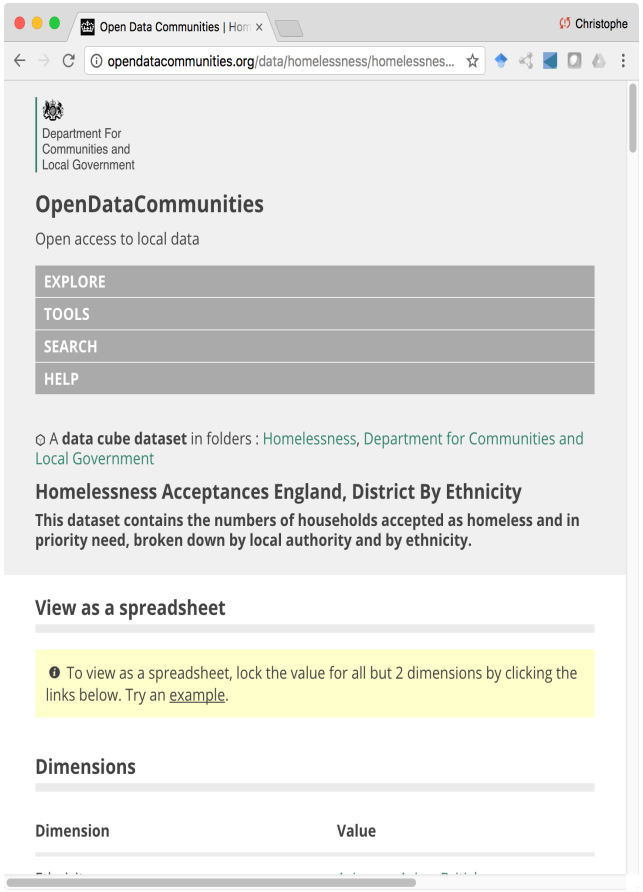


http://data.cso.ie/

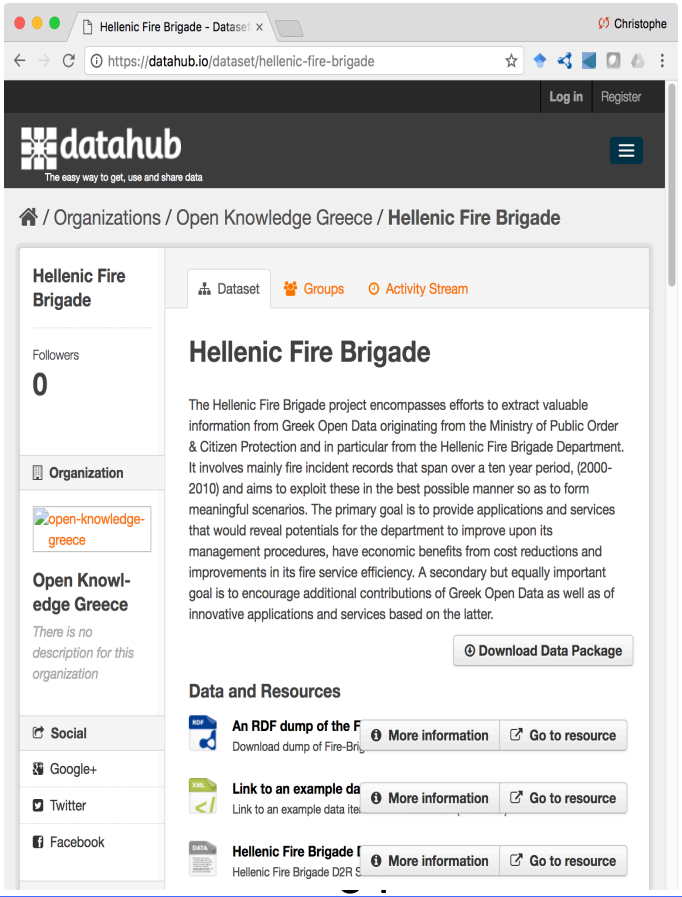


Examples

http://opendatacommunities.org/



https://datahub.io/



Web of Documents vs. Web of Data

The Web of Documents were created by humans for humans; the links between documents bore little meaning for machines and documents provided little structured information.

Structured information can be found on the Web such as XML, CSV, etc. – but, …

How do we link data rather than documents, and create a global “database” of information?

Web of Documents vs. Web of Data

Semantic Web is not only about data, but about making links between data instead of links between documents.

Enabling persons or machines to explore the Web of Data.

Links between arbitrary “things” in data are described in RDF.

Connect distributed data across the Web → http://linkeddata.org/

Web of Documents vs. Web of Data

	Web of Documents	Web of Data
Analogy	Global file system	Global database
Primary objects	Documents	(Descriptions of) Things
Links between	(Parts of) Documents	Things
Degree of structure	Low	High
Semantics between links and content	Implicit	Explicit
Designed for	Human consumption	Both human and computer-based agents

Compiled from http://www.w3.org/2008/Talks/WWW2008-W3CTrack-LOD.pdf

Linked Data

Linked Data is also a community effort to publish *(open)* data sets as Linked Data on the Web (to which anyone can refer to)

According to some “protocol” and ...

Interlink these data sets and ...

Develop clients that consume Linked Data from the Web

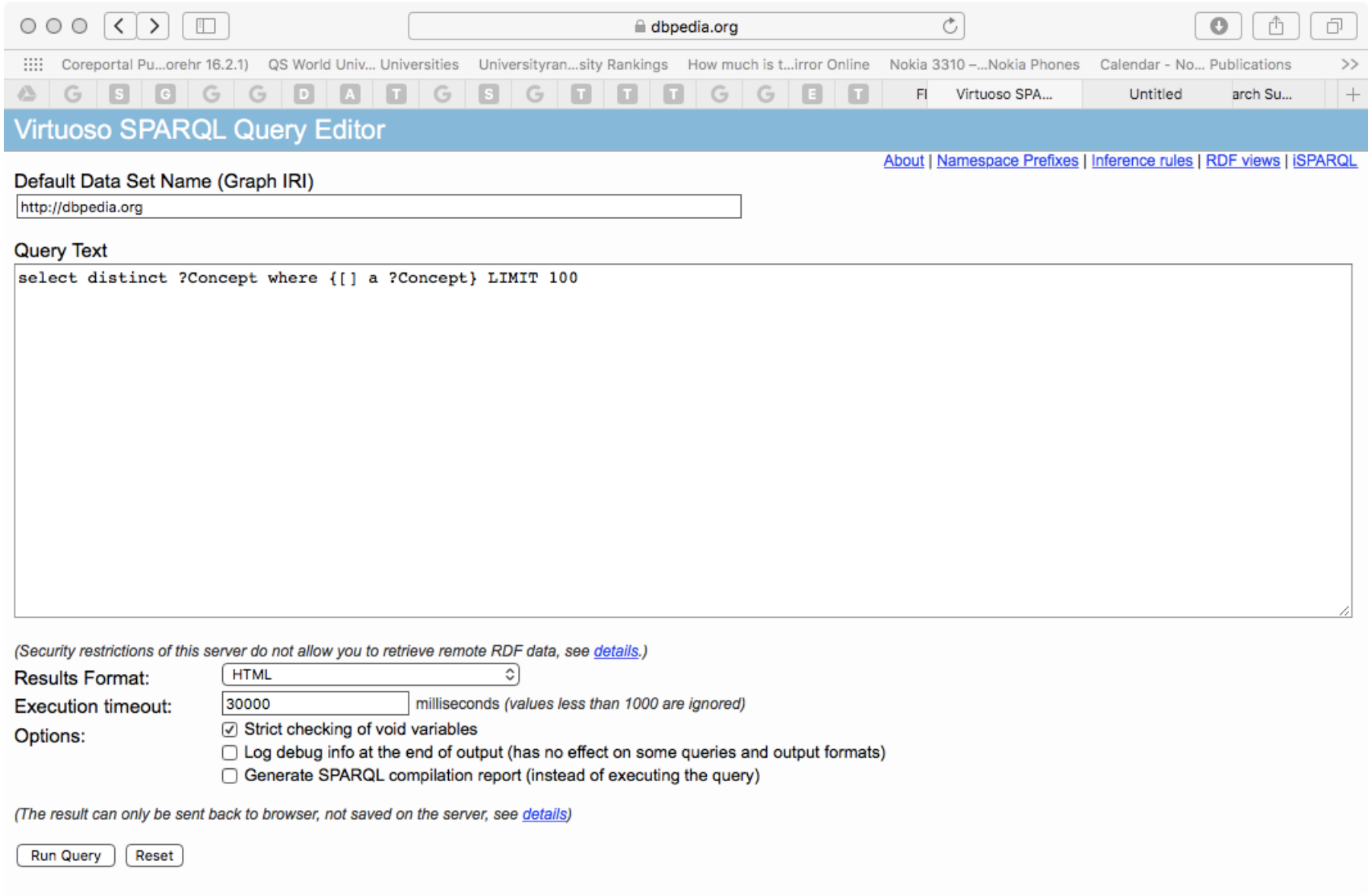
Example Linked Open Data Dataset: DBpedia

DBpedia

Initiative to make available Wikipedia knowledge as
- Linked Open Data

Structure from
- Infoboxes
- HTML (titles)
- Categories
- Links
  - other languages
  - redirects
  - disambiguations, etc

Also a good place to explore using SPARQL
https://dbpedia.org/sparql



Towards a Web of Data

We need appropriate methods (guidelines) and standards.

Tim Berners-Lee formulated four rules for creating and publishing Linked Data on the Web.

Number 1
"Use URIs as names for things."

Number 2
"Use HTTP URIs so that people can look up those names."

Number 3
"When someone looks up a URI, provide useful information using the standards"

Number 4
"Include links to other URIs, so that they can discover more things."

Provide useful Information for URI look-ups

When entities are identified by URIs that use the `http://` scheme, these entities can be looked up simply by dereferencing the URI over the HTTP protocol.

Simple, standardized mechanism for retrieving resources via these URIs.

Provide information suitable for the “consumer”

- Suitable for browsers vs. suitable for machines
- Humans rather see HTML pages, PDFs, pictures,
- Machines want machine-readable formats such as RDF

(Non-)Information Resources

Information resources are documents – referred to by a URI – that describe non-information resources – named with a URI – that represent things such as cars, people, etc.

The NIR http://dbpedia.org/resource/James_Joyce is described by the following IRs:

- The web page http://dbpedia.org/page/James_Joyce
- The RDF doc http://dbpedia.org/data/James_Joyce

Either is returned depending on what you need. How?

Content Negotiation

Resource identifiers:

HTTP URIs not only as a name, but also for a Web look-up.

Non-information resources can have multiple representations: HTML, RDF/XML, ...

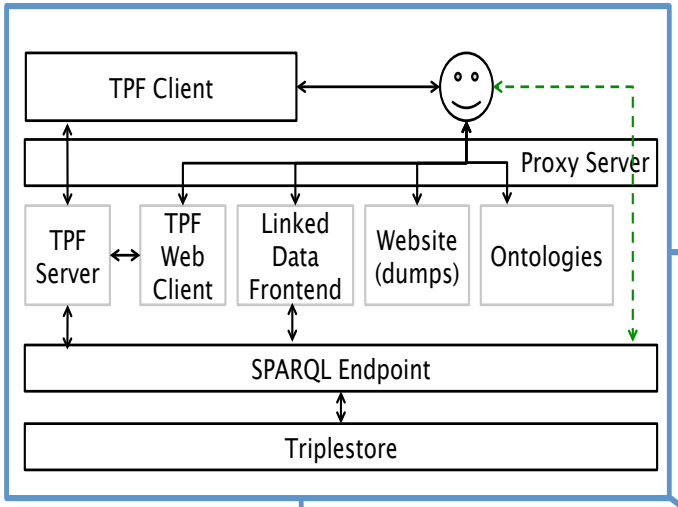
HTTP URI dereferencing:

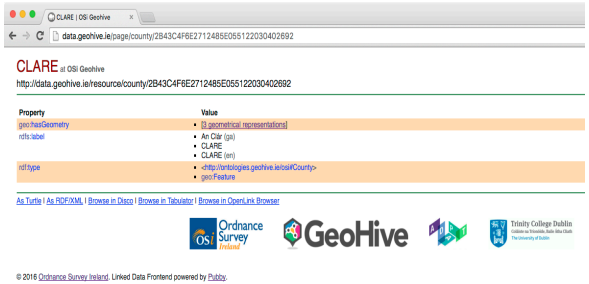
To dereference → “To obtain the address of a data item held in another location from a pointer”

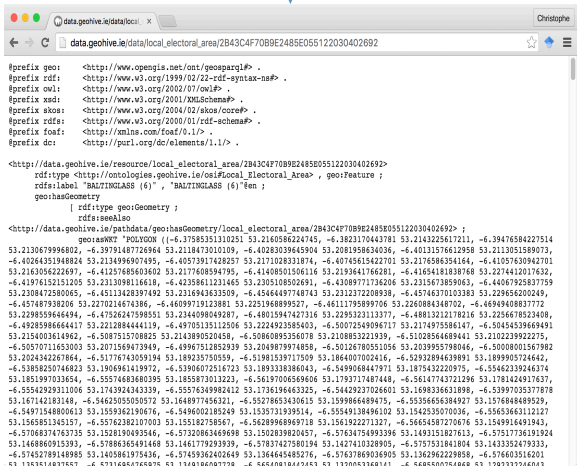
URI pointing to a IR returns the representation.

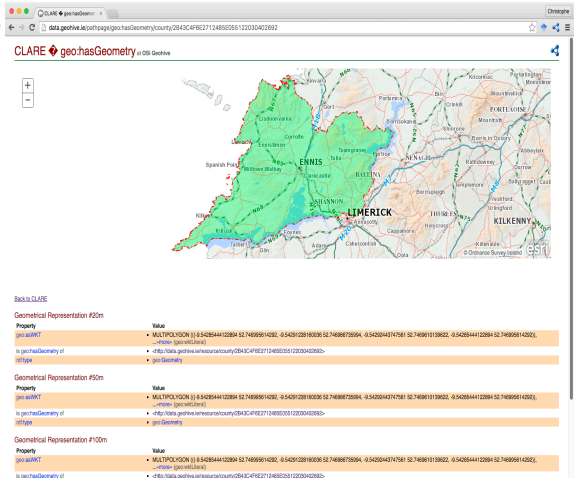
URI pointing to a NIR returns a redirect to an IR describing that NIR.

Content negotiation in data.geohive.ie









Include links to other URIs

Not only within the same dataset

```
<http://dbpedia.org/resource/James_Joyce>
dbpedia-owl:birthPlace
<http://dbpedia.org/resource/Dublin> .
```

But also across datasets

```
<http://dbpedia.org/page/Dublin>
owl:sameAs
<http://sws.geonames.org/7778677/> .
```

Uplifting Tabular data into RDF

R2RML: RDB to RDF Mapping Language

- A W3C Recommendation since fall 2012
- http://www.w3.org/TR/r2rml/

Creating an R2RML file that annotates a relational database with existing vocabularies and/or ontologies (RDFS or OWL).

That R2RML file goes through an *R2RML Mapping Engine* to produce RDF.

R2RML specifies

- An ontology to specify those mappings;
- How those mappings should be interpreted to produce RDF.
- R2RML files are thus stored as RDF.

Example

City	
ID	Name
1	Dublin
2	Ghent

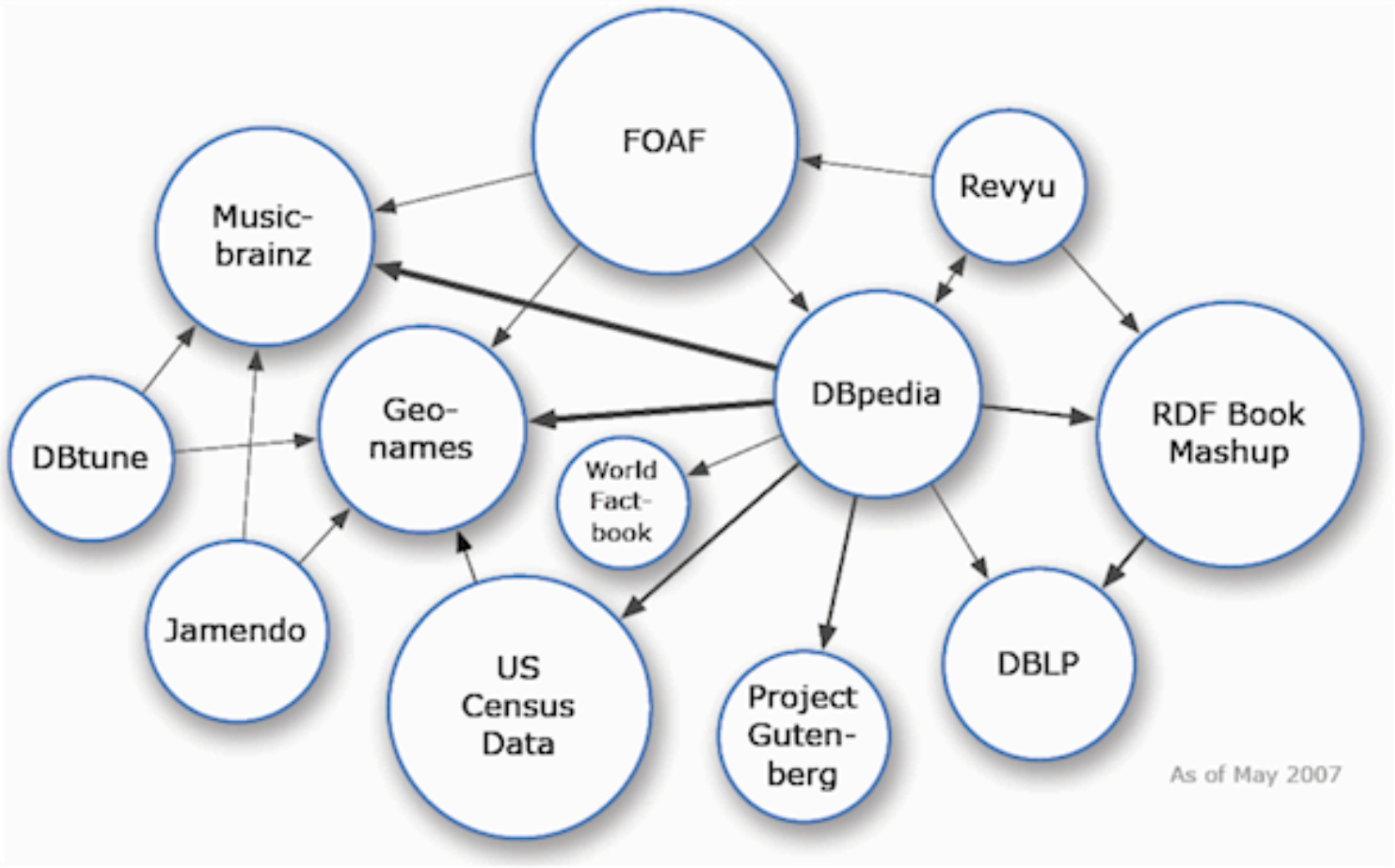
What is being mapped? A logical table/view or an SQL query.

How to generate and state something about the subject of those triples.

How to generate predicates and objects.

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dbpedia: <http://dbpedia.org/ontology/> .

<#CityTriplesMap>
        a rr:TriplesMap ;

        rr:logicalTable [ rr:tableName "City" ] ;

        rr:subjectMap [
                rr:template "http://foo.example/City/{ID}" ;
                rr:class dbpedia:Place ;
        ] ;

        rr:predicateObjectMap [
                rr:predicate foaf:name ;
                rr:objectMap [ rr:column "Name" ] ;
        ] ;
        .
```

Linked Open Data 2007



LOD Cloud 2014 Statistics

By Max Schmachtenberg, Christian Bizer, and Heiko Paulheim in the context of the EU PlanetData project.

Also provides information about vocabularies used as well as popular predicates for interlinking.

No statistics gathered for the 2017 diagram.

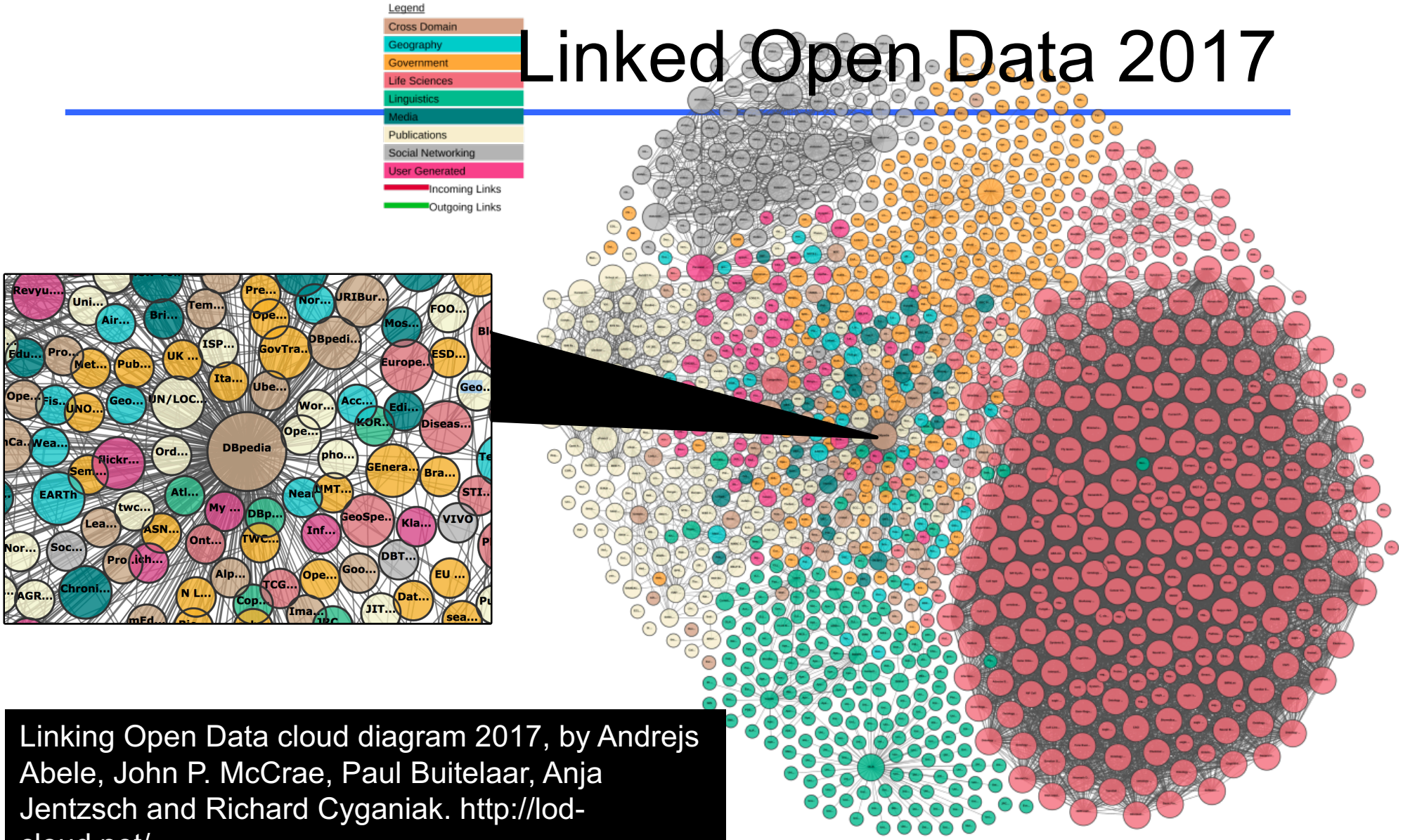
Datasets by topical domain.

Topic	Datasets	%
Government	183	18.05%
Publications	96	9.47%
Life sciences	83	8.19%
User-generated content	48	4.73%
Cross-domain	41	4.04%
Media	22	2.17%
Geographic	21	2.07%
Social web	520	51.28%
Total	**1014**	

Categorization by number of linked datasets

Number of linked datasets	Number of datasets
more than 10	79 (7.79%)
6 to 10	81 (7.99%)
5	31 (3.06%)
4	42 (4.14%)
3	54 (5.33%)
2	106 (10.45%)
1	176 (17.36%)
0	445 (43.89%)

Linked Open Data 2017



Linking Open Data cloud diagram 2017, by Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch and Richard Cyganiak. http://lod-cloud.net/

Reminder Group Project

Part 2: XML Implementation

STEP 1: XML DESIGN

From your group's UML Class diagram, pick **at least** 7 classes and for each create a different XML document (that is they each have a different DTD for each XML document), with the following characteristics for each XML document:

a) **At least 6** different XML elements/tags are used.

b) **At least one third** of the XML elements should have 1 XML attribute

c) There is **interlinks between** some of the documents (reflecting the assocations/relationships between the classes within the UML design), with enough information information to allow for interesting cross document XML Queries to be designed

1.For each DTD

Use comments to clearly state what is the purpose of the document, and comments describing purpose of each element and for each attribute, and why certain cardinality (*,+ etc.) is used.

You should end up **7 XML** documents with **7 commented DTDs**.

Part 2: XML Implementation

STEP 2: XML QUERY DESIGN

Design and Document **at minimum 8** interesting **XQuery** queries that support some of your UML use cases, with the following characteristics:

- At least 3 of the queries should retrieve information from two or more interlinked XML documents, using the WHERE clause
- At least 2 of the queries should use the FOR clause
- At least 1 of the queries should use the LET clause
- At least 2 of the queries should use a Built-in XQuery function
- At least 2 of the queries should use User Defined Functions

In the report, for each query, you need to document:

(a) identification of the UML use case that it supports
(b) description of the purpose of the query
(c) provide example of output that you expect when query is executed.

Part 2: XML Implementation

STEP 3: Submit Report and demonstrate XQueries

1. **ALL** GROUPS Sign in Group Report on **Monday 11th December 2017at 10am**

2. **Demonstrate** your XQueries at allocated lab on either Monday **December 11th** or Thursday **December 14th 2017 (see next slide)**

GROUP REPORT

- Final UML Design report with Ethics Canvas (as per part 1)
- What (if anything) did you need to change in going from UML design to XML implementation?
- List who did what in the group for XML implementation
- Strengths and Weaknesses of the XML design and XQueries design
- Include XML documents and commented XML DTDs (see earlier slides)
- Include the documented XML Queries (see earlier slides)

Demo Schedule

Monday 11th December 10 to 11am::

Groups 1 to 8 inclusive

Monday 11th December 11am to 12 noon::

Groups 9 to 16 inclusive

Thursday 14th December 11am to 12 noon::

Groups 17 to 25 inclusive