Introduction to the Semantic Web

2nd European Semantic Technology Conference
Vienna, Austria

September 24, 2008

Ivan Herman, W3C

Towards a Semantic Web

- The current Web represents information using
  - natural language (English, Hungarian, Chinese,…)
  - graphics, multimedia, page layout
- Humans can process this easily
  - can deduce facts from partial information
  - can create mental associations
  - are used to various sensory information
    - (well, sort of… people with disabilities may have serious problems on the Web with rich media!)

Towards a Semantic Web

- Tasks often require to combine data on the Web:
  - hotel and travel information may come from different sites
  - searches in different digital libraries
  - etc.
- Again, humans combine these information easily
  - even if different terminologies are used!

However…

- However: machines are ignorant!
  - partial information is unusable
  - difficult to make sense from, e.g., an image
  - drawing analogies automatically is difficult
  - difficult to combine information automatically
    - is `<foo:creator>` same as `<bar:author>`?
  - …

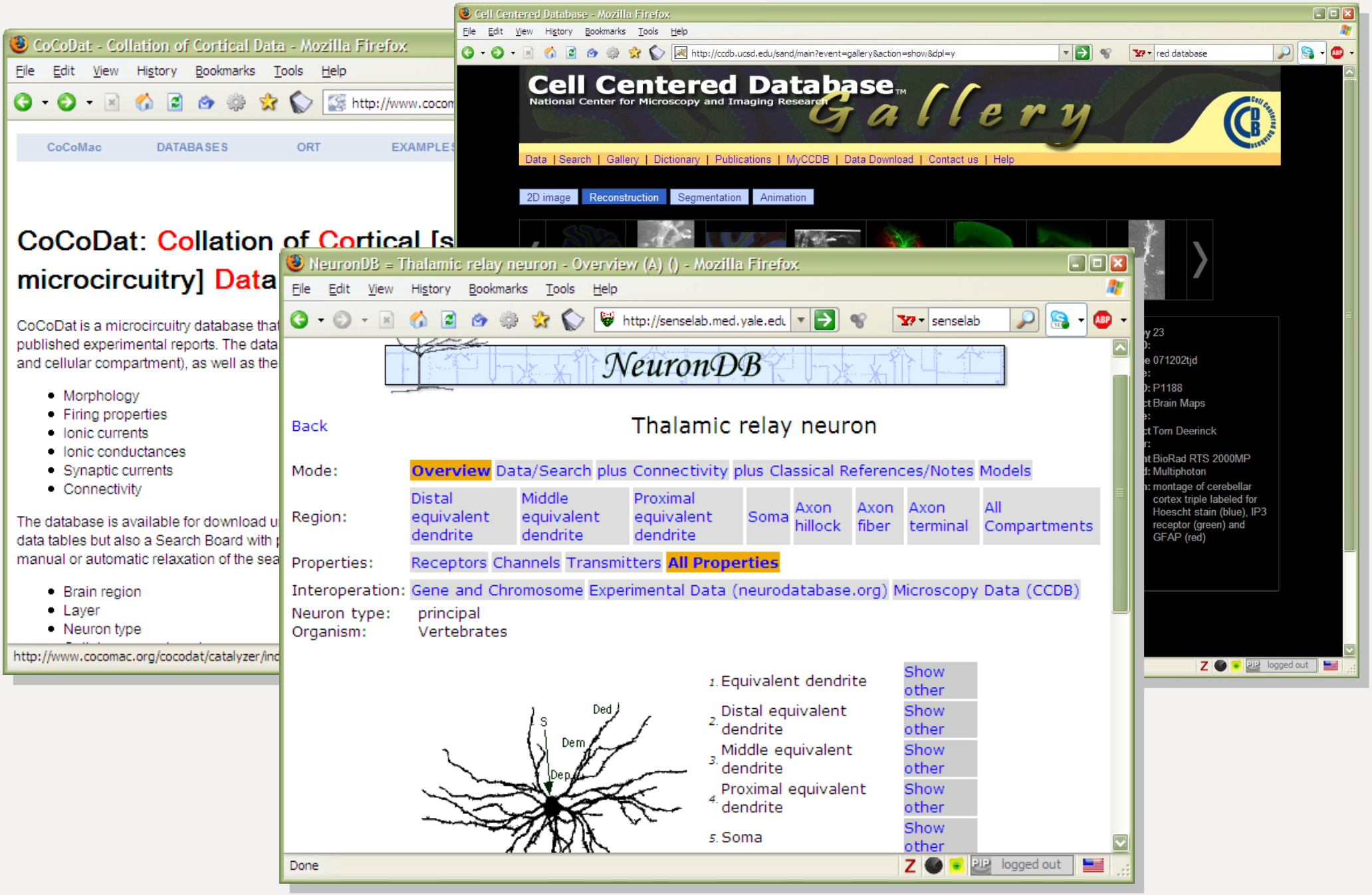
Example: automatic airline reservation

- Your automatic airline reservation
  - knows about your preferences
  - builds up knowledge base using your past
  - can combine the local knowledge with remote services:
    - airline preferences
    - dietary requirements
    - calendaring
    - etc
- It communicates with remote information
  - (M. Dertouzos: The Unfinished Revolution)

Example: data(base) integration

- Databases are very different in structure, in content
- Lots of applications require managing several databases
  - after company mergers
  - combination of administrative data for e-Government
  - biochemical, genetic, pharmaceutical research
  - etc.
- Most of these data are accessible from the Web (though not necessarily public yet)

And the problem *is* real…



Example: social networks

- Social sites are everywhere these days (LinkedIn, Facebook, Dopplr, Digg, Plexo, Zyb, …)
- How many times did you have to add your contacts?
- Applications should be able to get to those data via standard means
  - there are, of course, privacy issues…

What is needed?

- (Some) data should be available for machines for further processing
- Data should be possibly combined, merged on a Web scale
- Sometimes, data may describe other data…
- … but sometimes the data is to be exchanged by itself, like my calendar or my travel preferences
- Machines may also need to *reason* about that data

In what follows…

- We will use a simplistic example to introduce the main Semantic Web concepts
- We take, as an example area, data integration

The rough structure of data integration

1. Map the various data onto an abstract data representation
   - make the data independent of its internal representation…
2. Merge the resulting representations
3. Start making queries on the whole!
   - queries not possible on the individual data sets

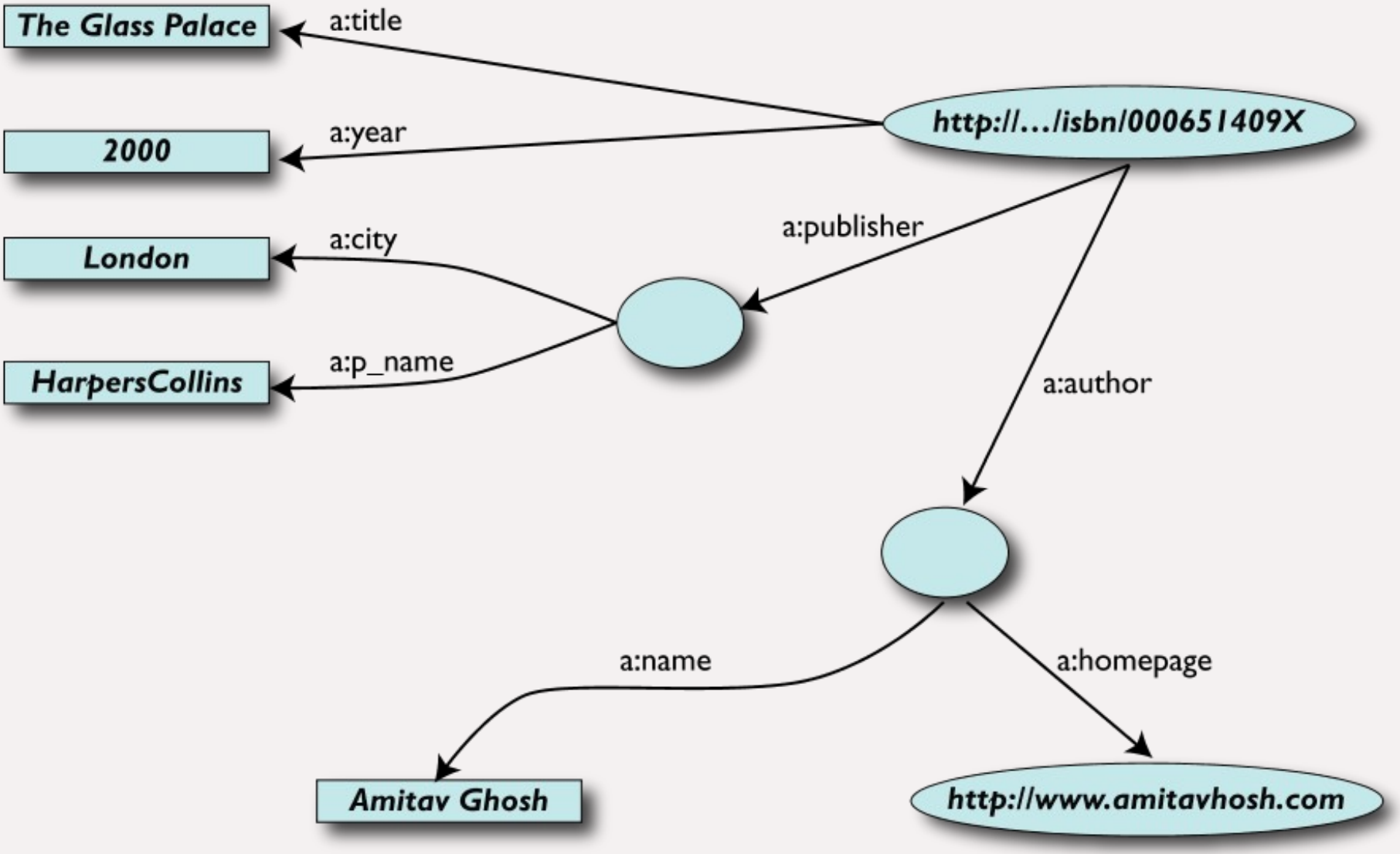
A *simplified* bookstore data (dataset "A")

ID	Author	Title	Publisher	Year
ISBN0-00-651409-X	id_xyz	The Glass Palace	id_qpr	2000

ID	Name	Home Page
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com

ID	Publ. Name	City
id_qpr	Harpers Collins	London

1st: export your data as a set of *relations*



Some notes on the exporting the data

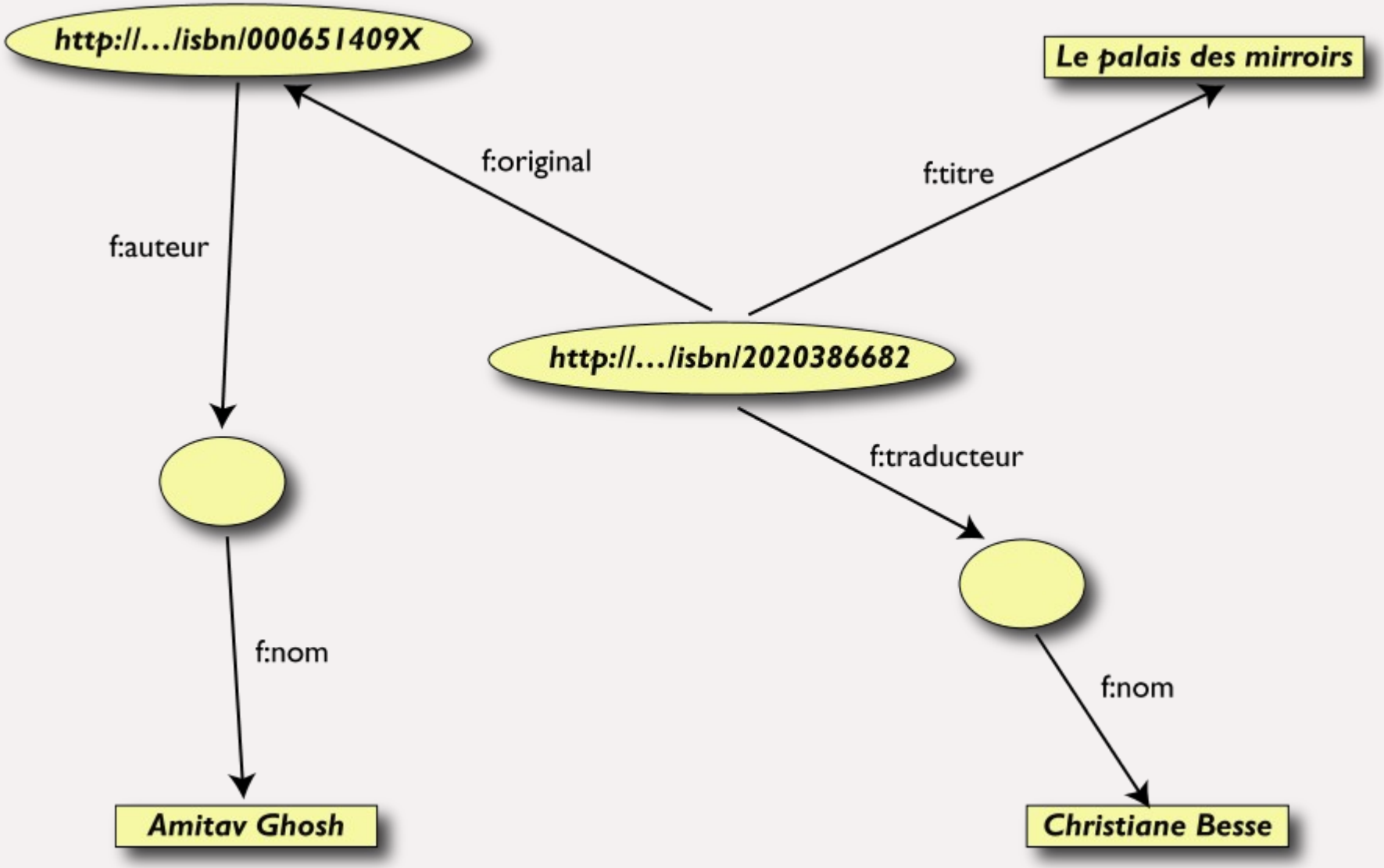
- Data export does *not* necessarily mean physical conversion of the data
  - relations can be generated on-the-fly at query time
    - via SQL “bridges”
    - scraping HTML pages
    - extracting data from Excel sheets
    - etc.
- One can export *part* of the data

Another bookstore data (dataset “F”)

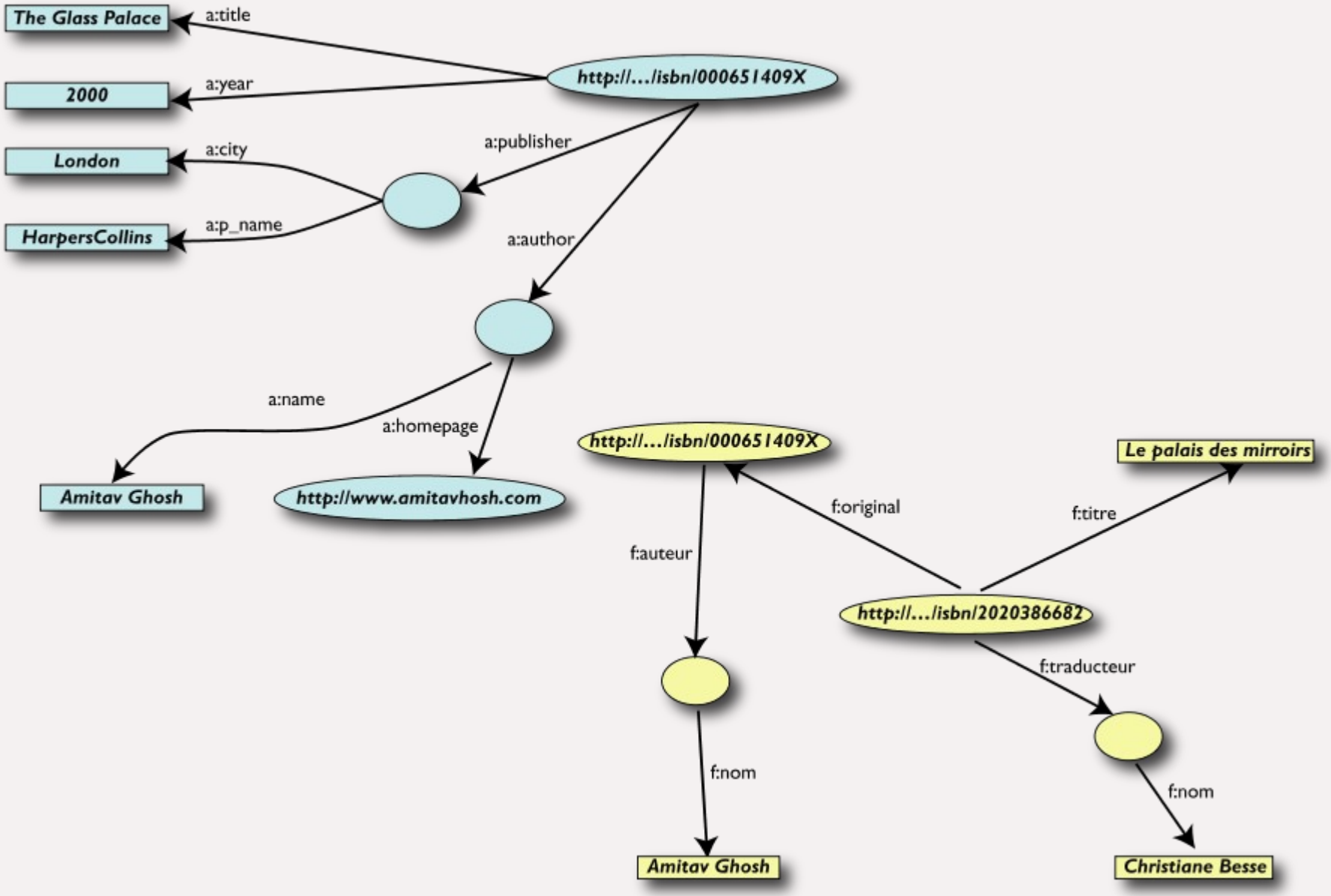
	A	B	C	D	E
1	**ID**	**Titre**	**Auteur**	**Traducteur**	**Original**
2	ISBN0 2020386682	Le Palais des miroirs	A7	A8	ISBN-0-00-651409-X
3					
4					
5					

6	**Nom**
7	Ghosh, Amitav
8	Besse, Christianne

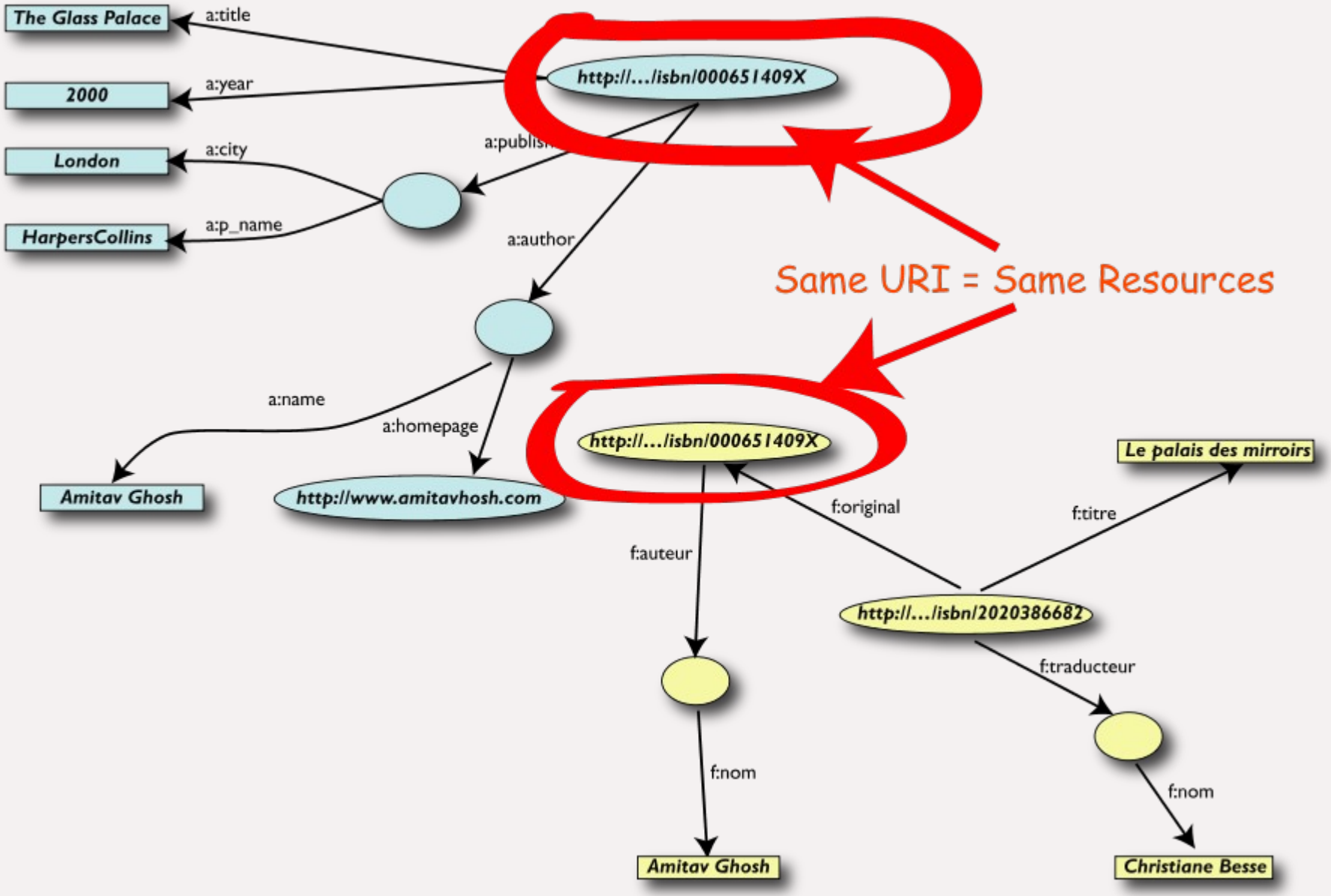
2nd: export your second set of data



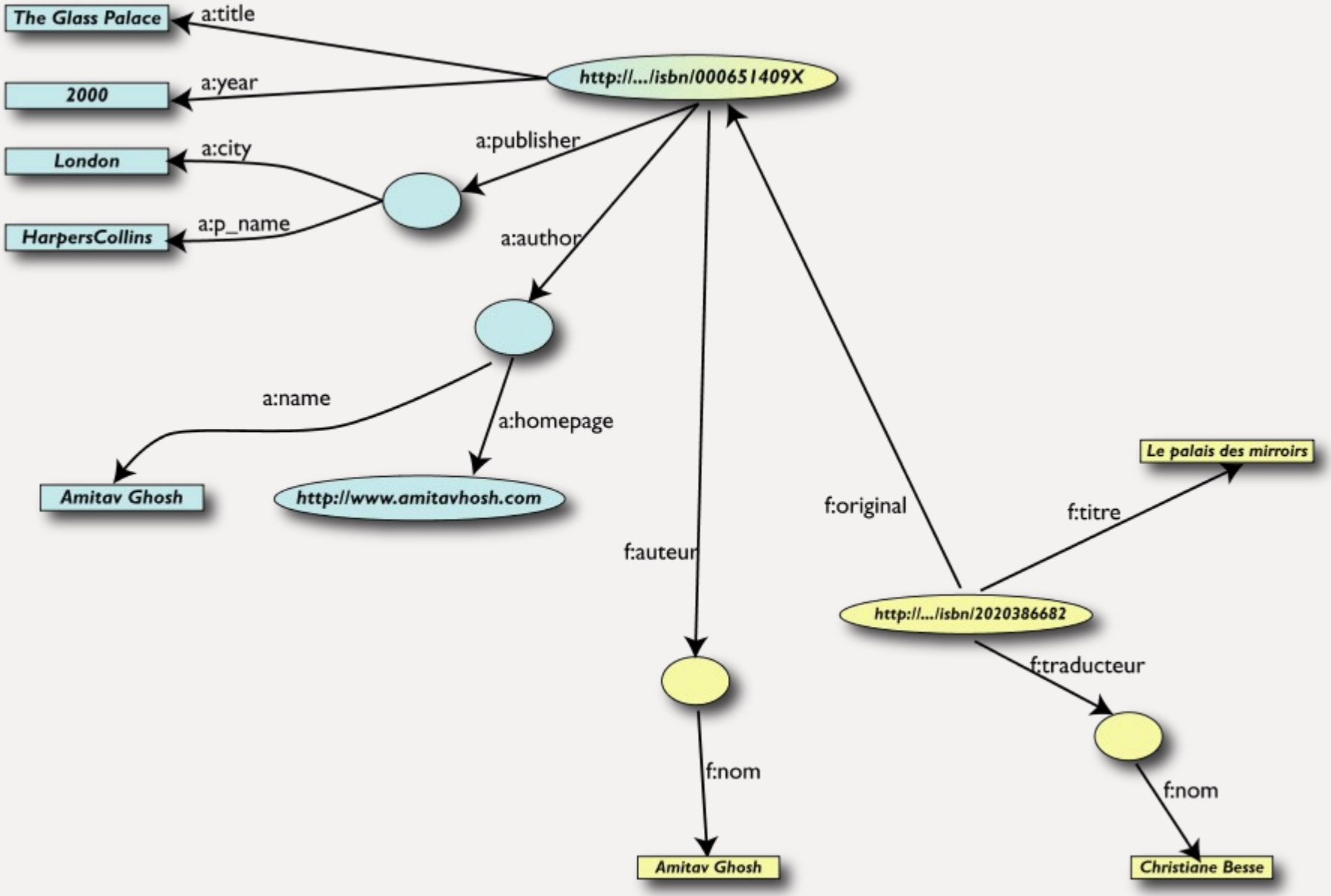
3rd: start merging your data



3rd: start merging your data (cont.)

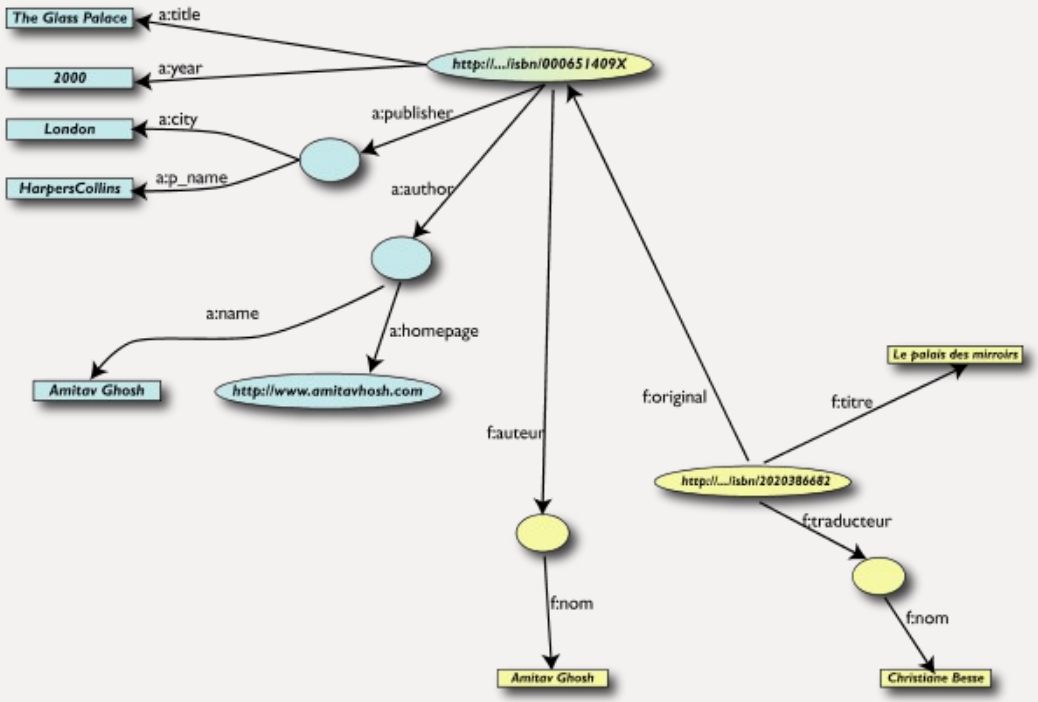


3rd: merge identical resources



Start making queries…

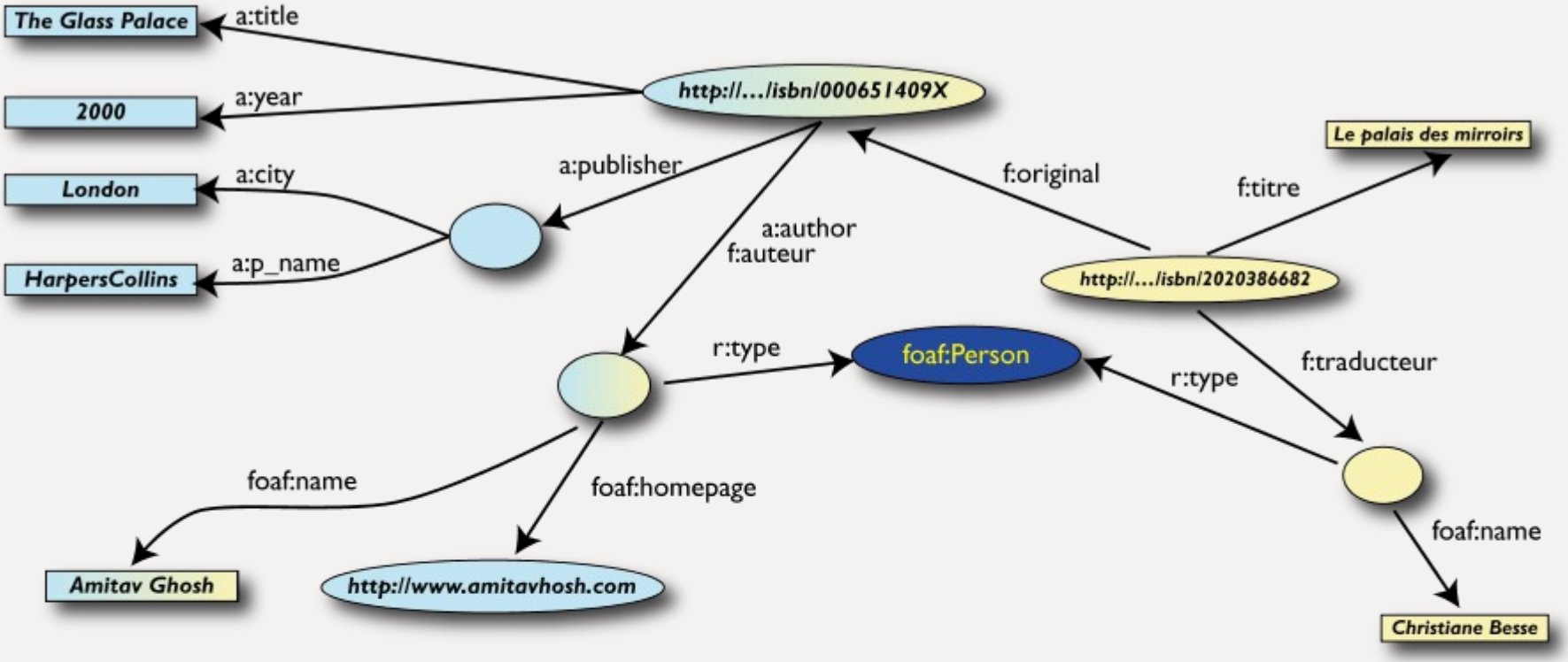
- User of data “F” can now ask queries like:
  - “give me the title of the original”
- This information is not in the dataset “F”…
- …but can be retrieved by merging with dataset “A”!

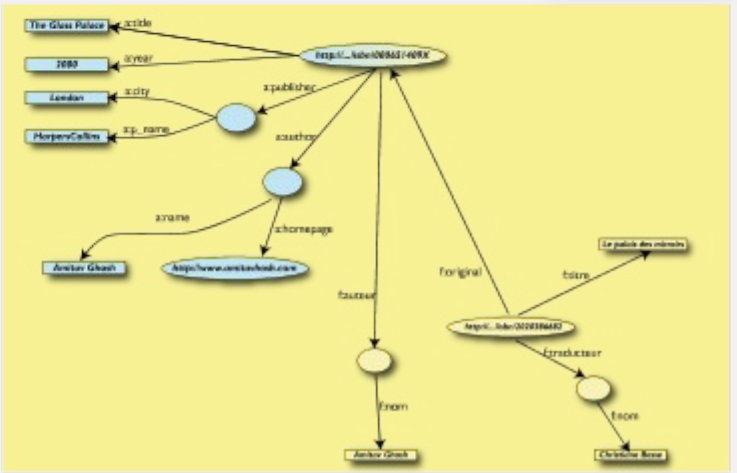


However, more can be achieved…

- We “feel” that **`a:author`** and **`f:auteur`** should be the same
- But an automatic merge does not know that!
- Let us add some extra information to the merged data:
  - **`a:author`** same as **`f:auteur`**
  - both identify a “Person”
  - a term that a community may have already defined:
    - a “Person” is uniquely identified by his/her name and, say, homepage
    - it can be used as a “category” for certain type of resources

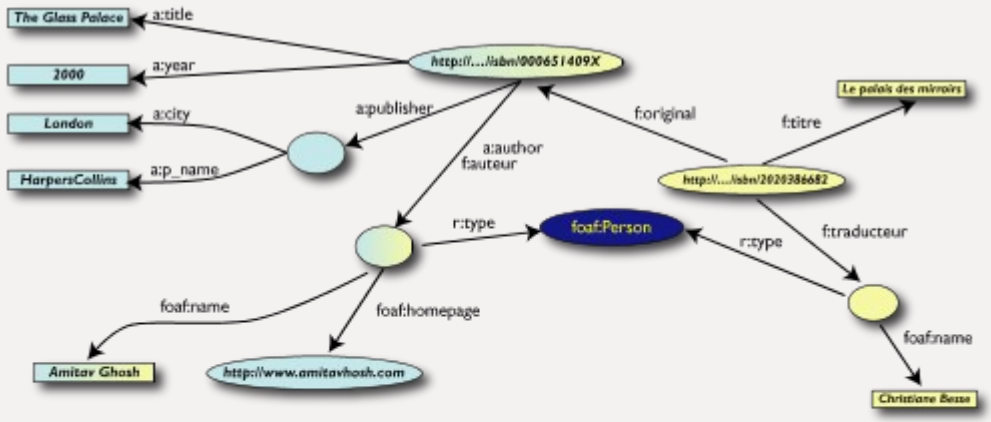
3rd revisited: use the extra knowledge





Start making richer queries!

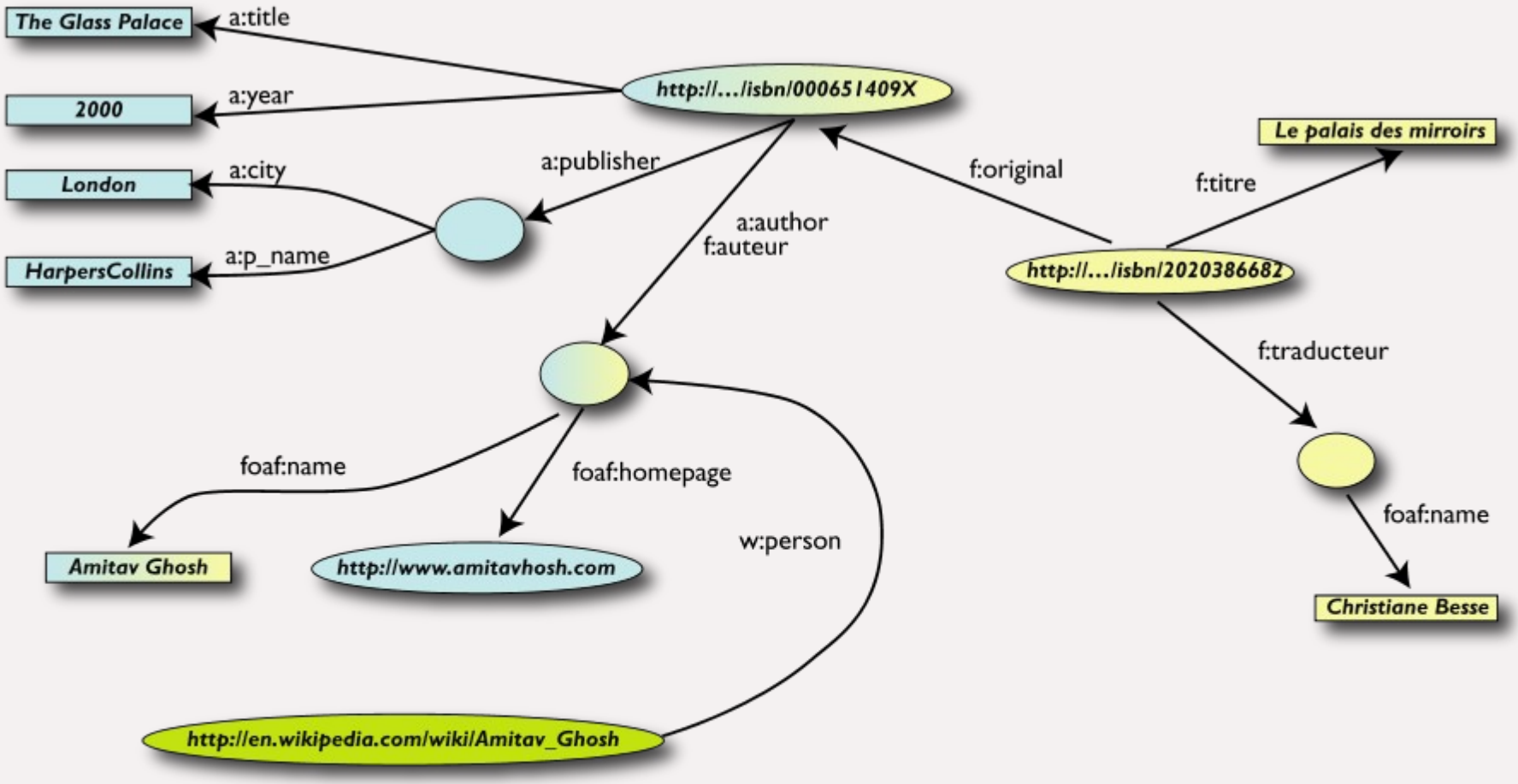
- User of dataset “F” can now query:
  - “give me the home page of the original’s author”
- The information is not in datasets “F” or “A”…
- …but was made available by:
  - merging datasets “A” and datasets “F”
  - adding three simple extra statements as an extra “glue”



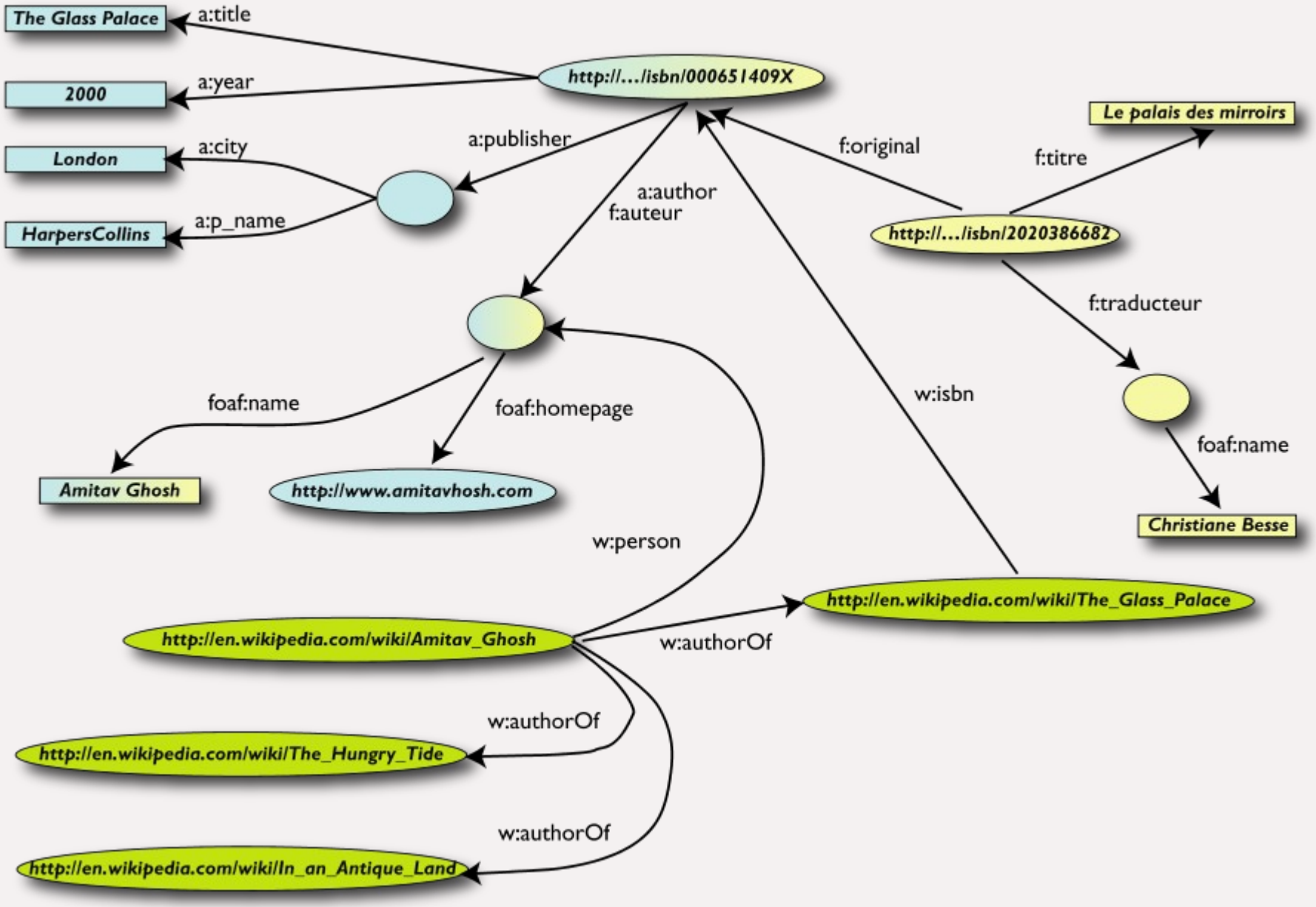
Combine with different datasets

- Via, e.g., the “Person”, the dataset can be combined with other sources
- For example, data in Wikipedia can be extracted using dedicated tools

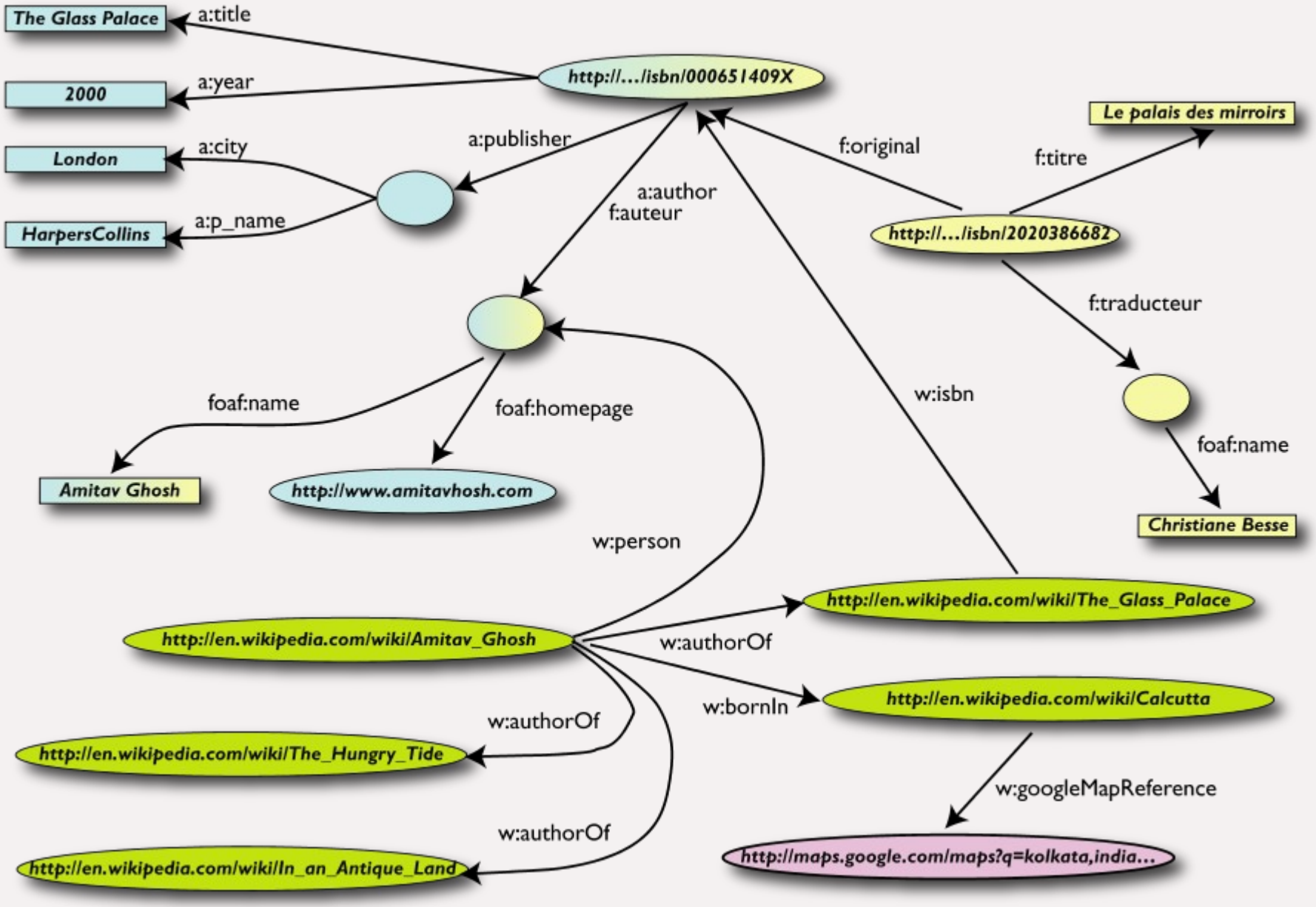
Merge with Wikipedia data



Merge with Wikipedia data



Merge with Wikipedia data



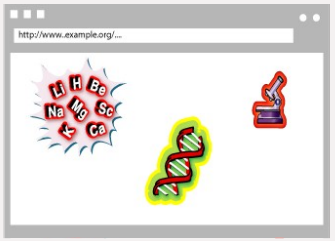
Is that surprising?

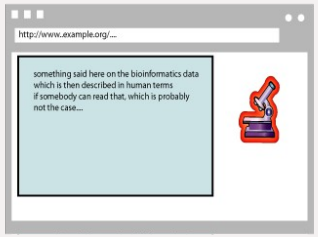
- Maybe but, in fact, no…
- What happened via automatic means is done every day by Web users!
- The difference: a bit of extra rigour so that machines could do this, too

It could become even more powerful

- We could add extra knowledge to the merged datasets
  - e.g., a full classification of various types of library data
  - geographical information
  - etc.
- This is where *ontologies*, extra *rules*, etc, come in
  - ontologies/rule sets can be relatively simple and small, or huge, or anything in between…
- Even more powerful queries can be asked as a result

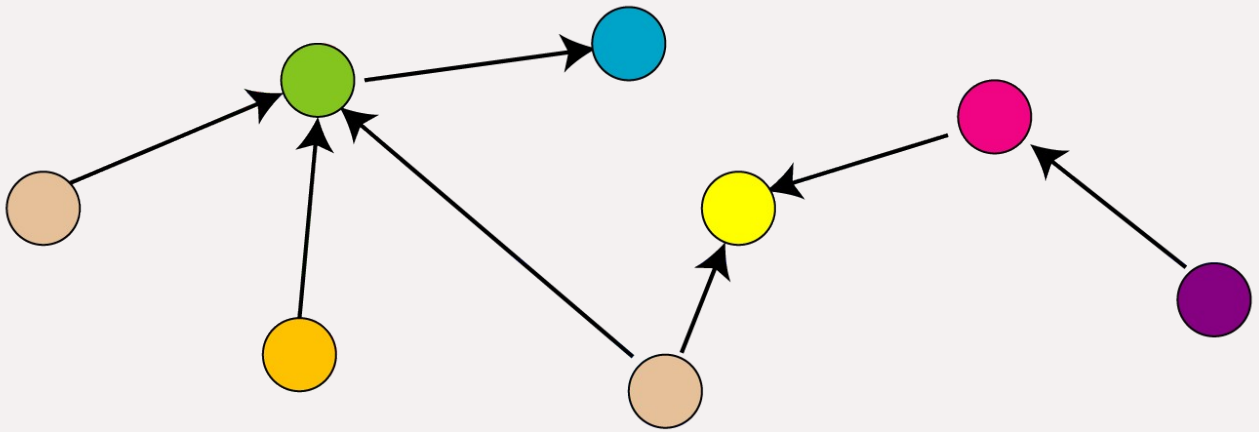
What did we do? (cont)





Applications

Query, Manipulate, etc.



Data represented in abstract format

Map, Expose, etc.

Data in various formats

The network effect

- Through URI-s we can link any data to any data
- The “network effect” is extended to the (Web) data
- “Mashup on steroids” become possible

So where is the Semantic Web?

- The Semantic Web provides technologies to make such integration possible!
- Hopefully you get a full picture at the end of the tutorial…

Basic RDF

RDF triples

- Let us begin to formalize what we did!
  - we “connected” the data…
  - but a simple connection is not enough… it should be named somehow
  - hence the RDF Triples: *a labelled connection between two resources*

RDF triples (cont.)

- An RDF Triple **(s,p,o)** is such that:
  - "**s**", "**p**" are URI-s, ie, resources on the Web; "**o**" is a URI or a literal
    - "**s**", "**p**", and "**o**" stand for "subject", "property", and "object"
  - here is the complete triple:

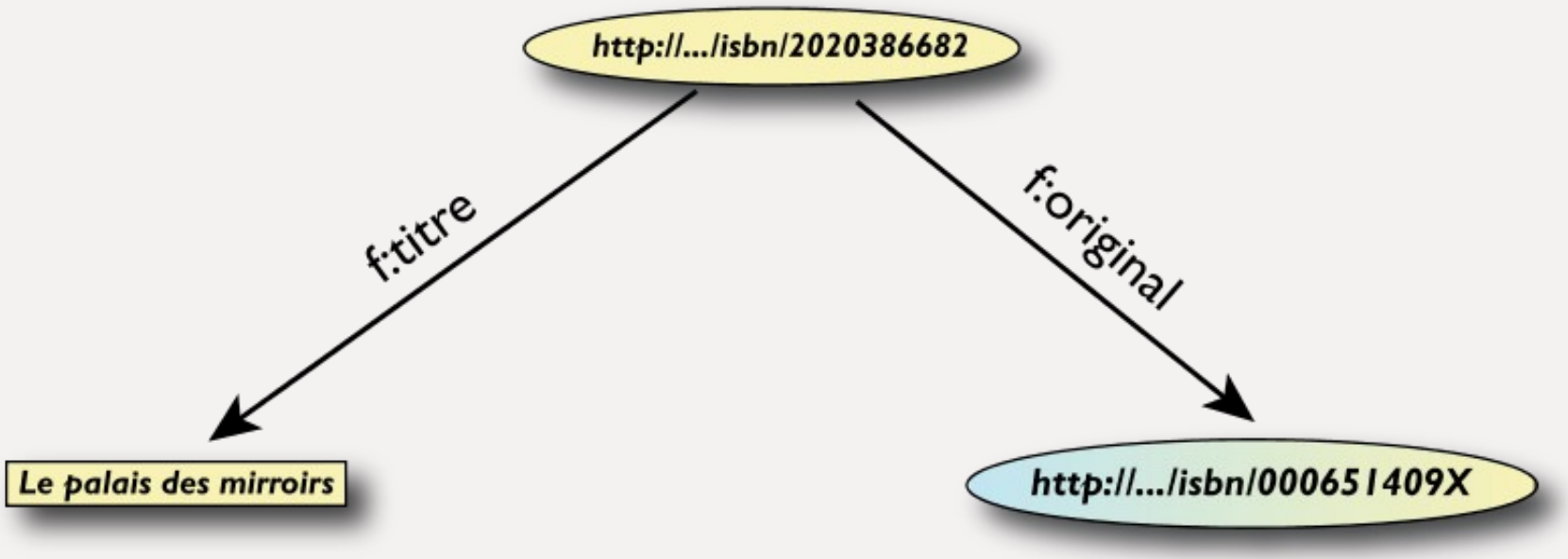
```
(<http://…isbn…6682>, <http://…/original>, <http://…isbn…409X>)
```

- *RDF* is a general model for such triples (with machine readable formats like RDF/XML, Turtle, N3, RXR, …)

RDF triples (cont.)

- Resources can use any URI; it can denote an element within an XML file on the Web, not only a "full" resource, e.g.:
  - `http://www.example.org/file.xml#element(home)`
  - `http://www.example.org/file.html#home`
  - `http://www.example.org/file2.xml#xpath1(//q[@a=b])`
- RDF triples form a directed, labelled graph (best way to think about them!)

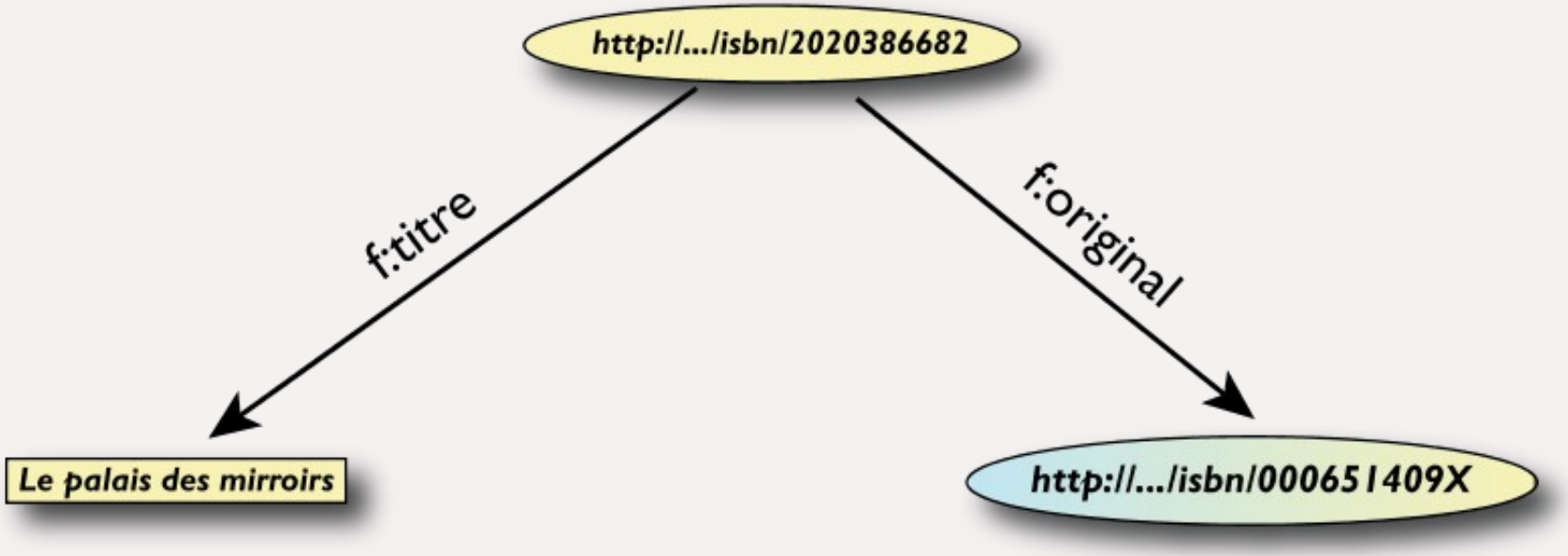
A simple RDF example (in RDF/XML)




```
<rdf:Description rdf:about="http://…/isbn/2020386682">
    <f:titre xml:lang="fr">Le palais des mirroirs</f:titre>
    <f:original rdf:resource="http://…/isbn/000651409X"/>
</rdf:Description>
```

(Note: namespaces are used to simplify the URI-s)

A simple RDF example (in Turtle)



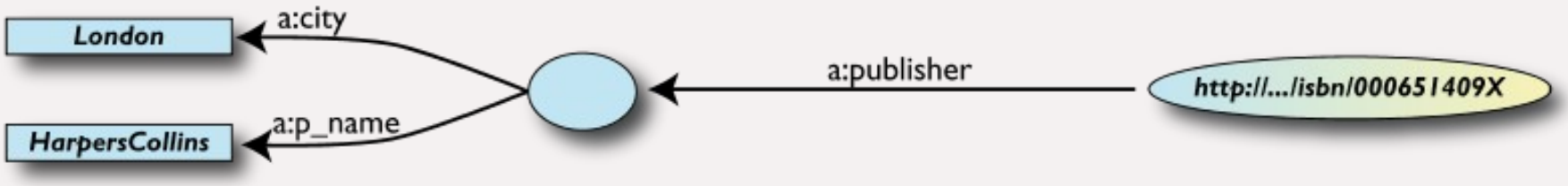

```
<http://…/isbn/2020386682>
    f:titre "Le palais des mirroirs"@fr ;
    f:original <http://…/isbn/000651409X> .
```

URI-s play a fundamental role

- URI-s made the merge possible
- *URI-s ground RDF into the Web*
  - information can be retrieved using existing tools
  - this makes the “Semantic Web”, well… “Semantic *Web*”

“Internal” nodes

- Consider the following statement:
  - “the publisher is a «thing» that has a name and an address”
- Until now, nodes were identified with a URI. But…
- …what is the URI of «thing»?



One solution: create an extra URI

```
<rdf:Description rdf:about="http://…/isbn/000651409X">
   <a:publisher rdf:resource="urn:uuid:f60ffb40-307d-…"/>
</rdf:Description>
<rdf:Description rdf:about="urn:uuid:f60ffb40-307d-…">
   <a:p_name>HarpersCollins</a:p_name>
   <a:city>HarpersCollins</a:city>
</rdf:Description>
```

- The resource will be “visible” on the Web
  - care should be taken to define *unique* URI-s
- Serializations may give syntactic help to define local URI-s

Internal identifier (“blank nodes”)

```
<rdf:Description rdf:about="http://…/isbn/000651409X">
   <a:publisher rdf:nodeID="A234"/>
</rdf:Description>
<rdf:Description rdf:nodeID="A234">
   <a:p_name>HarpersCollins</a:p_name>
   <a:city>HarpersCollins</a:city>
</rdf:Description>
```

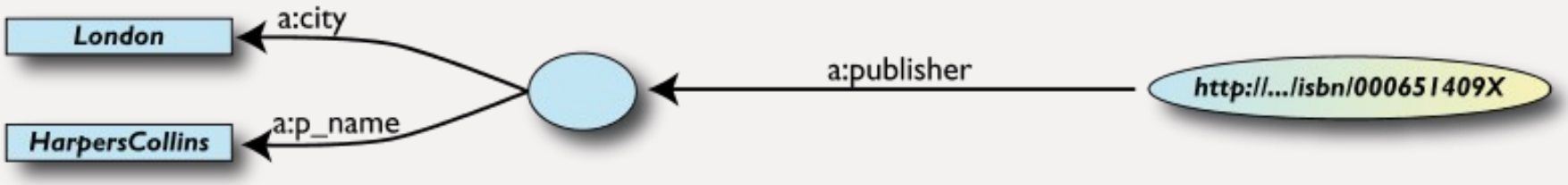
```
<http://…/isbn/2020386682> a:publisher _:A234.
_:A234 a:p_name "HarpersCollins".
```

- Syntax is serialization dependent
- A234 is invisible from outside (it is not a “real” URI!); it is an internal identifier for a resource

Blank nodes: the system can also do it

- Let the system create a “nodeID” internally (you do not really care about the name…). In Turtle:

```
<http://…/isbn/000651409X> a:publisher [
    a:p_name "HarpersCollins";
    …
].
```



Blank nodes: some more remarks

- Blank nodes require attention when merging
  - blanks nodes with identical nodeID-s in *different* graphs are *different*
  - implementations must be careful…
- Many applications prefer not to use blank nodes and define new URI-s “on-the-fly”
  - eg, when triples are in a database
- From a logic point of view, blank nodes represent an “existential” statement
  - “there is a resource such that…”

RDF in programming practice

- For example, using Java+Jena (HP's Bristol Lab):
  - a "Model" object is created
  - the RDF file is parsed and results stored in the Model
  - the Model offers methods to retrieve:
    - triples
    - (property,object) pairs for a specific subject
    - (subject,property) pairs for specific object
    - etc.
  - the rest is conventional programming…
- Similar tools exist in Python, PHP, etc.

Jena example

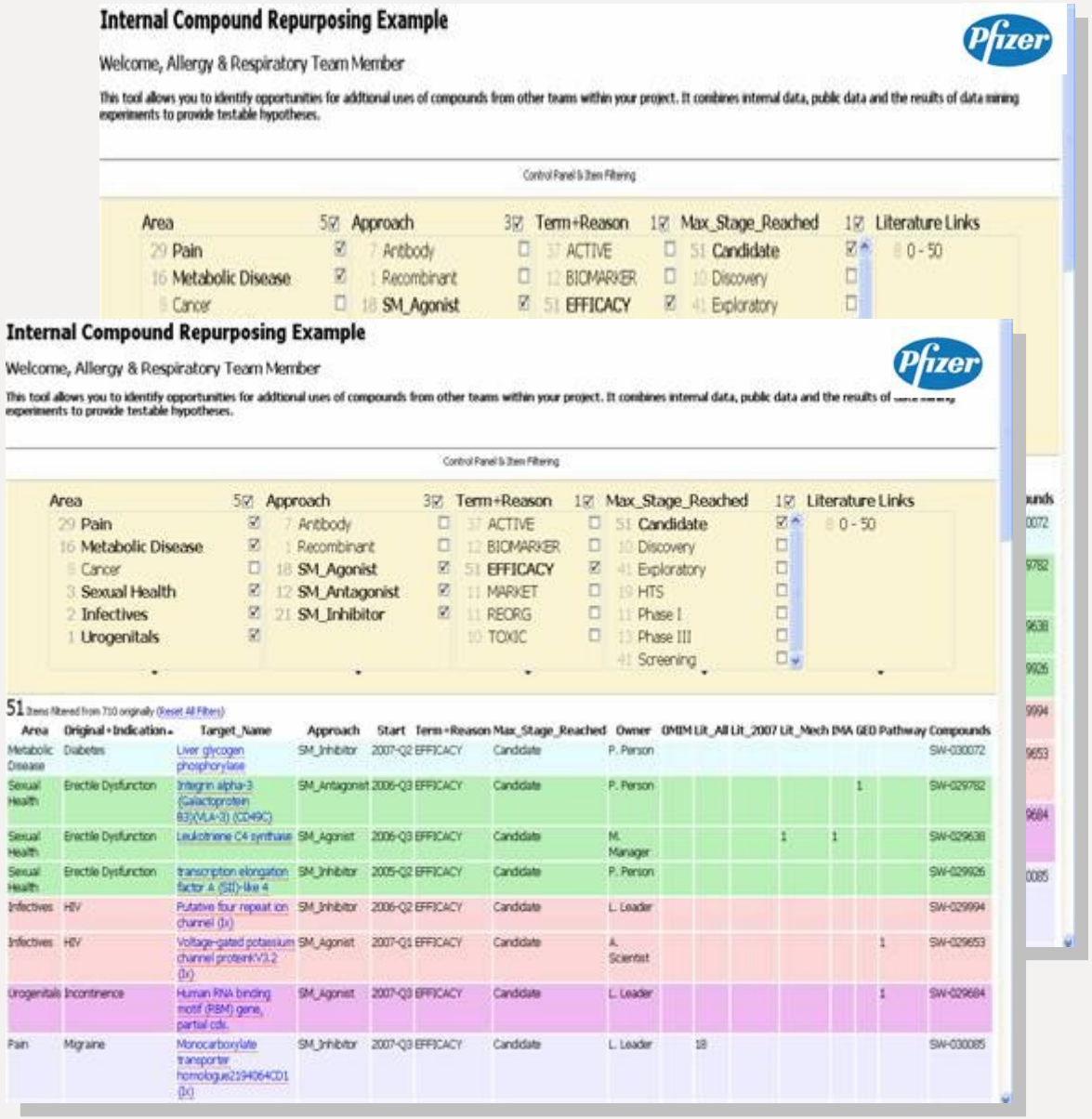
```
// create a model
Model model=new ModelMem();
Resource subject=model.createResource("URI_of_Subject")
// 'in' refers to the input file
model.read(new InputStreamReader(in));
StmtIterator iter=model.listStatements(subject,null,null);
while(iter.hasNext()) {
   st = iter.next();
   p = st.getProperty();
   o = st.getObject();
   do_something(p,o);
}
```

Merge in practice

- Environments merge graphs automatically
  - e.g., in Jena, the Model can load several files
  - the load merges the new statements automatically

A relatively simple RDF application

- Goal: reuse of older experimental data
- Keep data in databases or XML, just export key “fact” as RDF
- Use a faceted browser to visualize and interact with the result



Courtesy of Nigel Wilkinson, Lee Harland, Pfizer Ltd, Melliyal Annamalai, Oracle (SWEO Case Study)

RDF schemas

Need for RDF schemas

- First step towards the “extra knowledge”:
  - define the terms we can use
  - what restrictions apply
  - what extra relationships are there?
- Officially: “RDF Vocabulary Description Language”
  - the term “Schema” is retained for historical reasons…

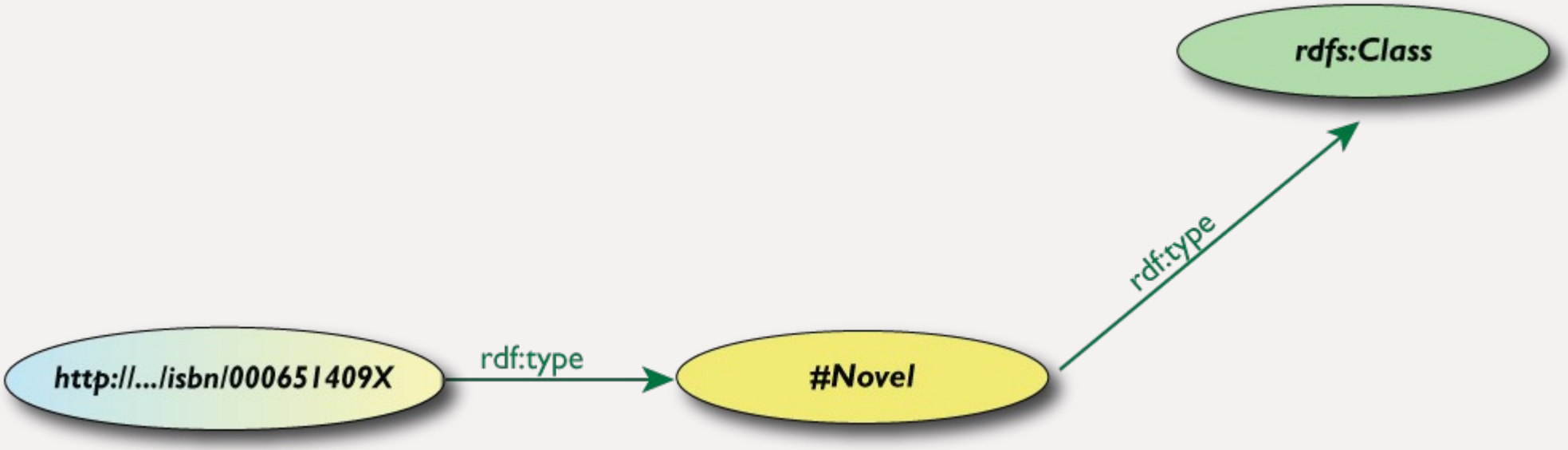
Classes, resources, …

- Think of well known traditional ontologies or taxonomies:
  - use the term “novel”
  - “every novel is a fiction”
  - “«The Glass Palace» is a novel”
  - etc.
- RDFS defines resources and classes:
  - everything in RDF is a “resource”
  - “classes” are also resources, but…
  - …they are also a collection of possible resources (i.e., “individuals”)
    - “fiction”, “novel”, …

Classes, resources, … (cont.)

- Relationships are defined among classes/resources:
  - "typing": an individual belongs to a specific class:
    - "«The Glass Palace» is a novel"
    - to be more precise: "«http://.../**000651409X**» is a novel"
  - "subclassing": *all* instances of one are also the instances of the other ("every novel is a fiction")
- *RDFS formalizes these notions in RDF*

Classes, resources in RDF(S)



- RDFS defines the meaning of these terms
  - (these are all special URI-s, we just use the namespace abbreviation)

Schema example in RDF/XML

- The schema part:

```
<rdf:Description rdf:ID="Novel">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
```

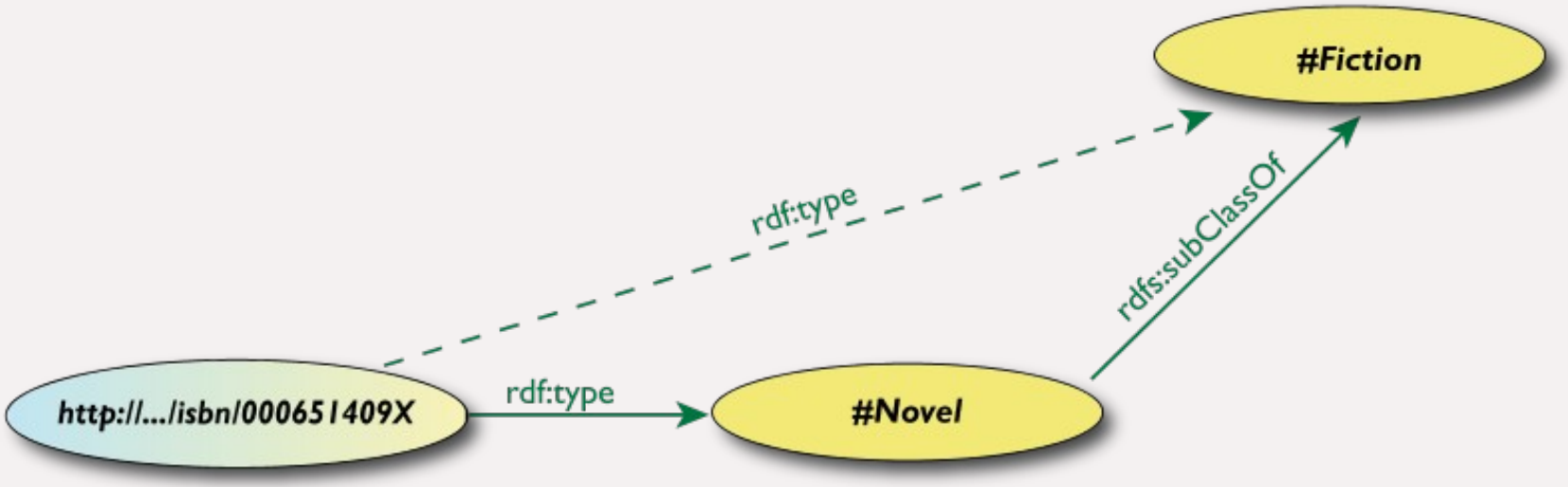
- The RDF data on a specific novel:

```
<rdf:Description rdf:about="http://…/isbn/000651409X">
   <rdf:type rdf:resource="http://…/bookSchema.rdf#Novel"/>
</rdf:Description>
```

On types

- The type information may be very important for applications
  - e.g., it may be used for a categorization of possible nodes
  - probably the most frequently used RDF property…
- (remember the “Person” in our example?)

Inferred properties




```
(<http://…/isbn/000651409X> rdf:type #Fiction)
```

- is not in the original RDF data…
- …but can be inferred from the RDFS rules
- RDFS environments return that triple, too

Inference: let us be formal…

- The RDF Semantics document has a list of (44) *entailment rules*:
  - "if such and such triples are in the graph, add this and this"
  - do that recursively until the graph does not change
- The relevant rule for our example:

```
If:
  uuu rdfs:subClassOf xxx .
  vvv rdf:type uuu .
Then add:
  vvv rdf:type xxx .
```

Properties

- Property is a special class (`rdf:Property`)
  - properties are also resources identified by URI-s
- Properties' range and domain can be specified
  - i.e., what type of resources can serve as object and subject
- There is also a possibility for a "sub-property"
  - all resources bound by the "sub" are also bound by the other

Property specification serialized

- In RDF/XML:

```
<rdf:Property rdf:ID="title">
  <rdfs:domain rdf:resource="#Fiction"/>
  <rdfs:range rdf:resource="http://...#Literal"/>
</rdf:Property>
```

- In Turtle:

```
:title
  rdf:type     rdf:Property;
  rdfs:domain  :Fiction;
  rdfs:range   rdfs:Literal.
```

What does this mean?

- Again, new relations can be deduced. Indeed, if

```
:title
  rdf:type     rdf:Property;
  rdfs:domain  :Fiction;
  rdfs:range   rdfs:Literal.

<http://…/isbn/000651409X> :title "The Glass Palace" .
```

- then the system can *infer* that:

```
<http://…/isbn/000651409X> rdf:type :Fiction .
```

A bit of RDFS can take you far…

- Remember the power of merge?
- We could have used, in our example:
  - `f:auteur` is a subproperty of `a:author` and vice versa (although we will see other ways to do that…)
- Of course, in some cases, more complex knowledge is necessary (see later…)

Vodafone live!

- Integrate various vendors' product descriptions via RDF
  - ring tones, games, wallpapers
  - manage complexity of handsets, binary formats
- A portal is created to offer appropriate content
- Significant increase in content download after the introduction

Courtesy of Kevin Smith, Vodafone Group R&D *(SWEO Case Study)*

Get to RDF(S) data

Simple approach

- Write RDF/XML or Turtle “manually”
- In some cases that is necessary, but it really does not scale…

RDF can also be extracted/generated

- Use intelligent “scrapers” or “wrappers” to extract a structure (hence RDF) from a Web pages or XML files…
- … and then generate RDF automatically (e.g., via an XSLT script)

Formalizing the scraper approach: GRDDL

- GRDDL formalizes scrapers:

```
<html xmlns="http://www.w3.org/1999/">
  <head profile="http://www.w3.org/2003/g/data-view">
    <title>Some Document</title>
    <link rel="transformation" href="http:…/dc-extract.xsl"/>
    <meta name="DC.Subject" content="Some subject"/>
    ...
  </head>
  ...
  <span class="date">2006-01-02</span>
  ...
</html>
```

- yields, via **dc-extract.xsl**:

```
<rdf:Description rdf:about="…">
  <dc:subject>Some subject</dc:subject>
  <dc:date>2006-01-02</dc:date>
</rdf:Description>
```

GRDDL

- The transformation itself has to be provided for each set of conventions (meta-s, class id-s, etc…)
- A “bridge” to “microformats”
- A method to get data in other formats to RDF (e.g., XBRL)

Another solution: RDFa

- For example:

```
<div about="http://uri.to.newsitem">
  <span property="dc:date">March 23, 2004</span>
  <span property="dc:title">Rollers hit casino for £1.3m</span>
  By <span property="dc:creator">Steve Bird</span>. See
  <a href="http://www.a.b.c/d.avi" rel="dcmtype:MovingImage">
  also video footage</a>…
</div>
```

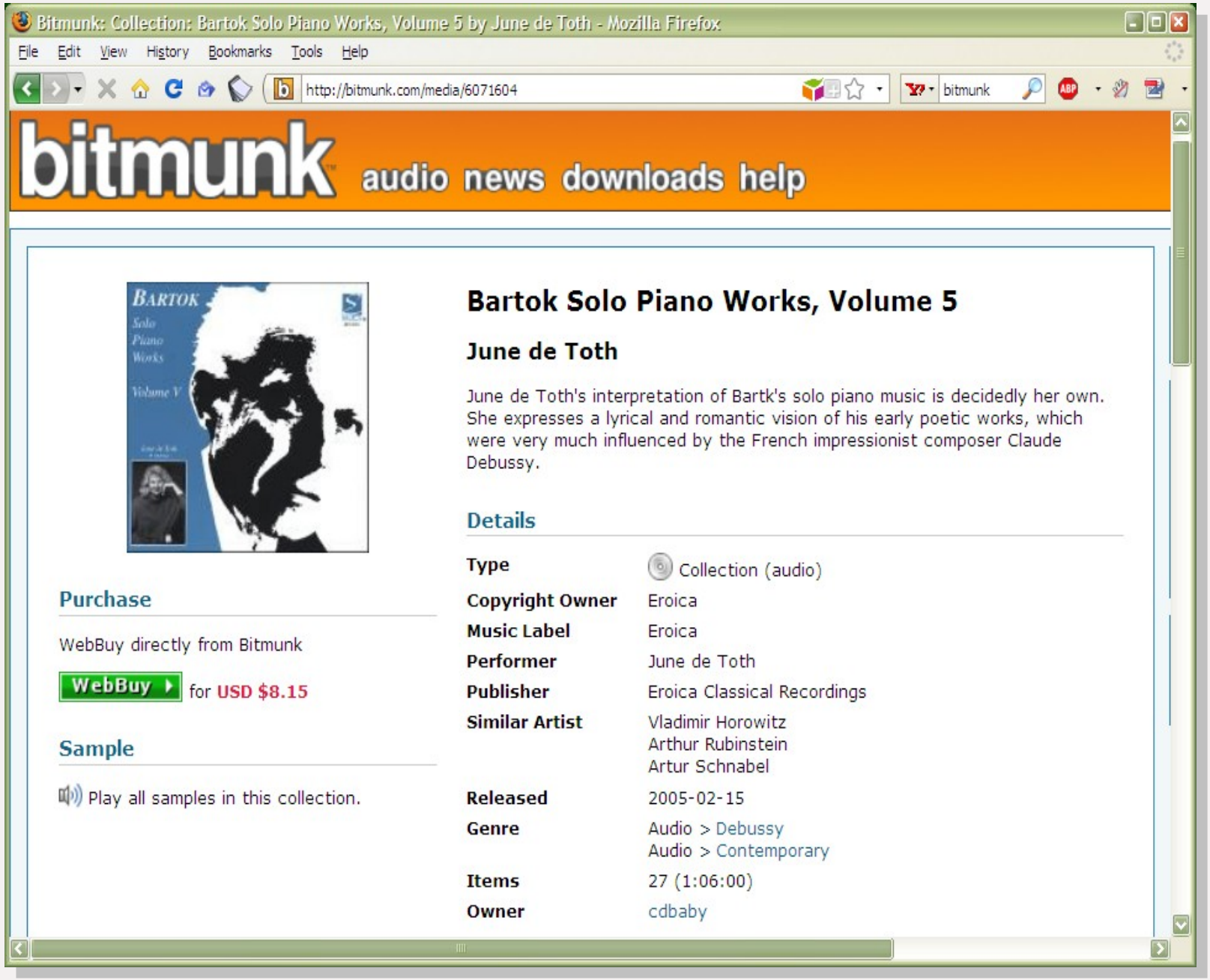
- yields:

```
<http://uri.to.newsitem>
  dc:date                "March 23, 2004";
  dc:title               "Rollers hit casino for £1.3m;
  dc:creator             "Steve Bird";
  dcmtype:MovingImage <http://www.a.b.c/d.avi>.
```

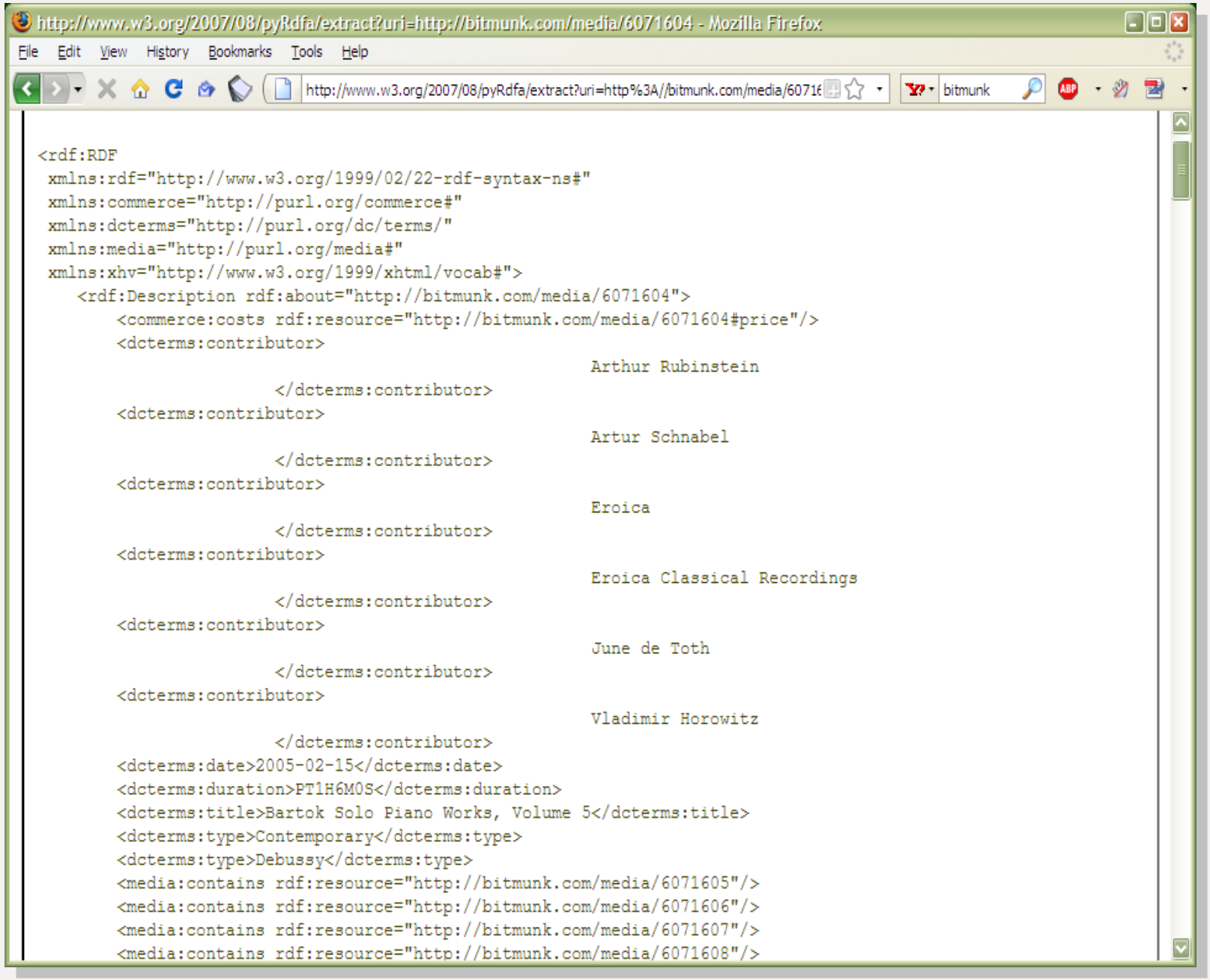
RDFa (cont.)

- RDFa extends (X)HTML a bit by:
  - defining general attributes to add data to any elements
  - an almost complete “serialization” of RDF in XHTML
- It is a bit like the microformats/GRDDL approach but with more rigour and fully generic

A typical RDFa usage: the visible page



… and what is inside as metadata



Yahoo's SearchMonkey

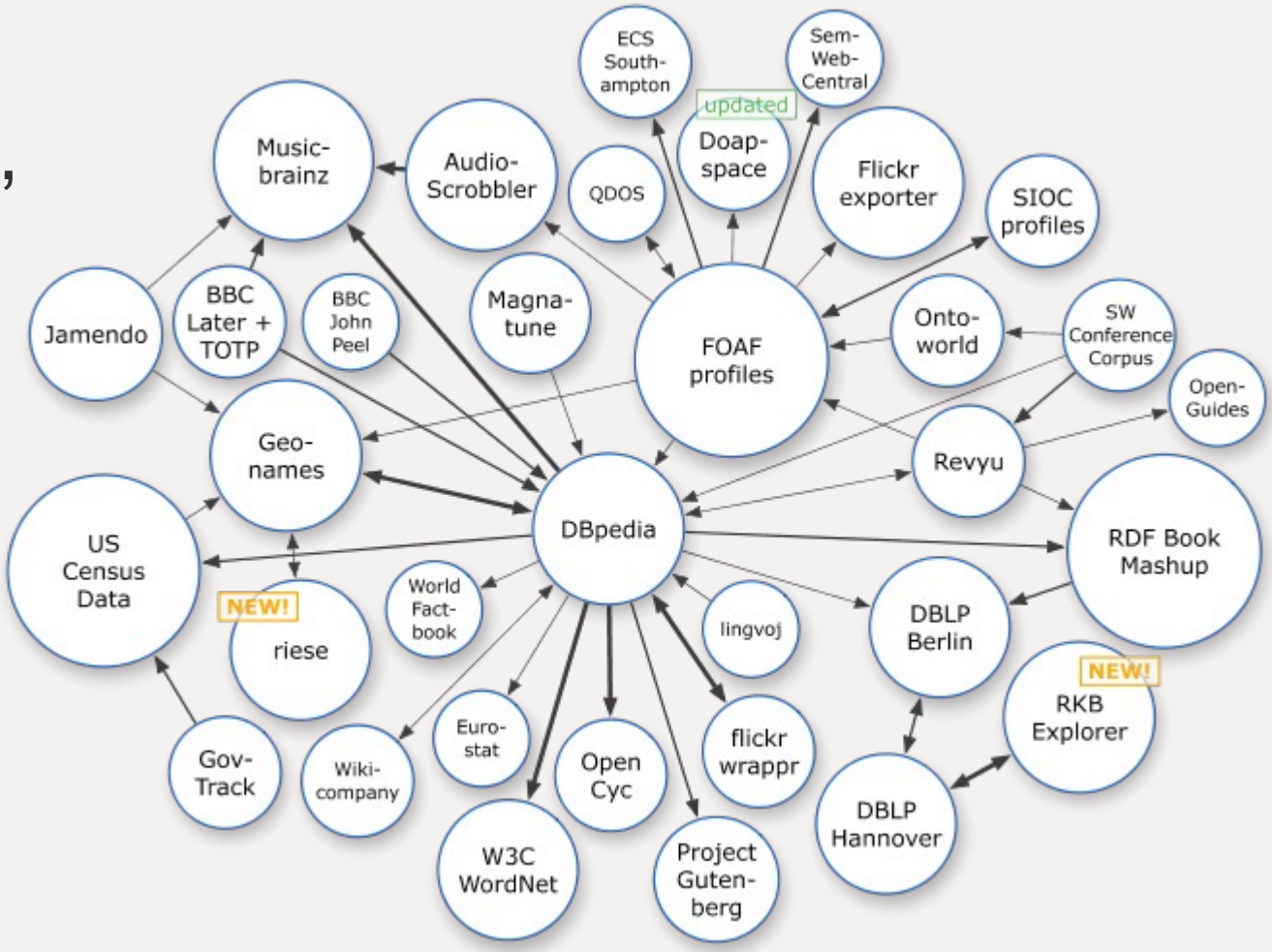
- Search results may be customized via small applications using content metadata in, eg, RDFa
- Users can customize their search pages

Bridges to relational databases

- Data on the Web are mostly stored in databases
- “Bridges” are being defined:
  - a layer between RDF and the relational data
    - RDB tables are “mapped” to RDF graphs, possibly on the fly
    - different mapping approaches are being used
  - a number RDB systems offer this facility already (eg, Oracle, OpenLink, …)
- Work for a survey on mapping techniques at W3C

Linking Open Data Project

- Goal: “expose” open datasets in RDF
- *Set RDF links among the data items* from different datasets
- Billions triples, millions of “links”



DBpedia: Extracting structured data from Wikipedia

```
http://en.wikipedia.org/wiki/Kolkata
```

```
<http://dbpedia.org/resource/Kolkata>
  dbpedia:native_name "Kolkata (Calcutta)"@en;
  dbpedia:altitude "9";
  dbpedia:populationTotal  "4580544";
  dbpedia:population_metro "14681589";
  geo:lat "22.56970024108887"^^xsd:float;
 ...
```

Kolkata (Chutiyon ka Shahar)

West Bengal • India

Victoria Memorial

Coordinates:	22.5697, 88.3697
Time zone	IST (UTC+5:30)
Area • Elevation	1,480 km² (571 sq mi) • 9 m (30 ft)
District(s)	Calcutta †
Population • Density • Metro	14,681,589 (2001) • 9,920/km² (25,693/sq mi) • 4,580,544
Mayor	Bikash Ranjan Bhattacharya
Codes • Pincode • Telephone	 • 700 xxx • +91 (33)

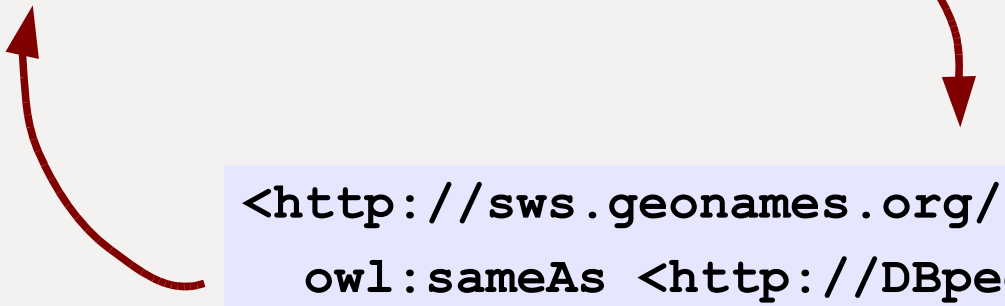
Website: www.kolkatamycity.com

† The Kolkata urban agglomeration also includes portions of North 24 Parganas and South 24 Parganas districts.

Automatic links among open datasets

```
<http://dbpedia.org/resource/Kolkata>
  owl:sameAs <http://sws.geonames.org/1275004/>;
 ...
```

DBpedia

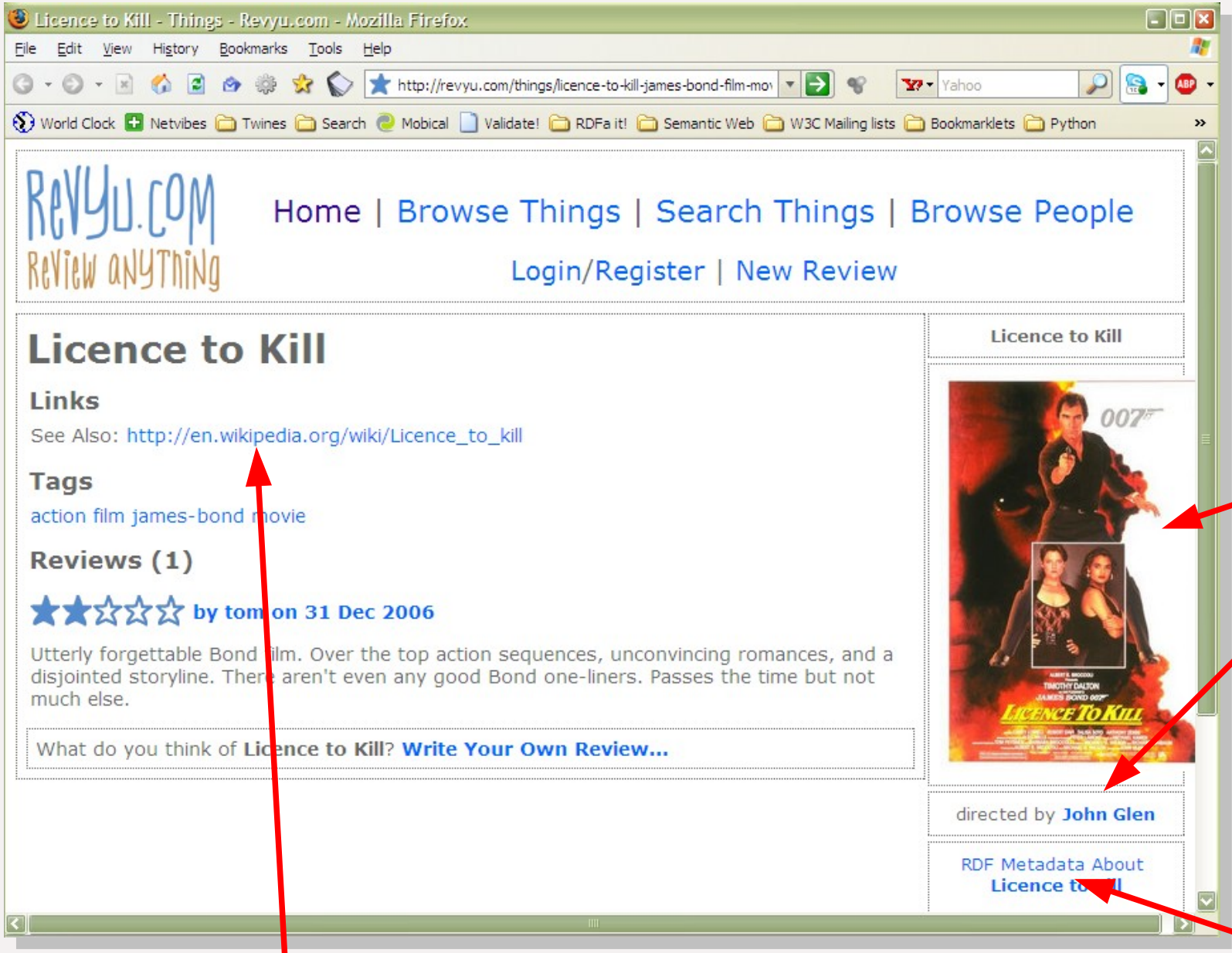


Geonames

```
<http://sws.geonames.org/1275004/>
  owl:sameAs <http://DBpedia.org/resource/Kolkata>
  wgs84_pos:lat "22.5697222";
  wgs84_pos:long "88.3697222";
  sws:population "4631392"
 ...
```

Processors can switch automatically from one to the other…

Example usage: “Review Anything”



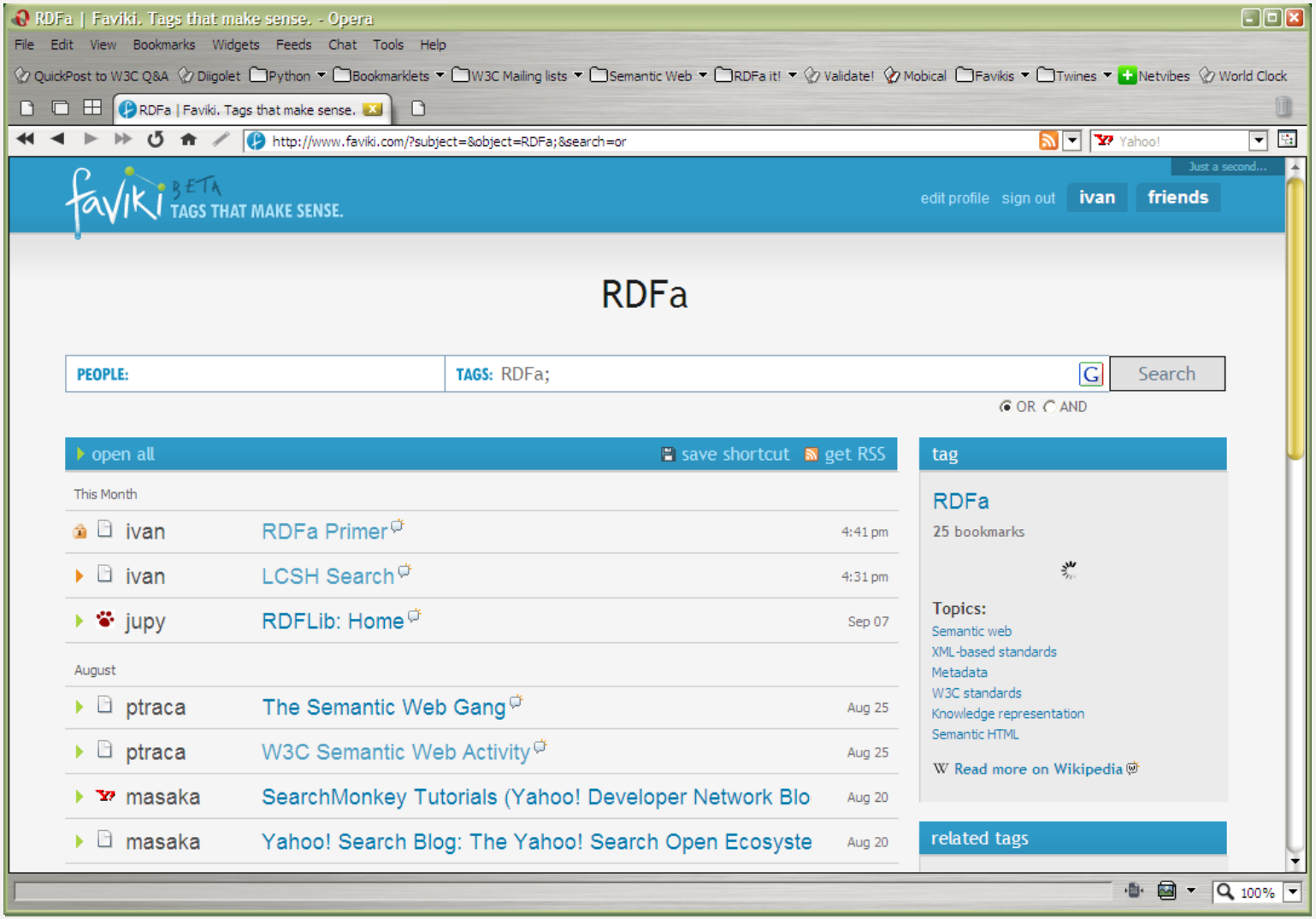
enhance output with linked data

data in RDF

links to, eg, (DB/Wiki)Pedia

Faviki: social bookmarking with Wiki tagging

- Tag bookmarks via Wikipedia terms/DBpedia URIs
- Helps disambiguating tag usage



RDF data access, a.k.a. query (SPARQL)

RDF data access

- How do I *query* the RDF data?
  - e.g., how do I get to the DBpedia data?

Querying RDF graphs

- Remember the Jena idiom:

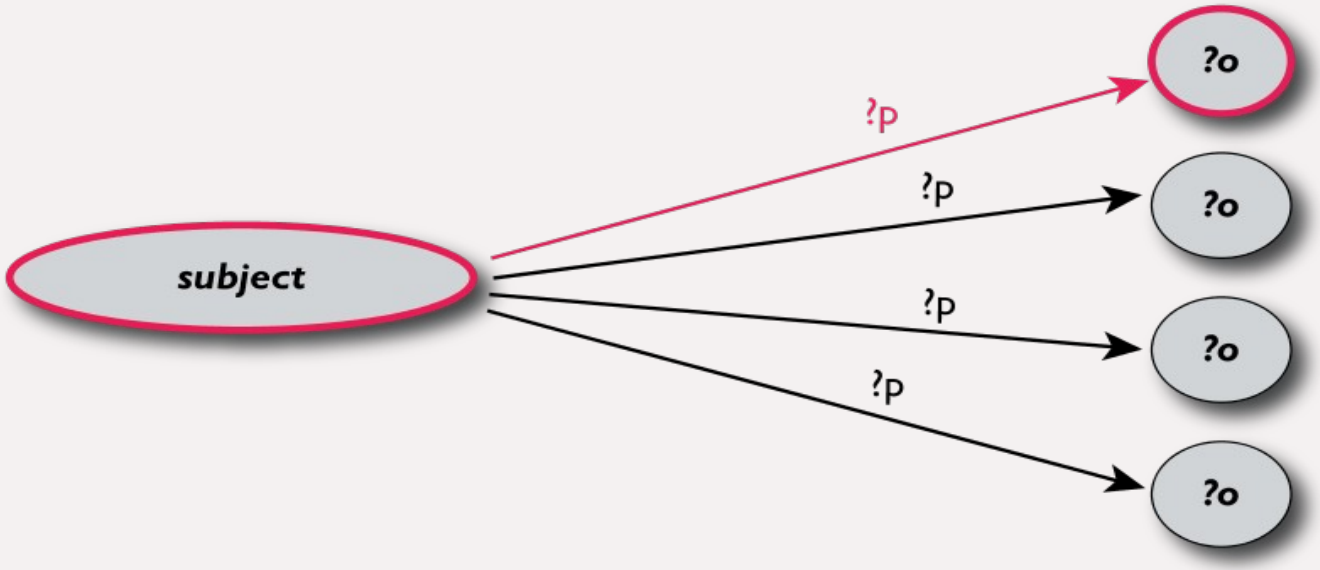
```
StmtIterator iter=model.listStatements(subject,null,null);
while(iter.hasNext()) {
    st = iter.next();
    p = st.getProperty(); o = st.getObject();
    do_something(p,o);
```

- In practice, more complex queries into the RDF data are necessary
  - something like: “give me the (a,b) pair of resources, for which there is an x such that (x parent a) and (b brother x) holds” (ie, return the uncles)
  - these rules may become quite complex
- The goal of SPARQL (Query Language for RDF)

Analyze the Jena example

```
StmtIterator iter=model.listStatements(subject,null,null);
while(iter.hasNext()) {
    st = iter.next();
    p = st.getProperty(); o = st.getObject();
    do_something(p,o);
```

- The `(subject,?p,?o)` is a *pattern* for what we are looking for (with `?p` and `?o` as "unknowns")



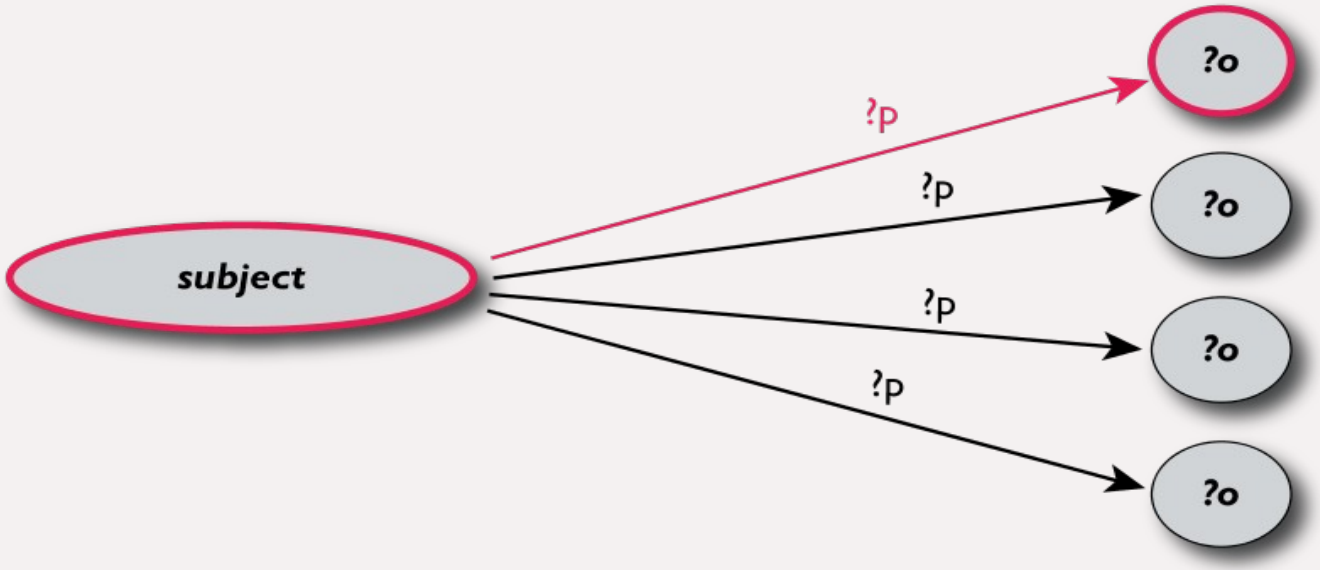
General: graph patterns

- The fundamental idea: use graph patterns
  - the pattern contains unbound symbols
  - by binding the symbols, subgraphs of the RDF graph are selected
  - if there is such a selection, the query returns bound resources

Our Jena example in SPARQL

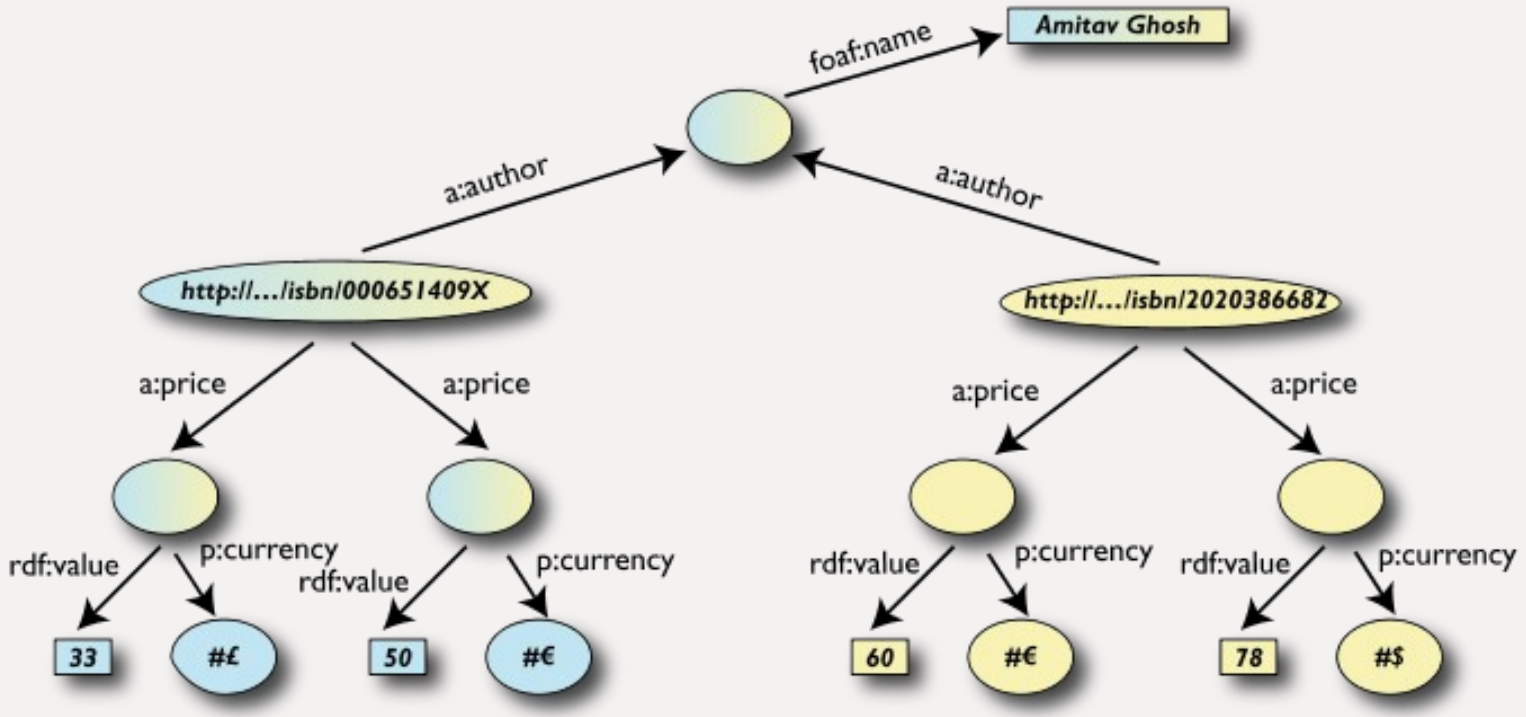
```
SELECT ?p ?o
WHERE {subject ?p ?o}
```

- The triples in **`WHERE`** define the graph pattern, with **`?p`** and **`?o`** "unbound" symbols
- The query returns *all* **`p`**,**`o`** pairs



Simple SPARQL example

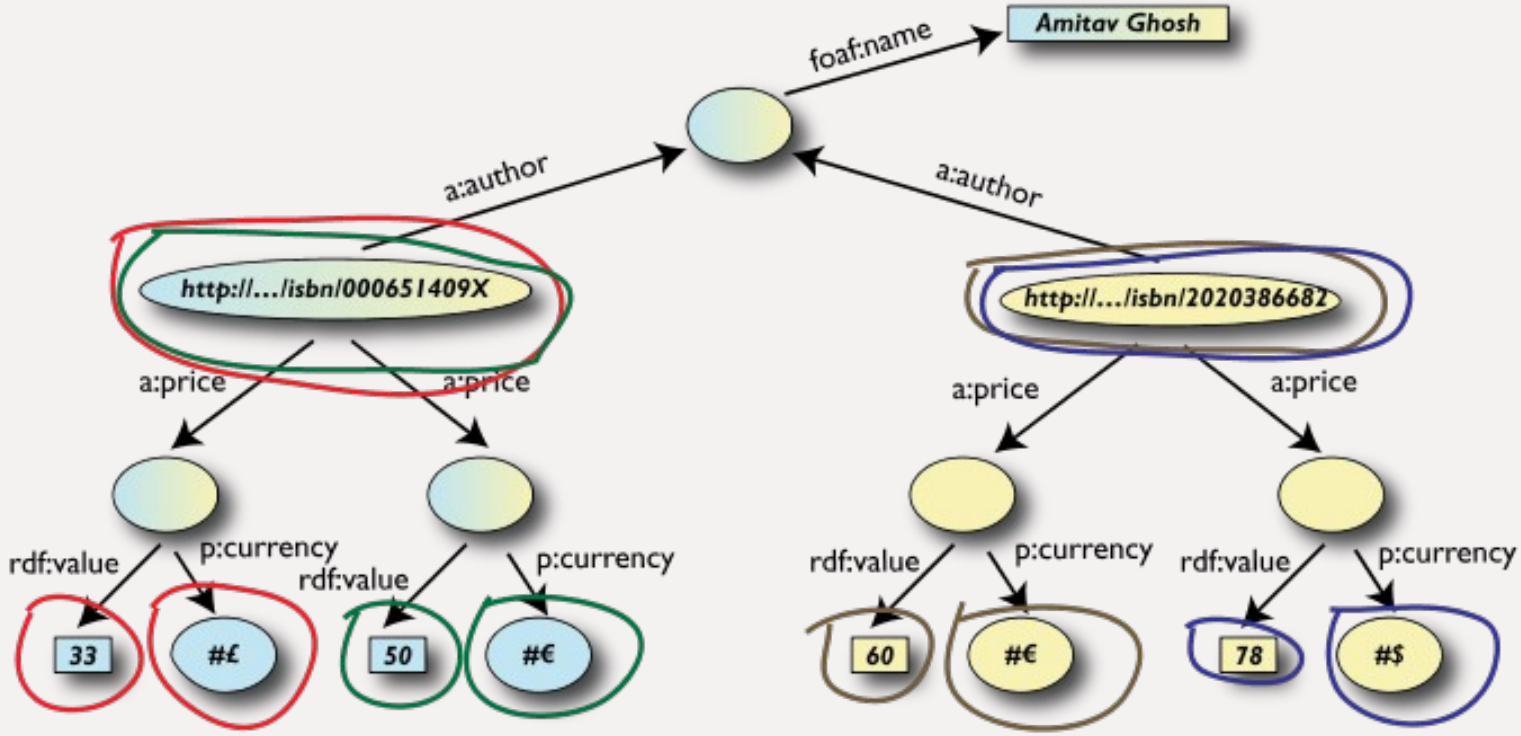
```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



Simple SPARQL example

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```

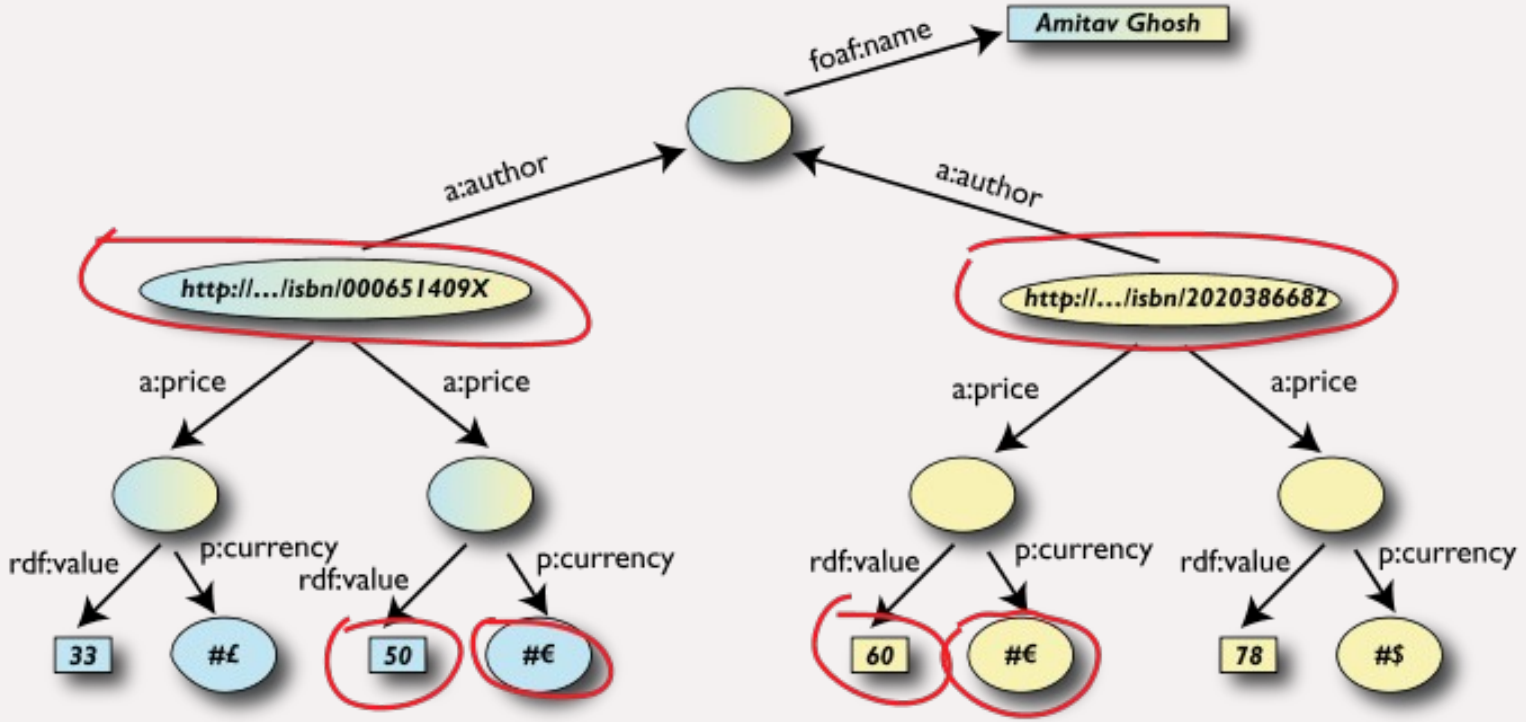
- Returns:
  [[<..49X>,33,£], [<..49X>,50,€], [<..6682>,60,€], [<..6682>,78,$]]



Pattern constraints

```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        FILTER(?currency == € }
```

- Returns: [[<..409X>,50,€], [<..6682>,60,€]]



Some other SPARQL features

- Some of the patterns may be optional
- Limit the number of returned results; remove duplicates, sort them, …
- Specify several data sources (via URI-s) within the query (essentially, a merge!)
- *Construct* a graph combining a separate pattern and the query results

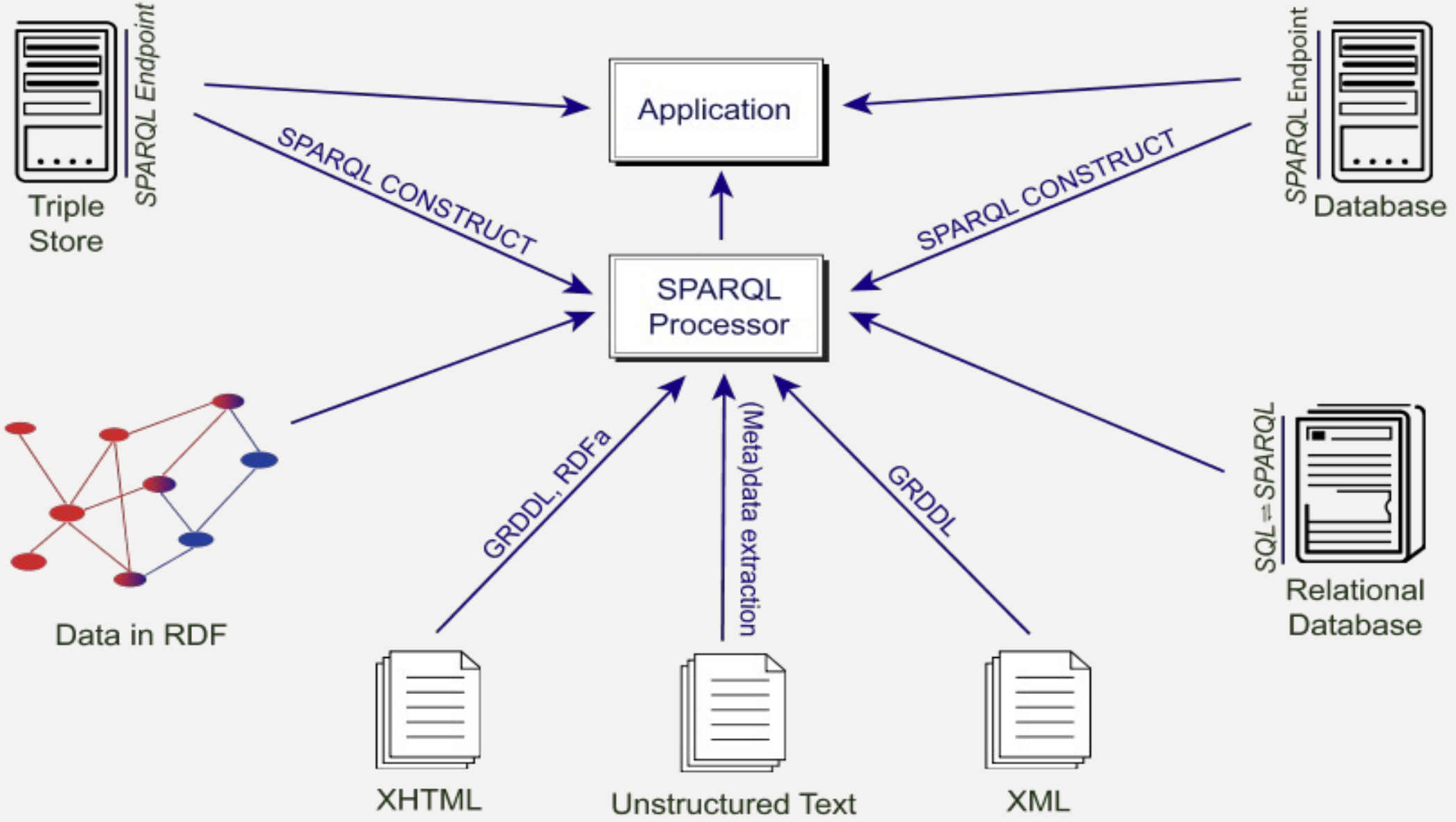
SPARQL usage in practice

- SPARQL is usually used over the network
  - separate documents define the protocol and the result format
    - SPARQL Protocol for RDF with HTTP and SOAP bindings
    - SPARQL results in XML or JSON formats
- Big datasets usually offer “SPARQL endpoints” using this protocol

A word of warning on SPARQL…

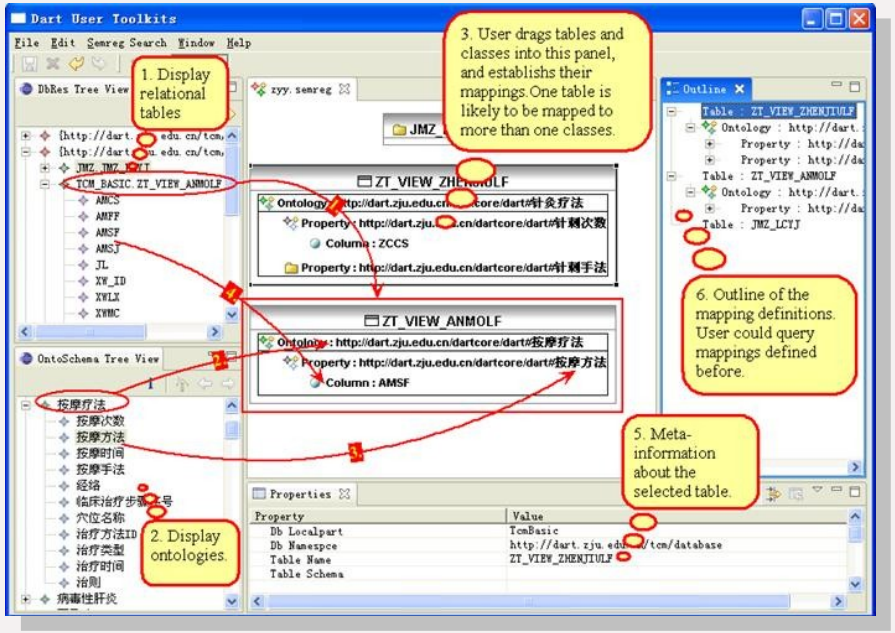
- Some features are missing
  - control and/or description on the entailment regimes of the triple store (RDFS? OWL-DL? OWL-Lite? …)
  - modify the triple store
  - querying collections or containers may be complicated
  - no functions for sum, average, min, max, …
  - ways of aggregating queries
  - …
- Delayed for a next version…

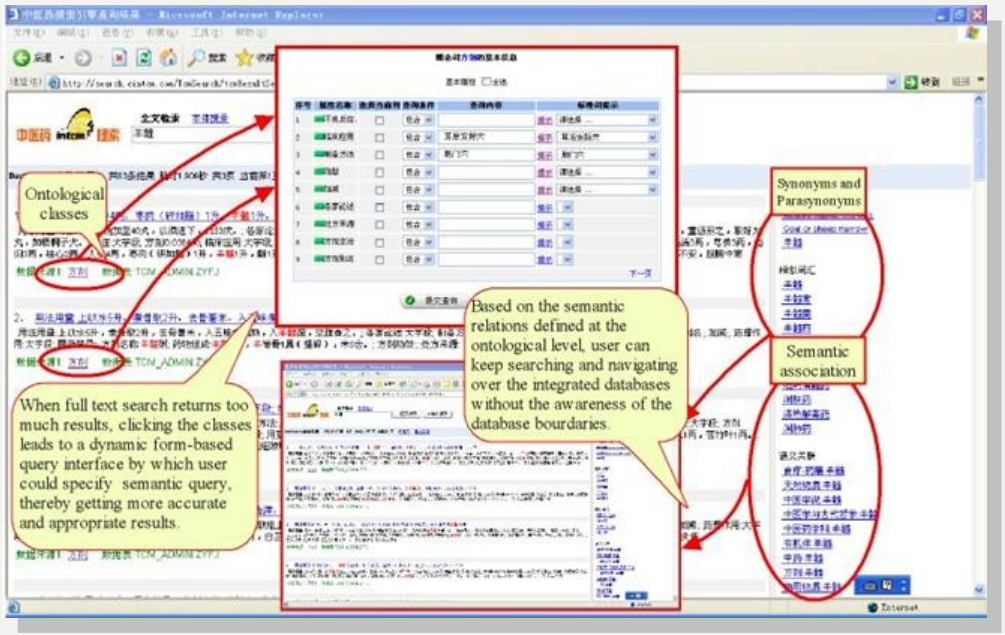
SPARQL as a unifying point



Integrate knowledge for Chinese Medicine

- Integration of a large number of TCM databases
  - around 80 databases, around 200,000 records each
- A visual tool to map databases to the semantic layer using a specialized ontology
- Form based query interface for end users





Courtesy of Huajun Chen, Zhejiang University, (SWEO Case Study)

Ontologies (OWL)

Ontologies

- RDFS is useful, but does not solve all possible requirements
- Complex applications may want more possibilities:
  - characterization of properties
  - identification of objects with different URI-s
  - disjointness or equivalence of classes
  - construct classes, not only name them
  - more complex classification schemes
  - can a program reason about some terms? E.g.:
    - “if «Person» resources «A» and «B» have the same «`foaf:email`» property, then «A» and «B» are identical”
  - etc.

Ontologies (cont.)

- The term *ontologies* is used in this respect:

"defines the concepts and relationships used to describe and represent an area of knowledge"

- Ie, there is a need for Web Ontology Language(s)
  - RDFS can be considered as a simple ontology language
- Languages should be a compromise between
  - rich semantics for meaningful applications
  - feasibility, implementability

Web Ontology Language = OWL

- OWL is an extra layer, a bit like RDF Schemas
  - own namespace, own terms
  - it relies on RDF Schemas
- It is a separate recommendation
- There is an active W3C Working Group working on *extensions* of the current standards
  - the new version will be called “OWL 2”
  - in what follows, some features will be referred to as “may come in future”, i.e., under consideration by that group

OWL is complex…

- OWL is a large set of additional terms
- We will not cover the whole thing here…

First some simple features

Term equivalence

- For classes:
  - **`owl:equivalentClass`**: two classes have the same individuals
  - **`owl:disjointWith`**: no individuals in common
- For properties:
  - **`owl:equivalentProperty`**
    - remember the **`a:author`** vs. **`f:auteur`**?
- For individuals:
  - **`owl:sameAs`**: two URIs refer to the same concept ("individual")
  - **`owl:differentFrom`**: negation of **`owl:sameAs`**

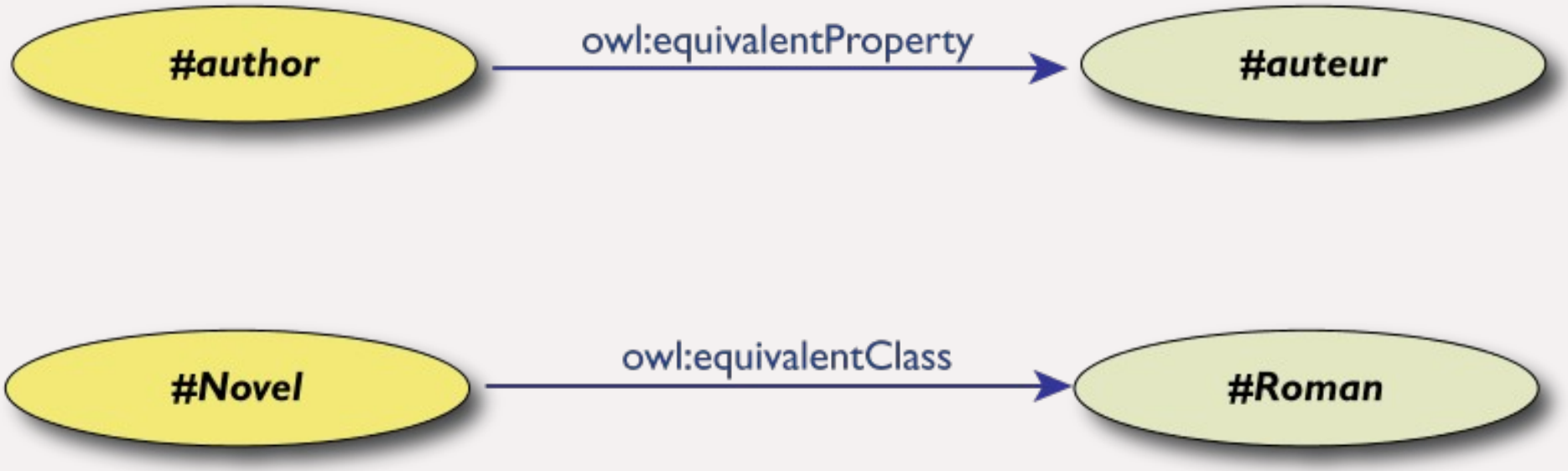
Typical usage of `owl:sameAs`

- Linking our example of Kolkata from one data set (DBpedia) to the other (Geonames):

```
<http://dbpedia.org/resource/Kolkata>
  owl.sameAs <http://sws.geonames.org/1275004/>;
```

- This is the main mechanism of “Linking” in the Linking Open Data project

Other example: connecting to French

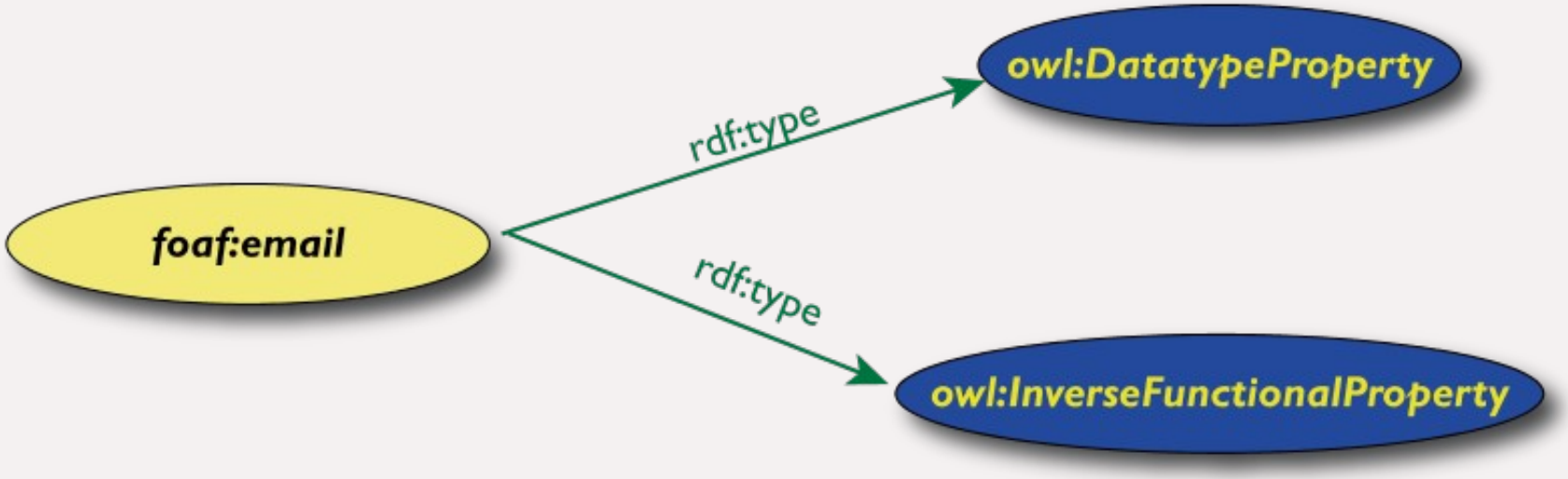


Property characterization

- In OWL, one can characterize the behaviour of properties (symmetric, transitive, functional, inverse functional…)
- OWL also separates *data* and *object* properties
  - “datatype property” means that its range are typed literals

Characterization example

- "`foaf:email`" is inverse functional (i.e., two different subjects cannot have identical objects)



What this means is…

- If the following holds in our triples:

```
:email rdf:type owl:InverseFunctionalProperty.
<A> :email "mailto:a@b.c".
<B> :email "mailto:a@b.c".
```

- then the following holds, too:

```
<A> owl:sameAs <B>.
```

- I.e., *new relationships* were discovered again (beyond what RDFS could do)

Other property characterizations

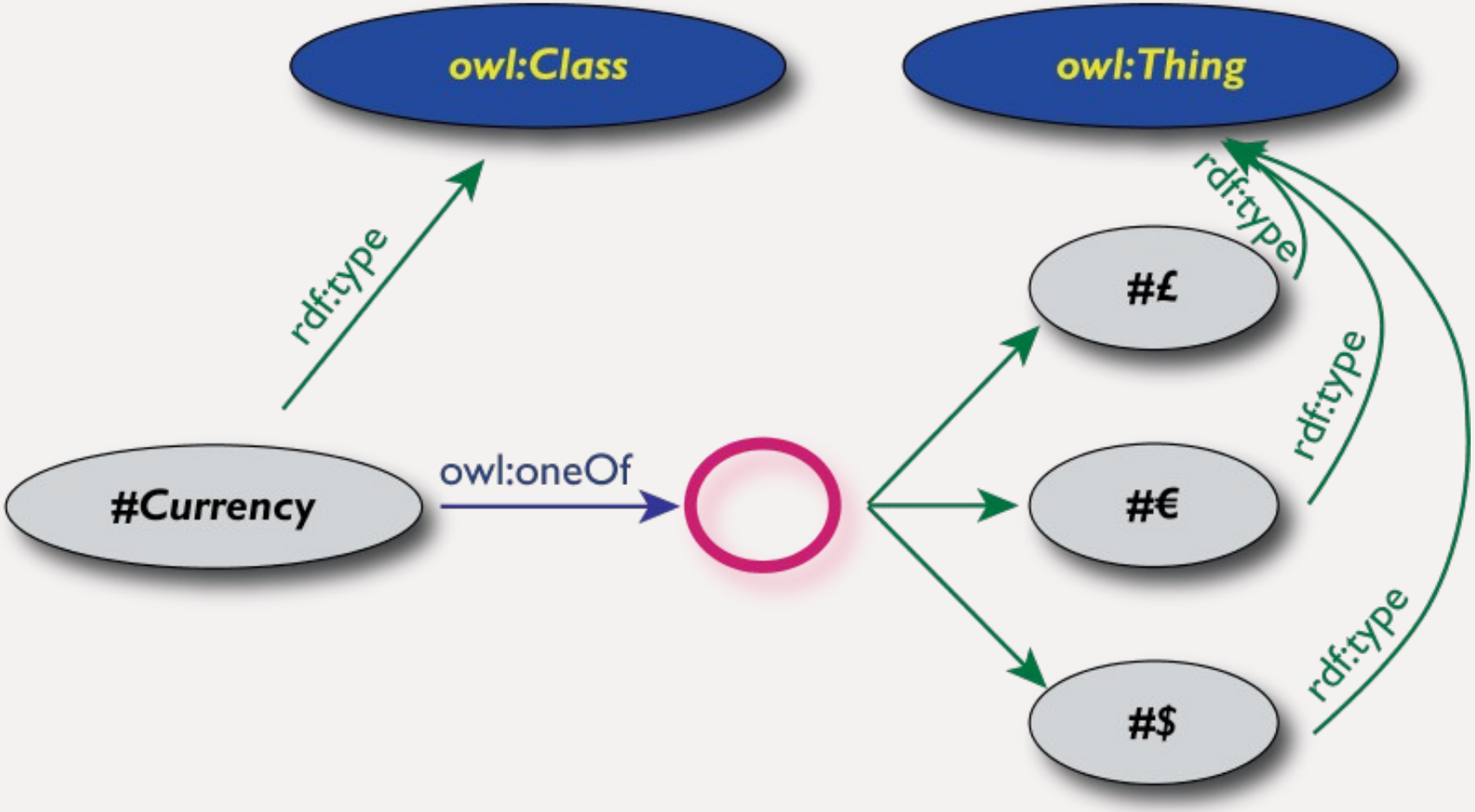
- Functional property ("`owl:FunctionalProperty`")
- Transitive property ("`owl:TransitiveProperty`")
- Symmetric property ("`owl:SymmetricProperty`")
- Inverse of another property ("`owl:inverseOf`")
- May come in future:
  - reflexive and irreflexive object properties
  - specify that properties are "disjoint"

Classes in OWL

- In RDFS, you can subclass existing classes… that's all
- In OWL, you can *construct* classes from existing ones:
  - enumerate its content
  - through intersection, union, complement
  - etc
- OWL makes a stronger distinction between *classes* and *individuals*
  - referring to its own **`Class`** and to "**`Thing`**", respectively

OWL classes can be “enumerated”

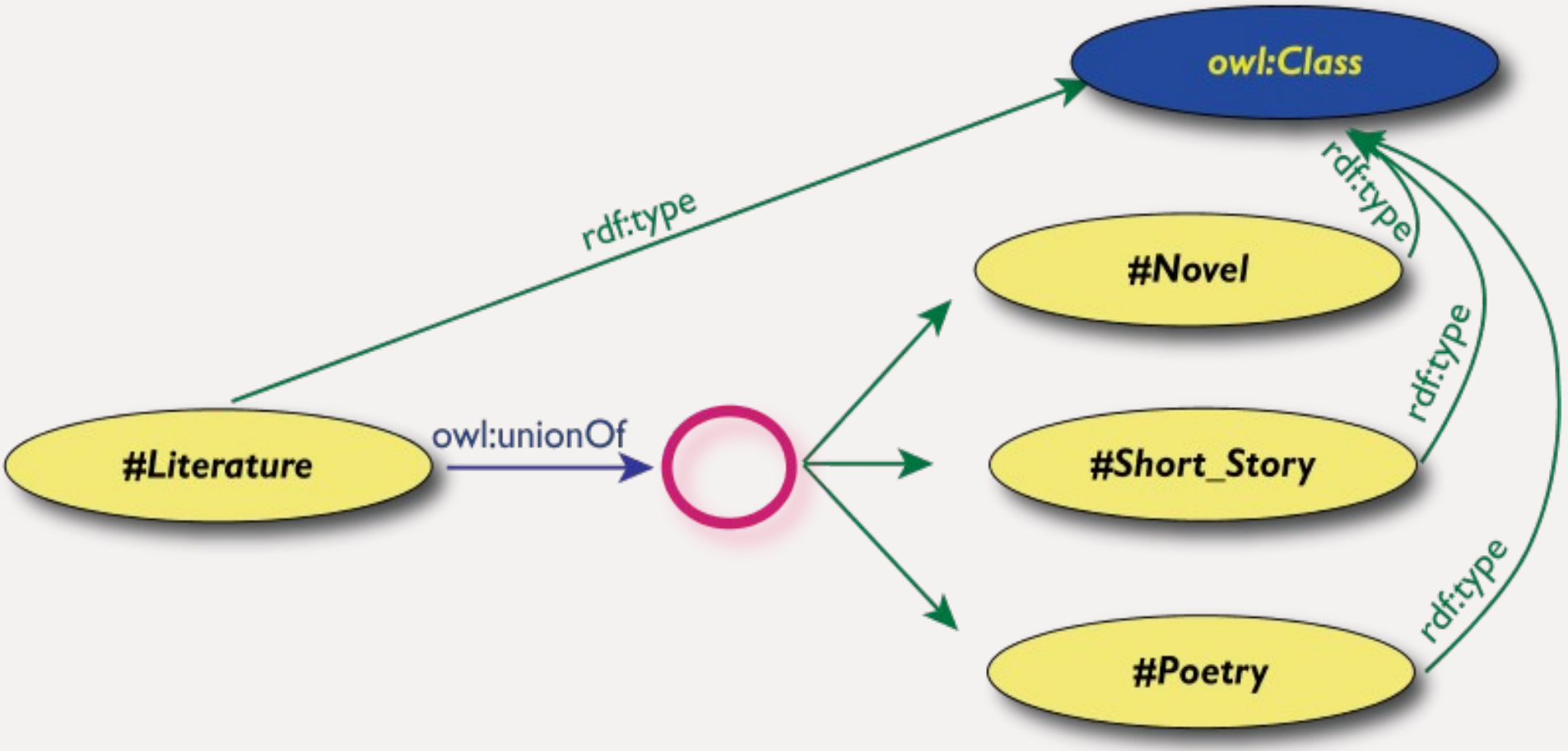
- The OWL solution, where possible content is explicitly listed:



(don’t worry about the syntax mapping…)

Union of classes

- Essentially, like a set-theoretical union:

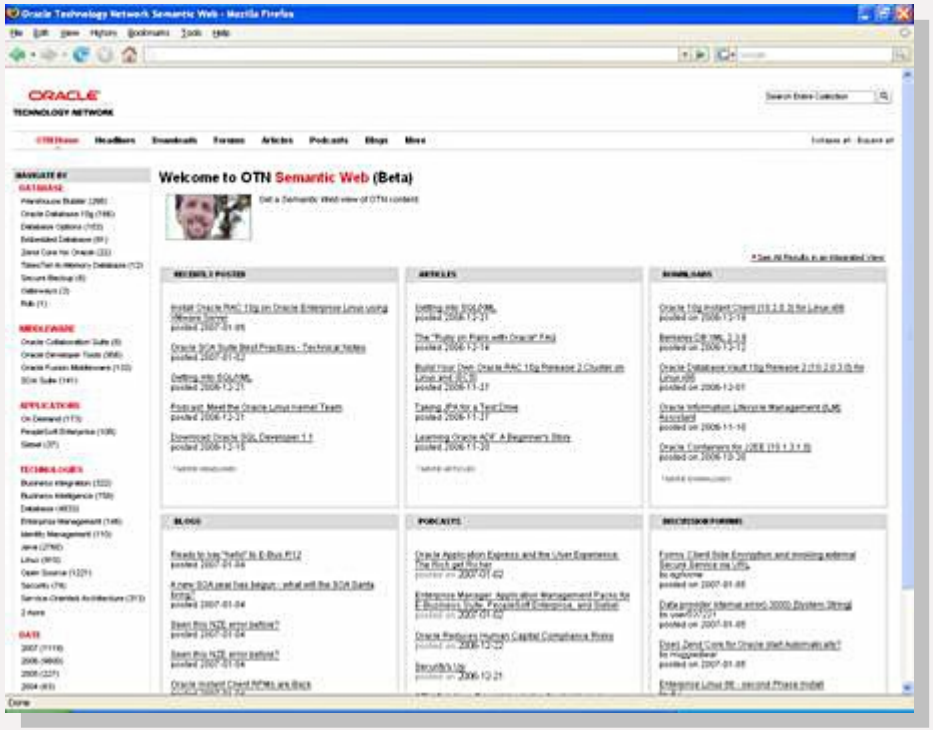


- Other possibilities: intersection, complement

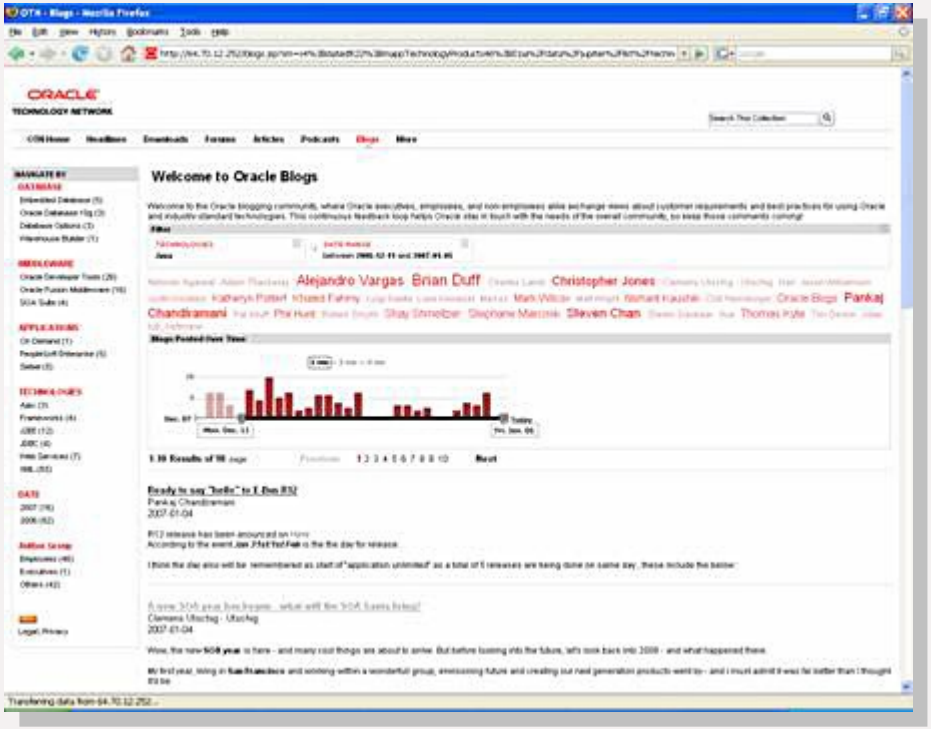
What we have so far…

- The OWL features listed so far are already fairly powerful
- E.g., various databases can be linked via **`owl:sameAs`**, functional or inverse functional properties, etc.
- It is still possible to find inferred relationship using a traditional rule engine
  - (more or less… there are some restrictions on details)

Oracle's Technology Network portal



Aggregates many source of content



Re-group, categorize, etc content (using a taxonomy)

Courtesy of Mike DiLascio, Siderean Software, and Justin Kestelyn, Oracle Corporation (SWEO Case Study)

However… that may not be enough

- Very large vocabularies might require even more complex features
  - typical example: definition of all concepts in a health care environment
- One major issue is the way classes (i.e., “concepts”) are defined
- OWL includes those extra features but… the inference engines become (much) more complex

Property value restrictions

- Classes are created by *restricting* the property values on a superclass
- For example: how would I characterize a “listed price”?
  - it is a price (which may be a general term), but one that is given in one of the “allowed” currencies (say, €, £, or $)
  - more formally:
    - the value of “`p:currency`”, when applied to a resource on listed price, *must* take one of those values…
    - …thereby defining the class of “listed price”

Restrictions expressed in OWL/RDF

```
:Listed_Price rdf:type owl:Class;
    rdfs:subClassOf [
      rdf:type              owl:Restriction;
      owl:onProperty        <http://…#currency>;
      owl:allValuesFrom     :Currency.
   ].
```

- "**allValuesFrom**" could be replaced by "**someValuesFrom**" to express another type of restriction
  - e.g., I could have said: there should be a price given in *at least one* of those currencies
- or "**hasValue**", when restricted to *one specific value*

Similar concept: cardinality restriction

- In a property restriction, the issue was to restrict the possible *values* of a property
- In a cardinality restriction, the *number* of relations with that property is restricted
- Eg: “a book being on offer” could be characterized as having at *least one price property* (i.e., the price of the book has been established)

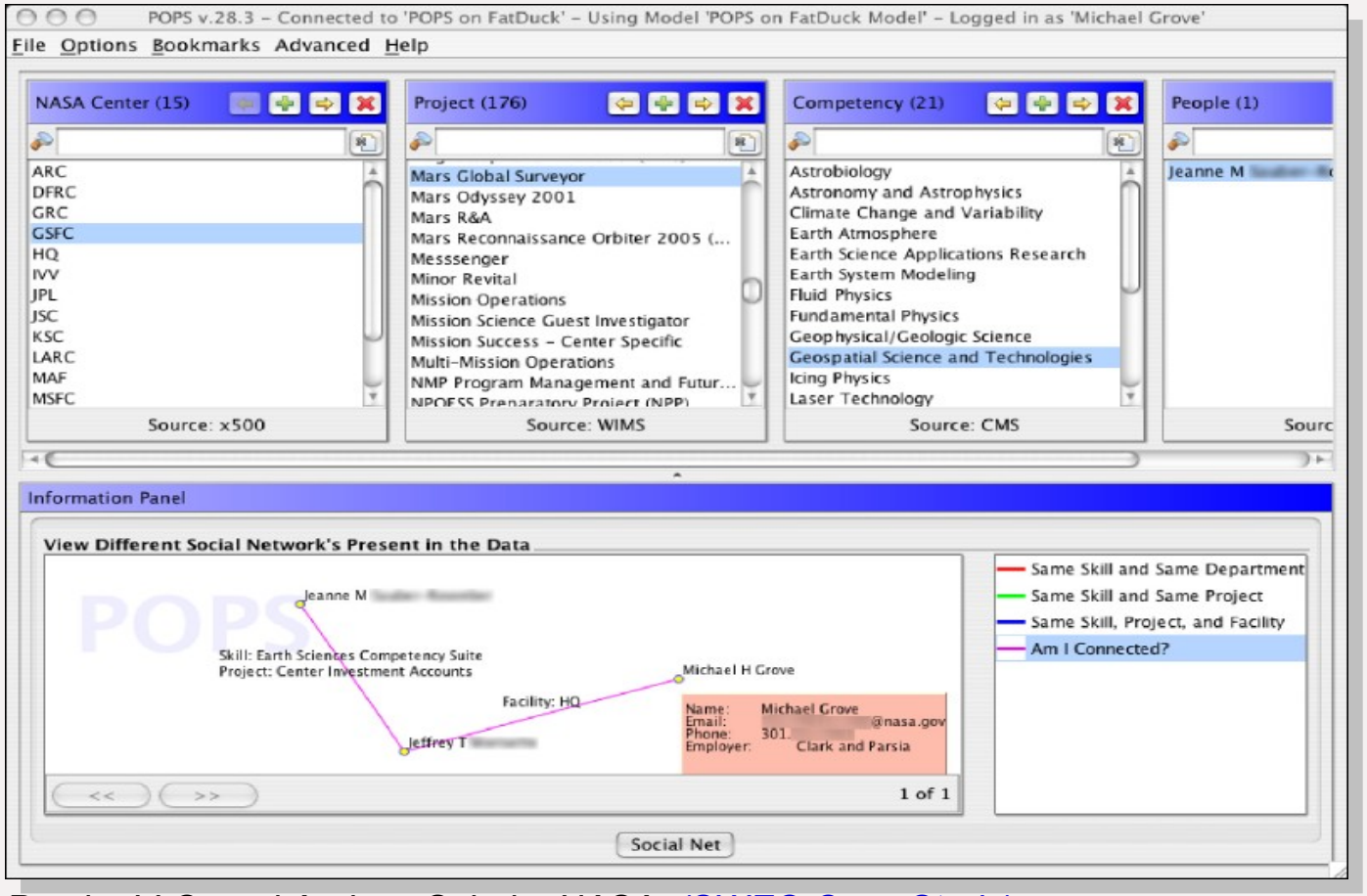
Cardinality restriction

```
:Book_on_sale rdf:type owl:Class;
    rdfs:subClassOf [
      rdf:type                owl:Restriction;
      owl:onProperty          <http://…#price>;
      owl:minCardinality      "1"^^xsd:integer.
   ].
```

- could also be "**owl:cardinality**" or "**owl:maxCardinality**"

Find the right experts at NASA

- Expertise locater for nearly 70,000 NASA civil servants, using RDF integration techniques over 6 or 7 geographically distributed databases, data sources, and web services…



Michael Grove, Clark & Parsia, LLC, and Andrew Schain, NASA, (SWEO Case Study)

But: OWL is hard!

- The combination of class constructions with various restrictions is extremely powerful
- What we have so far is following the same logic as before
  - extend the basic RDF and RDFS possibilities with new features
  - expect to infer new relationships based on those
- However… a full inference procedure is hard 
  - not implementable with simple rule engines, for example
  - in some cases, it may even be impossible

OWL profiles

- The term OWL “profiles” comes to the fore:
  - restricting *which* terms can be used and *under what circumstances (restrictions)*
  - if one abides to those restrictions, then simpler inference engines can be used

OWL profiles (cont.)

- In the *current* OWL standard, three such “profiles” are defined:
  - OWL Full: no restrictions whatsoever
  - OWL DL (and its “sub profile” OWL Lite): major restrictions to ensure implementability
- The OWL 2 work will add new profiles
  - profiles that are simple enough to be implementable with simple rule engines (like the first few examples we had)
  - profiles that are optimized to a small number of class and property definition but a large amount of data
  - etc.

OWL Full

- No constraints on the various constructs
  - this means that:
    - Class can also be an individual, a URI can denote a property as well as a Class
      - e.g., it is possible to talk about class of classes, etc.
    - one can make statements on RDFS constructs (e.g., declare **`rdf:type`** to be functional…)
    - etc.
- But: *an OWL Full ontology may be undecidable!*

OWL Description Logic (DL)

- A number of restrictions are defined
  - classes, individuals, properties strictly separated: a class *cannot* be an individual of another class
  - strict separation of the user's and the reserved (RDFS, OWL) terms
    - no statements on RDFS and OWL resources, for example
  - the values of user's object properties must be individuals
    - i.e., they are used to create relationships *between individuals*
  - no characterization of *datatype* properties
  - …
- But: *well known inference algorithms exist!*

Note on OWL profiles

- OWL profiles are defined to reflect compromises:
  - expressibility vs. implementability
- Some application just need to express and interchange terms (with possible scruffiness): OWL Full is fine
  - they may build application-specific reasoning instead of using a general one
- Some applications need rigour, but only a simple set of statements: a rule engine based profile might be o.k.
- Some applications need rigour and complex term classification: OWL DL might be the good choice

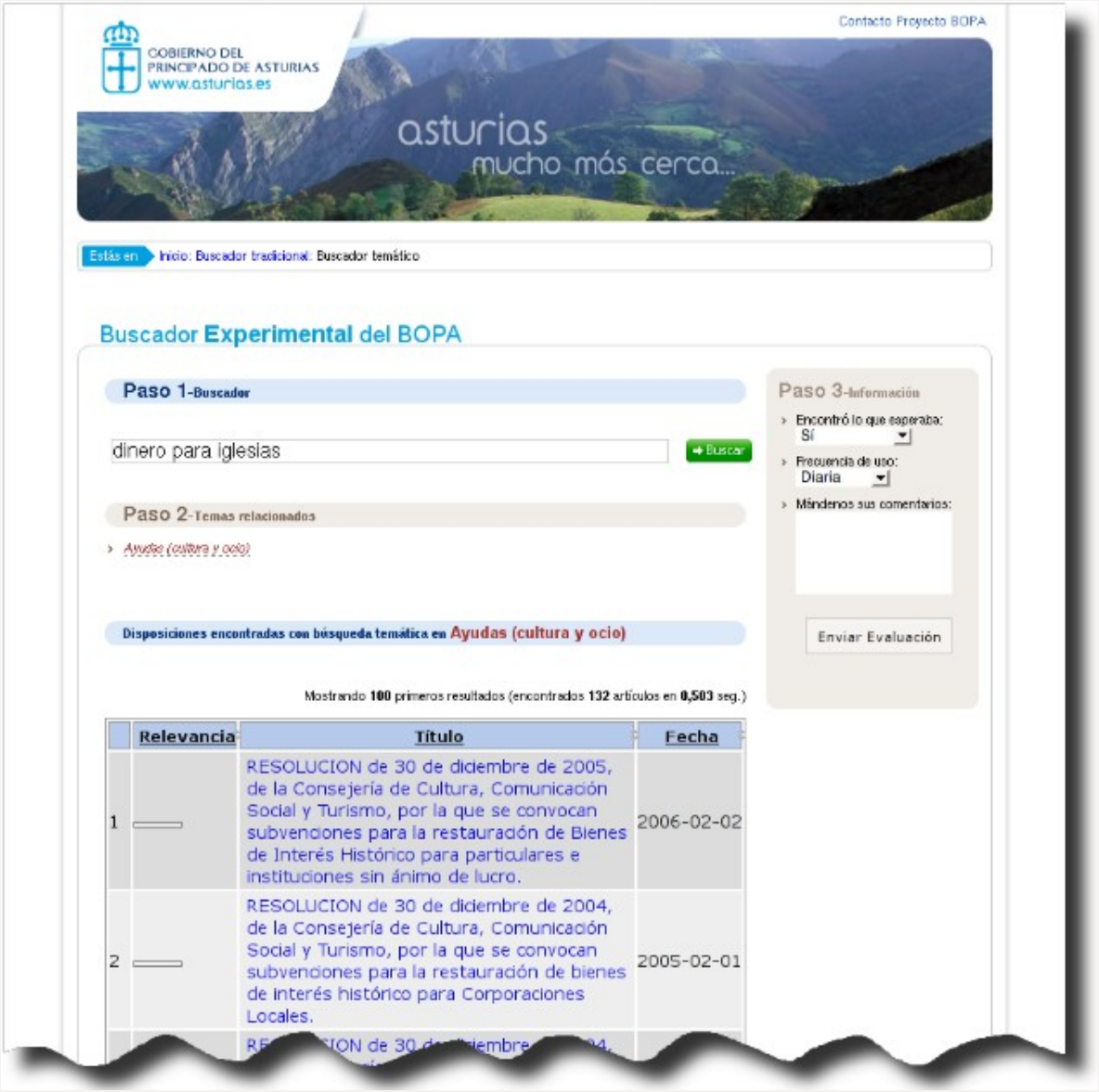
Ontology development

- The hard work is to *create* the ontologies
  - requires a good knowledge of the area to be described
  - some communities have good expertise already (e.g., librarians)
  - *OWL is just a tool to formalize ontologies*
- Large scale ontologies are often developed in a community process
- Ontologies should be *shared* and *reused*
  - can be via the simple namespace mechanisms…
  - …or via explicit inclusions
- Applications can also be developed with very small ontologies, though

Ontologies examples

- International Country List
  - example for an OWL Lite ontology
- Large ontologies are being developed
  - eClassOwl: eBusiness ontology for products and services, 75,000 classes and 5,500 properties
  - National Cancer Institute's ontology: about 58,000 classes
  - Open Biomedical Ontologies Foundry: a collection of ontologies, including the Gene Ontology to describe gene and gene product attributes in any organism or protein sequence and annotation terminology and data (UniProt)
  - BioPAX: for biological pathway data

Portal to Principality of Asturias' documents

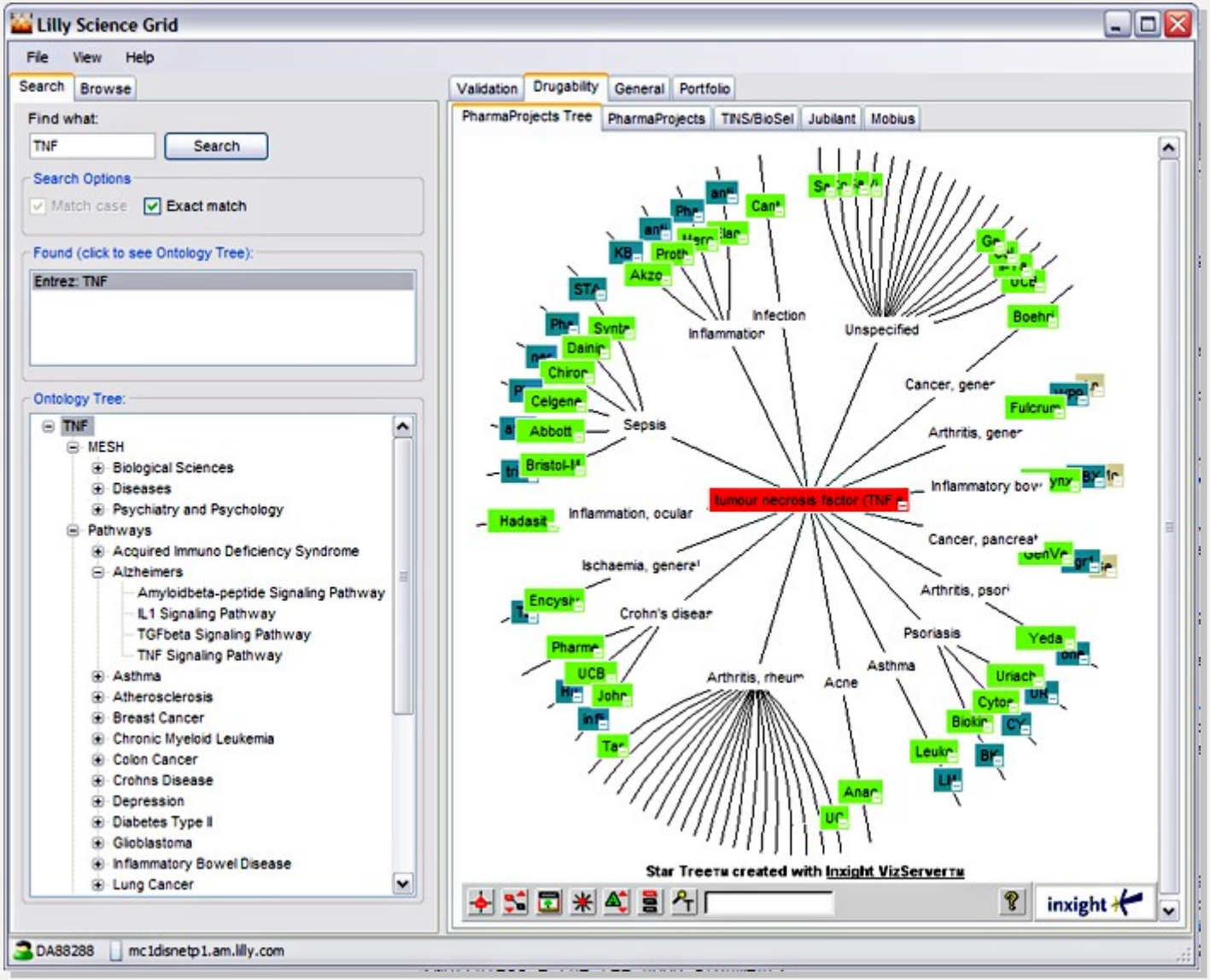


- Search through governmental documents
- A "bridge" is created between the users and the juridical jargon using SW ontologies and tools

Courtesy of Diego Berrueta and Luis Polo, CTIC, U. of Oviedo, and the Principality of Asturias, (SWEO Case Study)

Eli Lilly's Target Assessment Tool

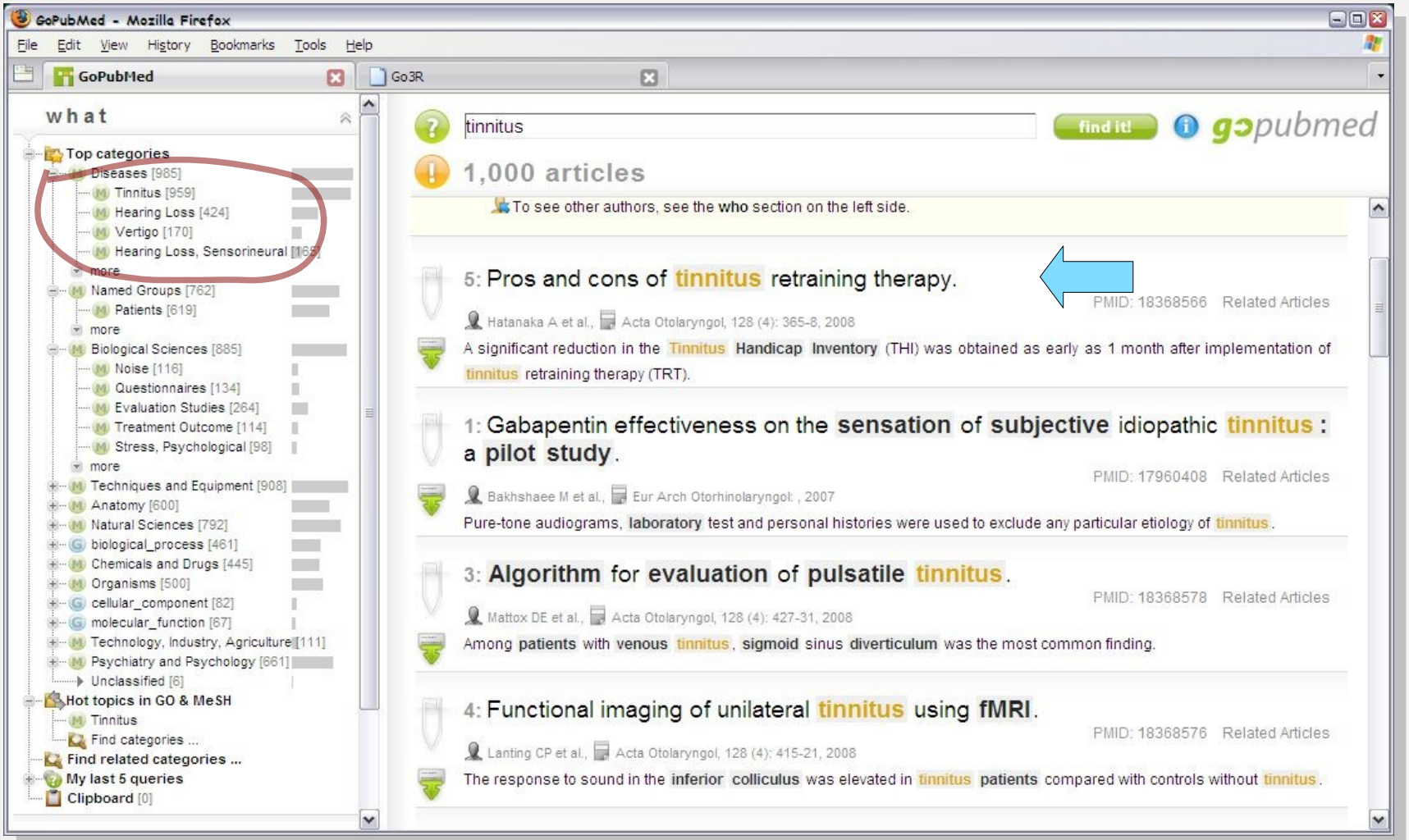
- Prioritization of drug target, integrating data from different sources and formats
- Integration, search via ontologies (proprietary and public)



Courtesy of Susie Stephens, Eli Lilly (SWEO Case Study)

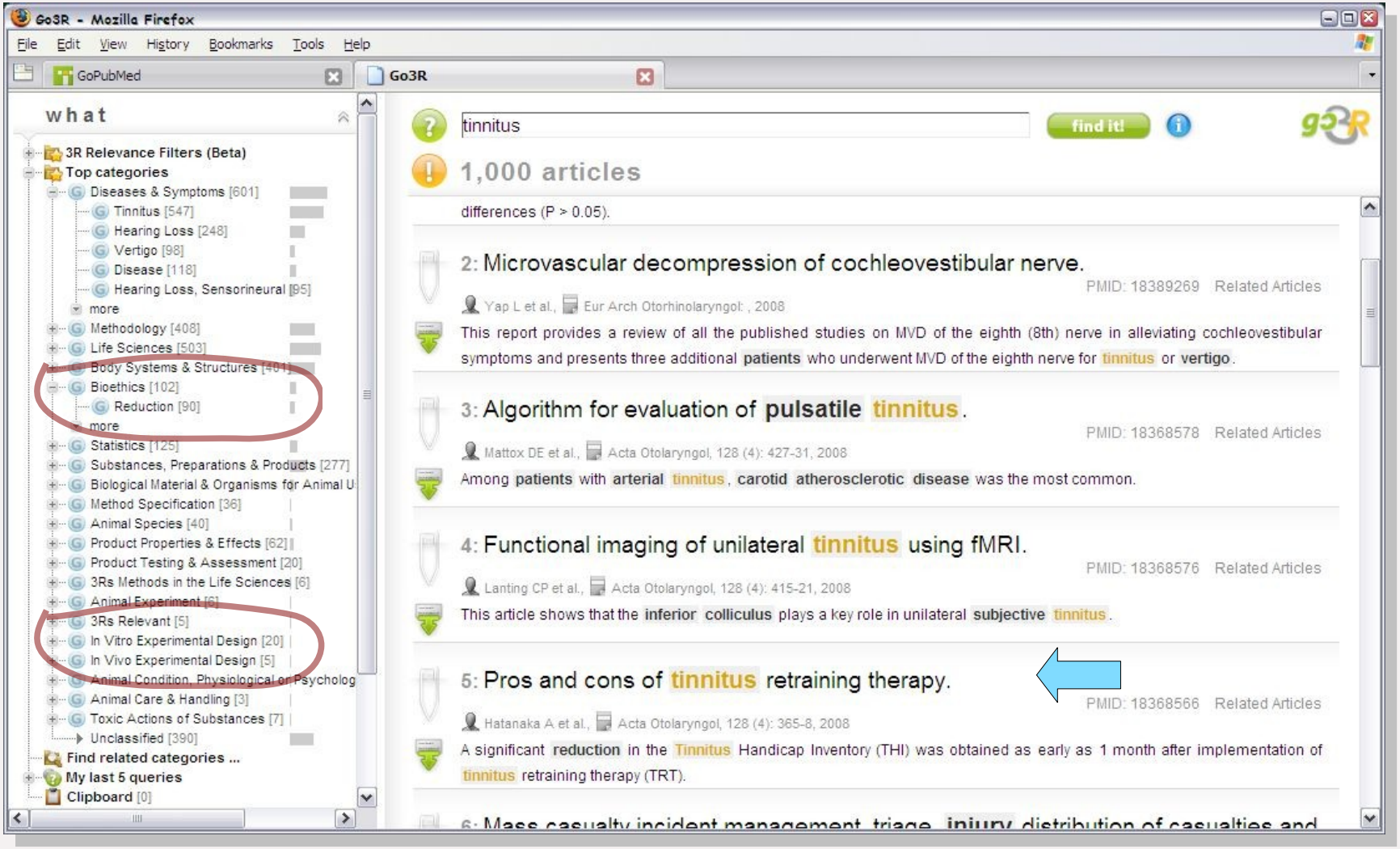
Improved Search via Ontology (GoPubMed)

- Search results are re-ranked using ontologies
- Related terms are highlighted

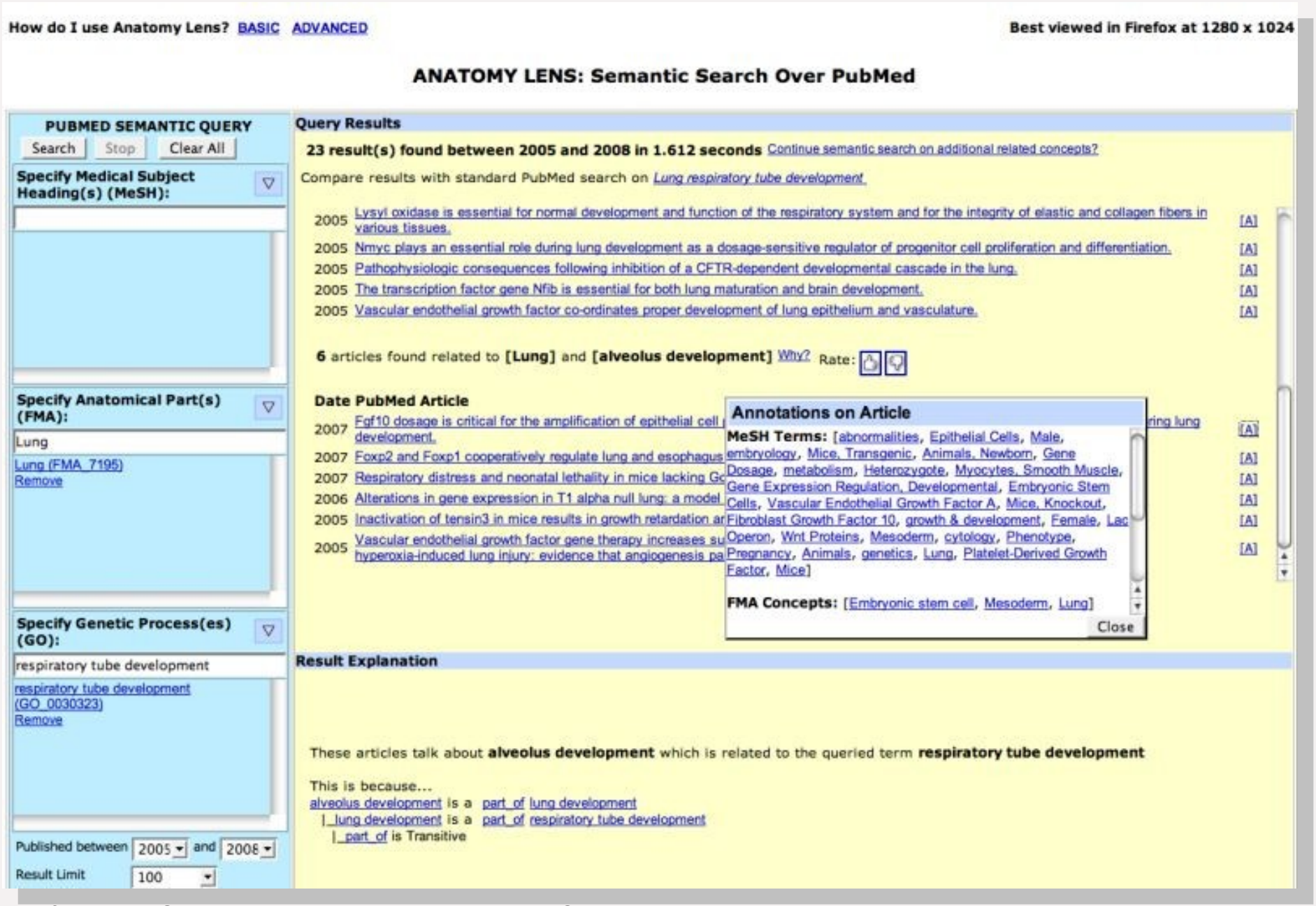


Improved Search via Ontology (Go3R)

- Same dataset, different ontology
  - (ontology is on non-animal experimentation)



Same problem, different solution…



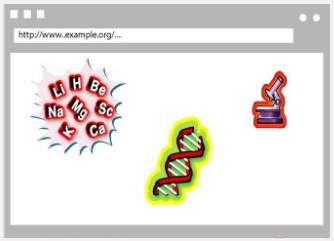
Courtesy of Kavitha Srinivas, IBM J Watson Research Center

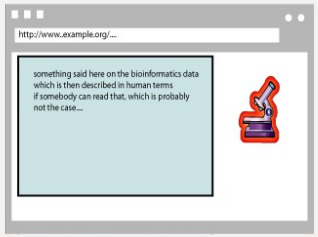
What have we achieved?

We have not covered everything…

- Some other aspects of SW are being developed
  - some at W3C, others are still research
- For example:
  - RIF: using general rule engines with SW data; also *interchange* rule descriptions (just like data are interchanged)
  - SKOS: general framework to express term structures like vocabularies, taxonomies, glossaries
    - eg, to interface bibliographic records

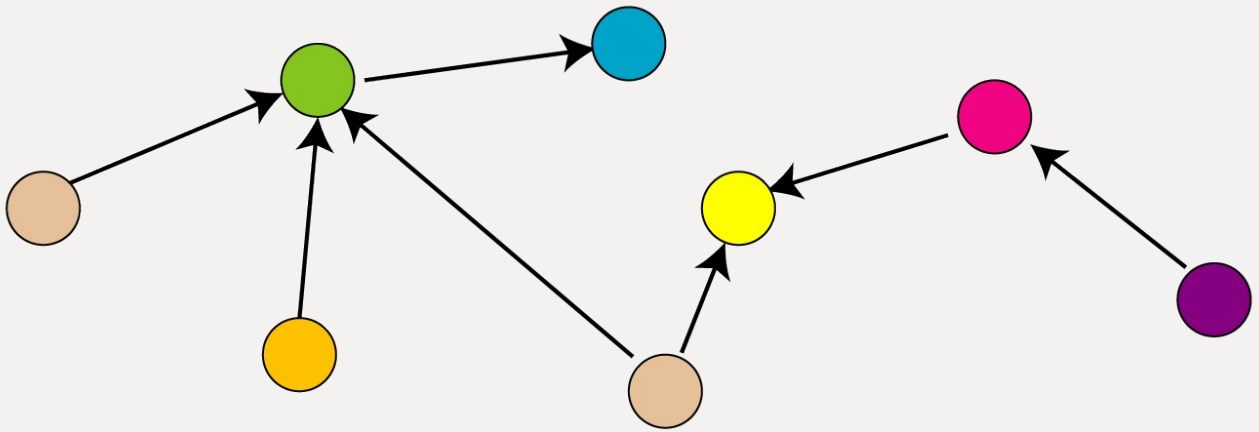
Remember the integration example?





Applications

Query, Manipulate, etc.

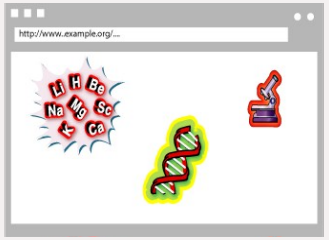


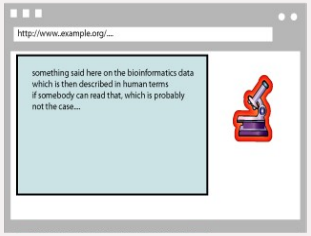
Data represented in abstract format

Map, Expose, etc.

Data in various formats

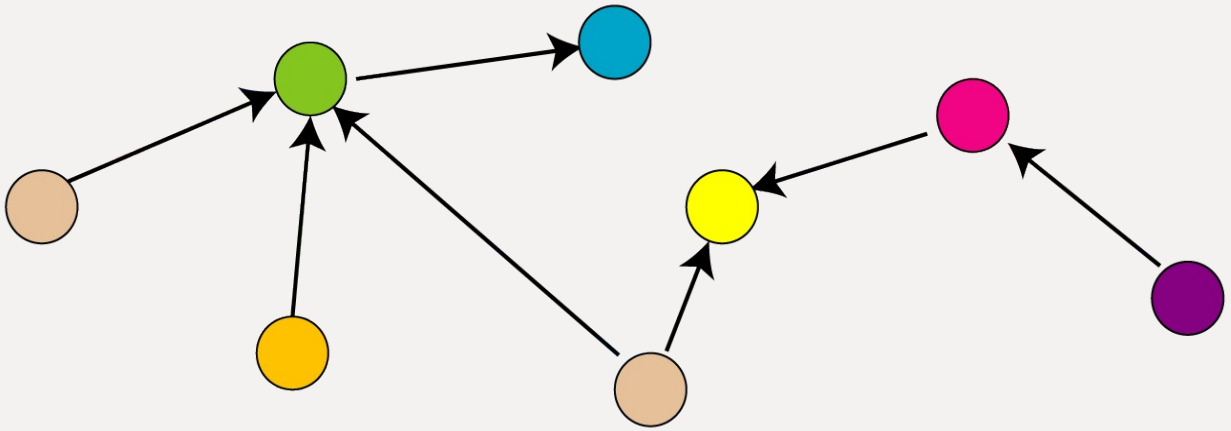
Same with what we learned





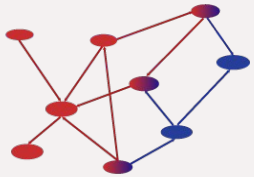
Applications

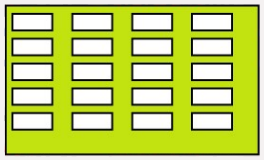
SPARQL,
OWL inferences,
etc.

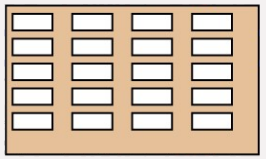


Data represented in RDF, possibly with extra knowledge (RDFS, OWL, SKOS, Rules, …)

SQL <=> RDF,
GRDDL, RDFa
etc.

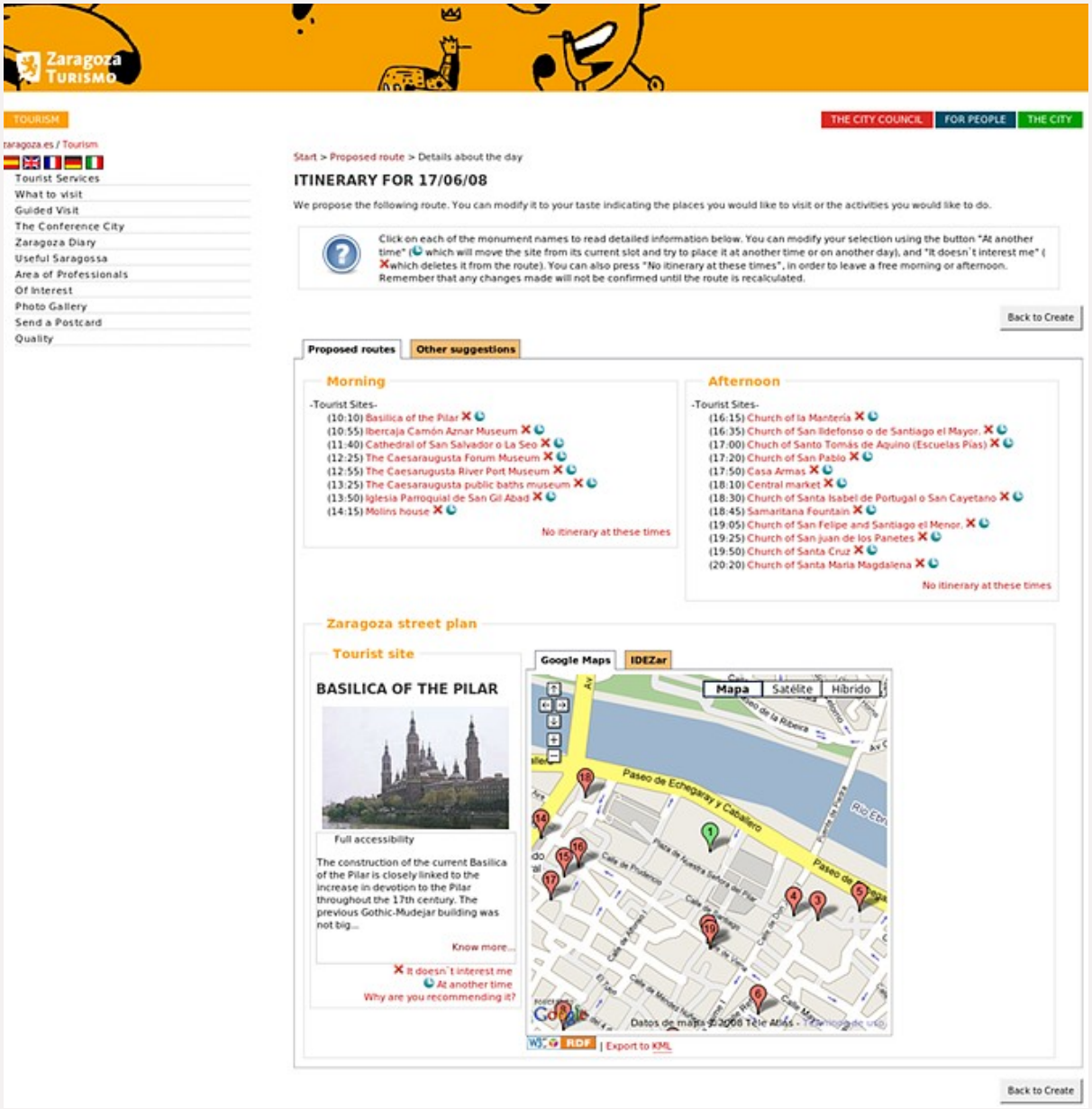






Data in various formats

eTourism: provide personalized itinerary

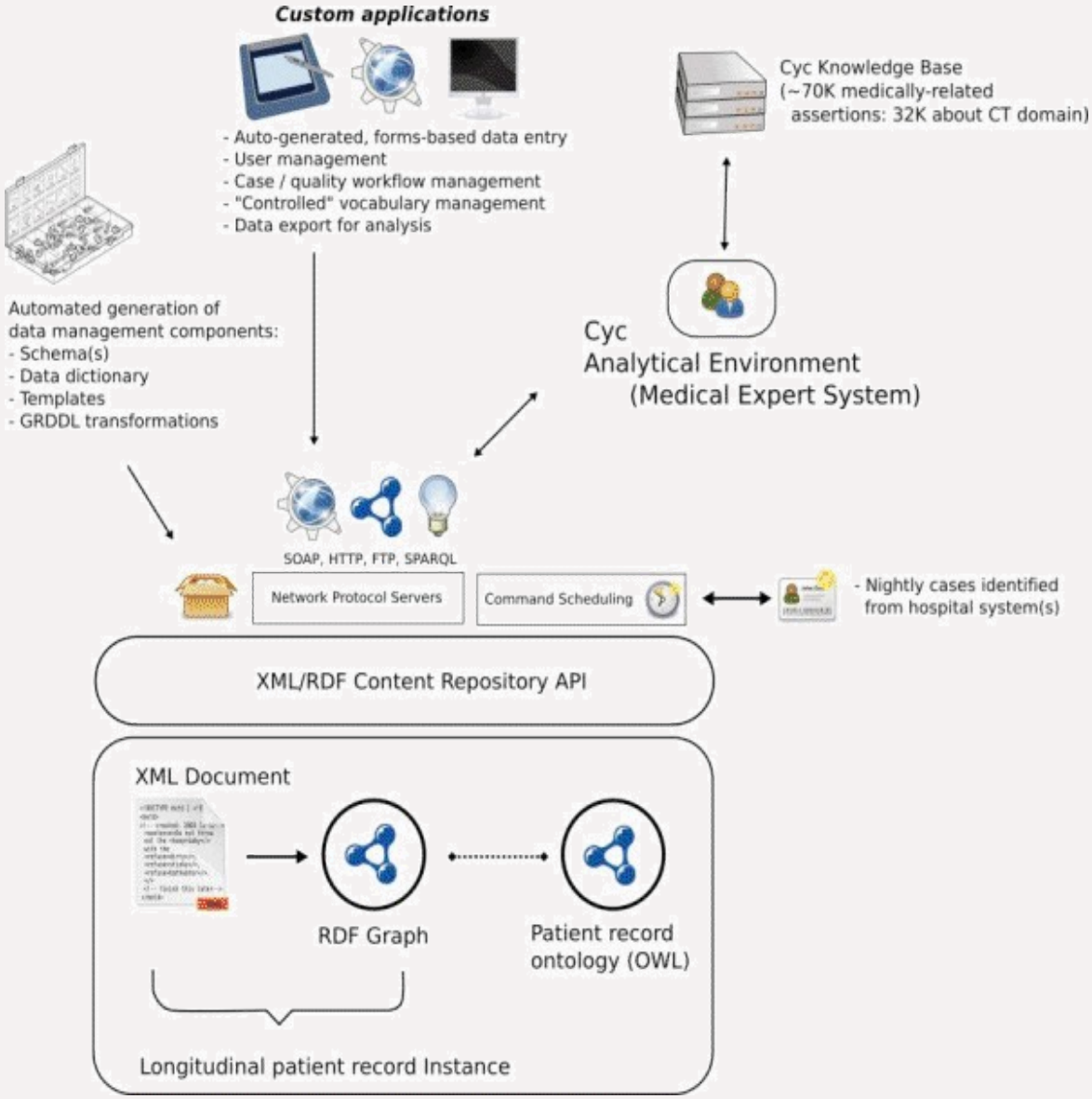


- Integration of relevant data in Zaragoza (using RDF and ontologies)
- Use rules on the RDF data to provide a proper itinerary

Courtesy of Jesús Fernández, Municipality of Zaragoza, and Antonio Campos, CTIC (SWEO Use Case)

Semantic DB at the Cleveland Clinic

- Problem: extreme compartmentalization of medical knowledge
- Unified repository collects and stores various data
- Usage of OWL and rules allow high level operations on the data



Courtesy of Chimezie Ogbuji, ClevelandClinic, (SWEO Case Study)

National Archives of Korea

- Ontology based metadata infrastructure for NAK (over 12 million metadata statements)
- Usage of rules to retrieve information

Courtesy of Tony Lee, Jin Woo Kim, and Bok Ju Lee, Saltlux, Kyu Hyup Kim, Yoon Jung Kang, NAK (SWEO Case Study)

Available documents, tools

Available specifications: Primers, Guides

- The “RDF Primer” and the “OWL Guide” give a formal introduction to RDF(S) and OWL
- GRDDL Primer and RDFa Primer have been published
- The W3C Semantic Web Activity Homepage has links to all the specifications

“Core” vocabularies

- There are also a number “core vocabularies” (not necessarily OWL based)
  - Dublin Core: about information resources, digital libraries, with extensions for rights, permissions, digital right management
  - FOAF: about people and their organizations
  - DOAP: on the descriptions of software projects
  - SIOC: Semantically-Interlinked Online Communities
  - vCard in RDF
  - …
- One should never forget: ontologies/vocabularies must be shared and reused!

Some books

- P. Mika: Social Networks and the Semantic Web, 2007
- L. Yu: Semantic Web and Semantic Web Services, 2007
- G. Antoniu and F. van Harmelen: Semantic Web Primer, 2nd edition in 2008
- D. Allemang and J. Hendler: Semantic Web for the Working Ontologist, 2008
- …

See the separate Wiki page collecting book references

Further information

- Dave Beckett's Resources at Bristol University
  - *huge* list of documents, publications, tools, …
- Planet RDF aggregates a number of SW blogs
- Semantic Web Interest Group
  - a forum developers with archived (and public) mailing list, and a constant IRC presence on freenode.net#swig
  - anybody can sign up on the list

Lots of Tools (*not* an exhaustive list!)

- Categories:
  - Triple Stores
  - Inference engines
  - Converters
  - Search engines
  - Middleware
  - CMS
  - Semantic Web browsers
  - Development environments
  - Semantic Wikis
  - …
- Some names:
  - Jena, AllegroGraph, Mulgara, Sesame, flickurl, …
  - TopBraid Suite, Virtuoso environment, Falcon, Drupal 7, Redland, Pellet, …
  - Disco, Oracle 11g, RacerPro, IODT, Ontobroker, OWLIM, Tallis Platform, …
  - RDF Gateway, RDFLib, Open Anzo, DartGrid, Zitgist, Ontotext, Protégé, …
  - Thetus publisher, SemanticWorks, SWI-Prolog, RDFStore…
  - …

Tools

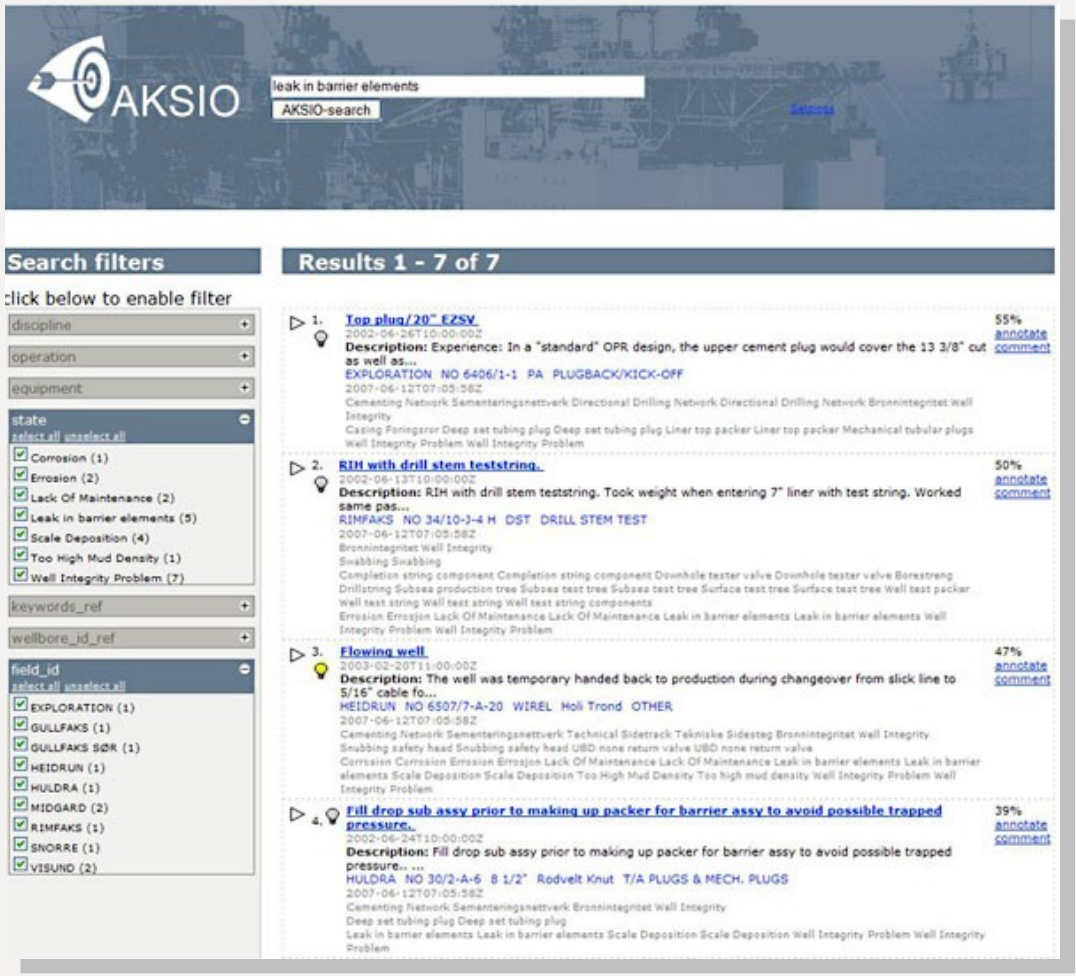
- Worth noting: major companies offer (or will offer) Semantic Web tools or systems using Semantic Web: Adobe, Oracle, IBM, HP, Software AG, webMethods, Northrop Gruman, Altova, Dow Jones, BBN, …
- See also the W3C Wiki page on tools

Application patterns

- It is fairly difficult to “categorize” applications (there are always overlaps)
- With this caveat, some of the application patterns:
  - data integration (ie, integrating data from major databases)
  - intelligent (specialized) portals (with improved local search based on vocabularies and ontologies)
  - content and knowledge organization
  - knowledge representation, decision support
  - X2X integration (often combined with Web Services)
  - data registries, repositories
  - collaboration tools (eg, social network applications)

Help for deep sea drilling operations

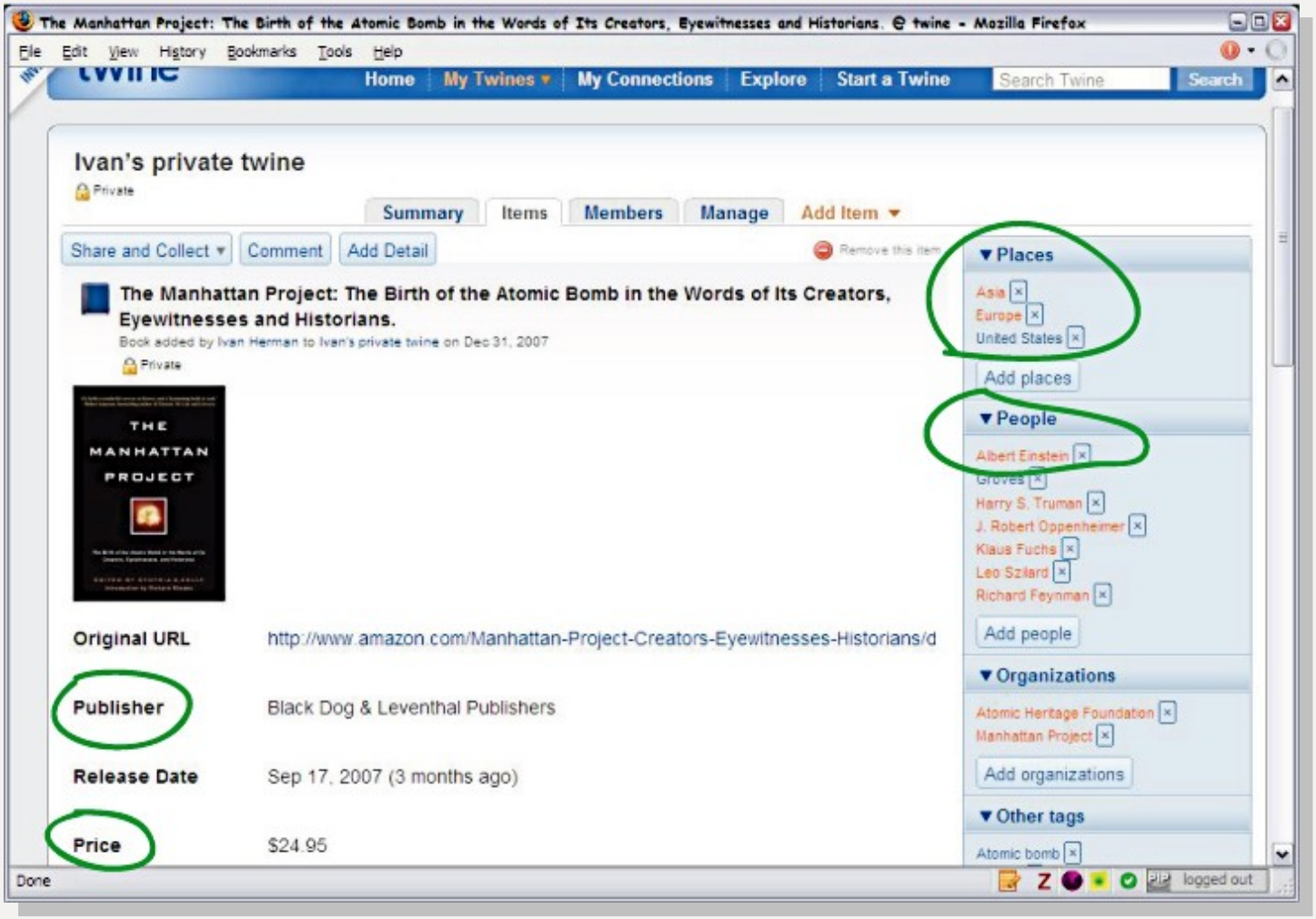
- Integration of experience and data in the planning and operation of deep sea drilling processes
- Discover relevant experiences that could affect current or planned drilling operations
  - uses an ontology backed search engine



Courtesy of David Norheim and Roar Fjellheim, Computas AS (SWEO Use Case)

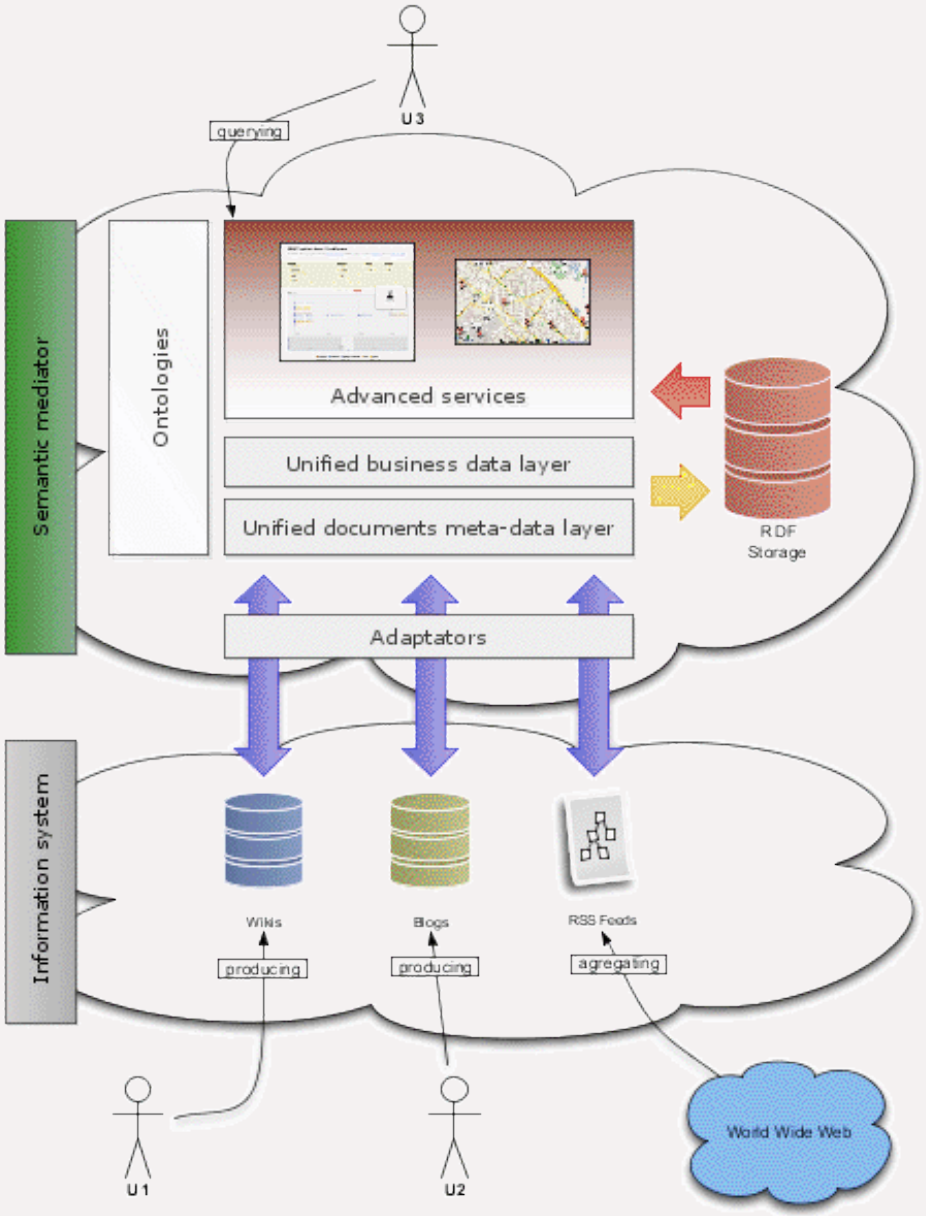
Radar Network's Twine

- "Social bookmarking on steroids"
- Relationships are based on ontologies
  - evolving over time, possibly enriched by users
- Internals in RDF



Integration of “social” software data

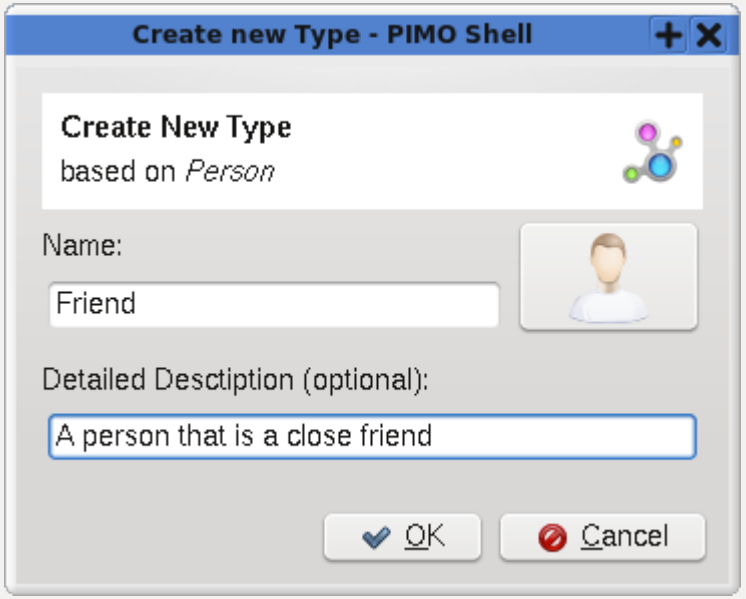
- Internal usage of wikis, blogs, RSS, etc, at EDF
  - uses:
    - public ontologies (SIOC, FOAF, DC, Geonames)
    - public datasets for tagging
    - SPARQL as integration tool for queries
- Details are hidden from end users (via plugins, extra layers, etc)

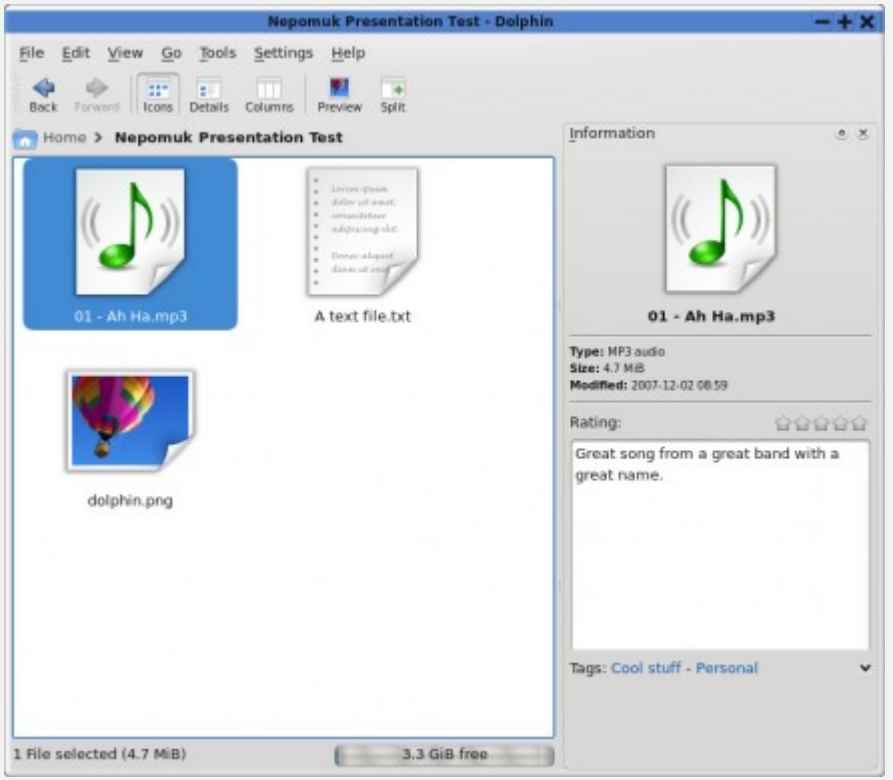


Courtesy of Alexandre Passant, EDF R&D and LaLIC, Université Paris-Sorbonne, (SWEO Case Study)

Semantic tagging and search in KDE

- Metadata backend fully based on RDF
- Each file can be tagged, rated, commented, and automatically indexed
- Queries are made via the combination of all those





Courtesy of Leo Sauermann, DFKI, and Sebastian Trüg, Mandriva Linux (SWEO Case Study)

Conclusions

- The Semantic Web is there to integrate data on the Web
- The goal is the creation of a *Web of Data*

Thank you for your attention!

- These slides are publicly available on:

http://www.w3.org/2008/Talks/0924-Vienna-IH/