

Introduction to the Semantic Web (tutorial)

*Johnson & Johnson
Philadelphia, USA
October 30, 2009*

*Ivan Herman, W3C
ivan@w3.org*

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.
- Humans combine these information easily even if
 - different terminologies are used
 - the information is incomplete, or buried in images, videos, ...

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 (i.e., on the Web!)
 - (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
 - combination of online library data
 - etc.
- Most of these data are accessible from the Web (though not necessarily public yet)

This problem you know very well...

CoCoDat - Collation of Cortical Data - Mozilla Firefox

File Edit View History Bookmarks Tools Help

CoCoMac DATABASES ORT EXAMPLES

CoCoDat: Collation of Cortical [sic] microcircuitry] Data

CoCoDat is a microcircuitry database that published experimental reports. The data and cellular compartment), as well as the

- Morphology
- Firing properties
- Ionic currents
- Ionic conductances
- Synaptic currents
- Connectivity

The database is available for download under data tables but also a Search Board with p manual or automatic relaxation of the sea

- Brain region
- Layer
- Neuron type

<http://www.cocomac.org/cocodat/catalyzer/index.html>

Cell Centered Database - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Cell Centered Database™ National Center for Microscopy and Imaging Research **Gallery**

Data | Search | Gallery | Dictionary | Publications | MyCCDB | Data Download | Contact us | Help

2D image Reconstruction Segmentation Animation

NeuronDB = Thalamic relay neuron - Overview (A) () - Mozilla Firefox

File Edit View History Bookmarks Tools Help

NeuronDB

Back Thalamic relay neuron

Mode: **Overview** Data/Search plus Connectivity plus Classical References/Notes Models

Region: Distal equivalent dendrite Middle equivalent dendrite Proximal equivalent dendrite Soma Axon hillock Axon fiber Axon terminal All Compartments

Properties: Receptors Channels Transmitters **All Properties**

Interoperation: Gene and Chromosome Experimental Data (neurodatabase.org) Microscopy Data (CCDB)

Neuron type: principal Organism: Vertebrates

1. Equivalent dendrite Show other
2. Distal equivalent dendrite Show other
3. Middle equivalent dendrite Show other
4. Proximal equivalent dendrite Show other
5. Soma Show other

Done

Example: social networks

- Social sites are everywhere these days (LinkedIn, Facebook, Dopplr, Digg, Plexo, Zyb, ...)
- Data is not interchangeable: 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...

Example: digital libraries

- Sort of catalogues on the Web
 - librarians have known how to do that for centuries
 - goal is to have this on the Web, World-wide
 - extend it to multimedia data, too
- But it is more: software agents should also be librarians!
 - e.g., help you in finding the right publications

What is needed?

- (Some) data should be available for machines for further processing
- Data should be possibly combined, merged on a Web scale
- Machines may also need to reason about that data
- Create a Web of Data (beyond the Web of Documents)

Find the right experts at NASA

- Expertise locator for nearly 70,000 NASA civil servants, integrating 6 or 7 geographically distributed databases, data sources, and web services...

The screenshot displays the POPS v.28.3 software interface, which is a knowledge management system for NASA. The window has a menu bar with File, Options, Bookmarks, Advanced, and Help. Below the menu are four main panels:

- NASA Center (15):** Lists facilities including ARC, DFRC, GRC, GSFC (selected), HQ, IIV, JPL, JSC, KSC, LARC, MAF, and MSFC. Source: x500.
- Project (176):** Lists projects such as Mars Global Surveyor, Mars Odyssey 2001, Mars R&A, Mars Reconnaissance Orbiter 2005, Messenger, Minor Revital, Mission Operations, Mission Science Guest Investigator, Mission Success – Center Specific, Multi-Mission Operations, NMP Program Management and Future..., and NPOESS Preparatory Project (NPP). Source: WIMS.
- Competency (21):** Lists competencies including Astrobiology, Astronomy and Astrophysics, Climate Change and Variability, Earth Atmosphere, Earth Science Applications Research, Earth System Modeling, Fluid Physics, Fundamental Physics, Geophysical/Geologic Science, Geospatial Science and Technologies (selected), Icing Physics, and Laser Technology. Source: CMS.
- People (1):** Shows a single entry for Jeanne M. Source: None.

At the bottom is an Information Panel titled "View Different Social Network's Present in the Data". It shows a network graph with nodes for Jeanne M., Michael H. Grove, and Jeffrey T. The graph indicates connections based on skill, project, facility, and department. A legend on the right defines the colors: red for Same Skill and Same Department, green for Same Skill and Same Project, blue for Same Skill, Project, and Facility, and magenta for Am I Connected? (which connects Jeanne M. to Michael H. Grove).

So what is the Semantic Web?

It is, essentially, the Web of Data.

“Semantic Web Technologies” is a collection of standard technologies to realize a Web of Data

- It is that simple...
- Of course, the devil is in the details
 - a common model has to be provided for machines to describe, query, etc, the data and their connections
 - the “classification” of the terms can become very complex for specific knowledge areas: this is where ontologies, thesauri, etc, enter the game...



In what follows...

- We will use a simplistic example to introduce the main technical concepts
- The details will be for later during the course

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 that could not have been done on the individual data sets

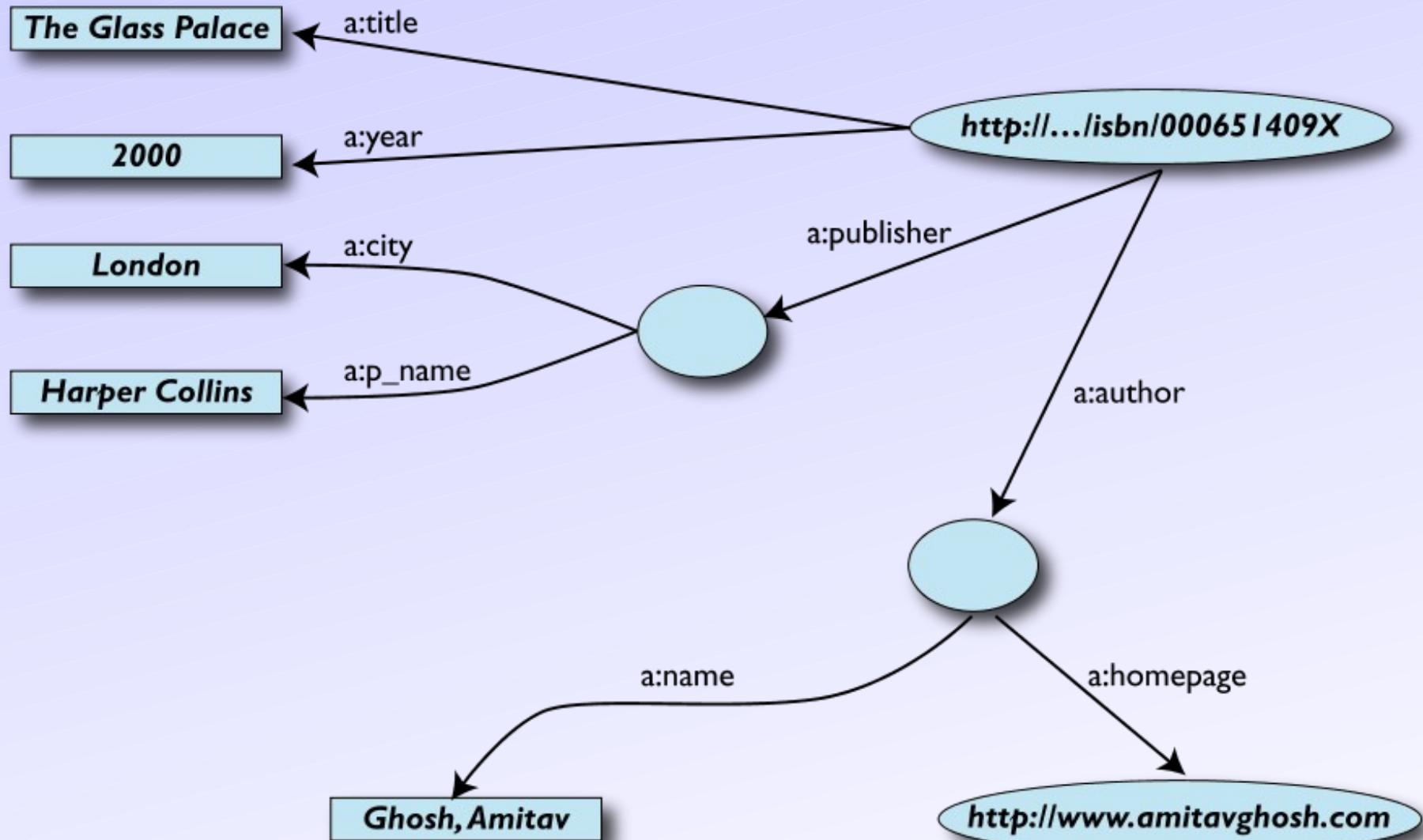
A *simplified* bookstore data (dataset “A”)

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	Harper Collins	London

1st: export your data as a set of relations



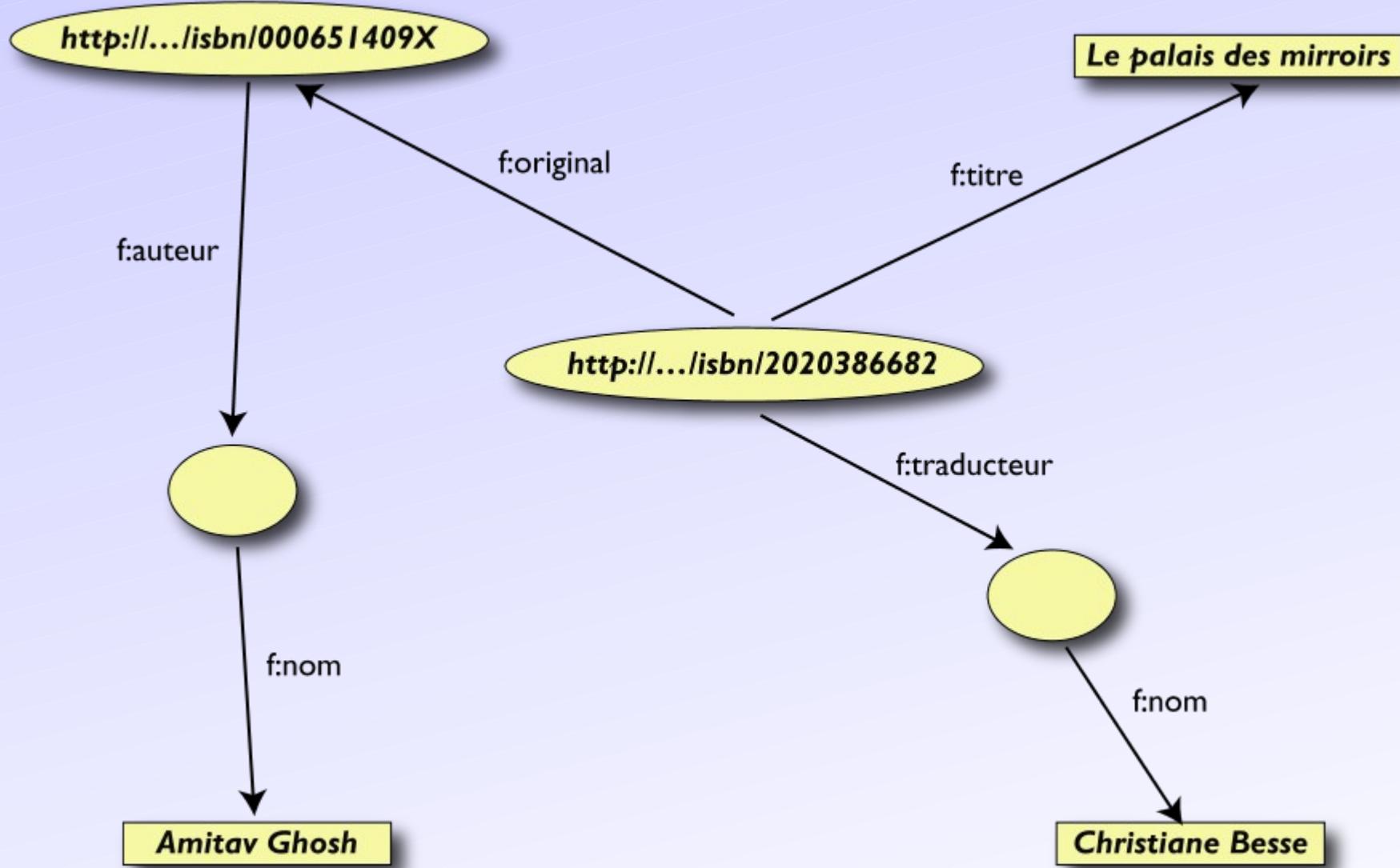
Some notes on the exporting the data

- Relations form a graph
 - the nodes refer to the “real” data or contain some literal
 - how the graph is represented in machine is immaterial for now
- 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

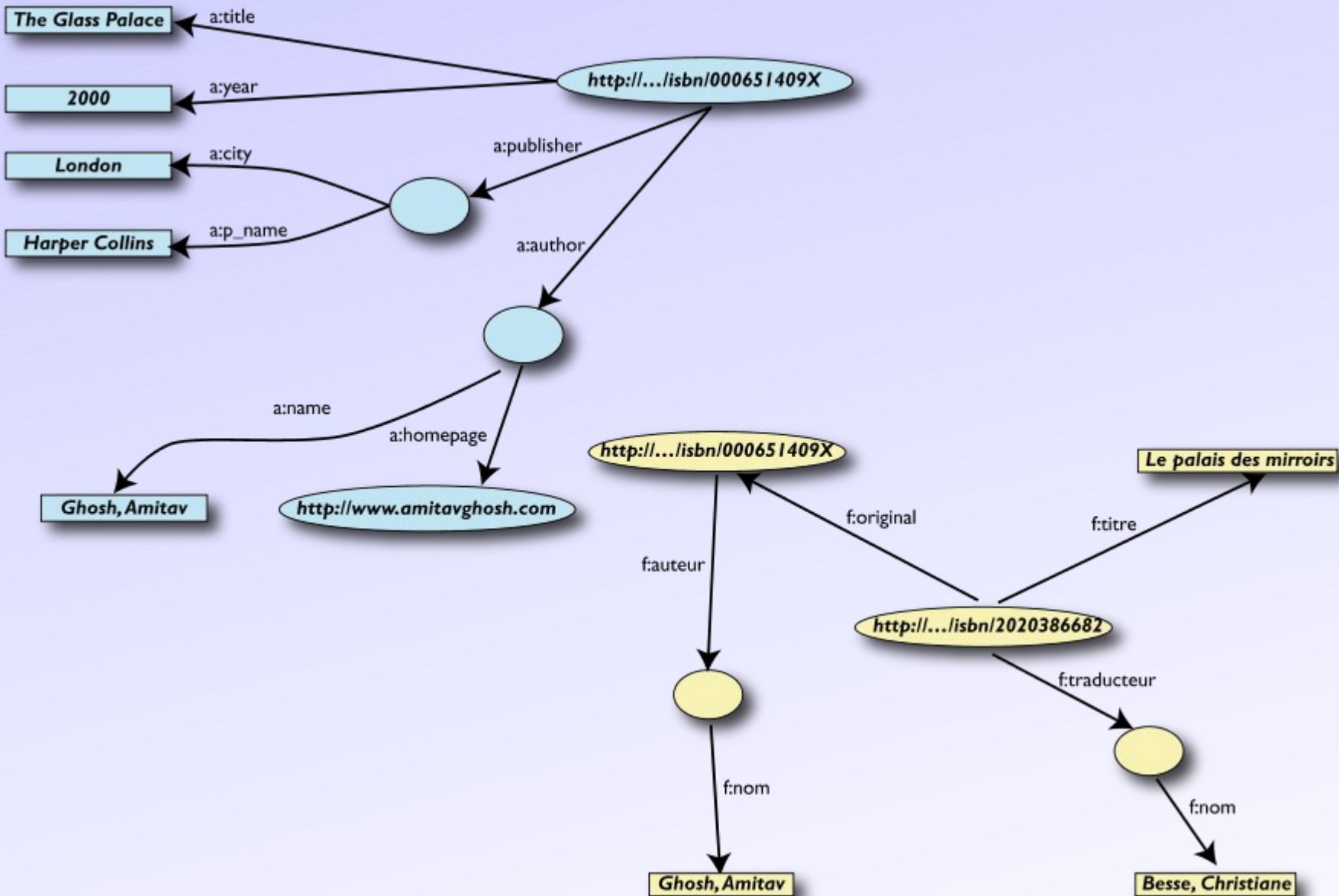
Another bookstore data (dataset “F”)

	A	B	D	E
1	ID	Titre	Traducteur	Original
2	ISBN0 2020386682	Le Palais des miroirs	A13	ISBN-0-00-651409-X
3				
6	ID	Auteur		
7	ISBN-0-00-651409-X	A12		
11		Nom		
12		Ghosh, Amitav		
13		Besse, Christianne		

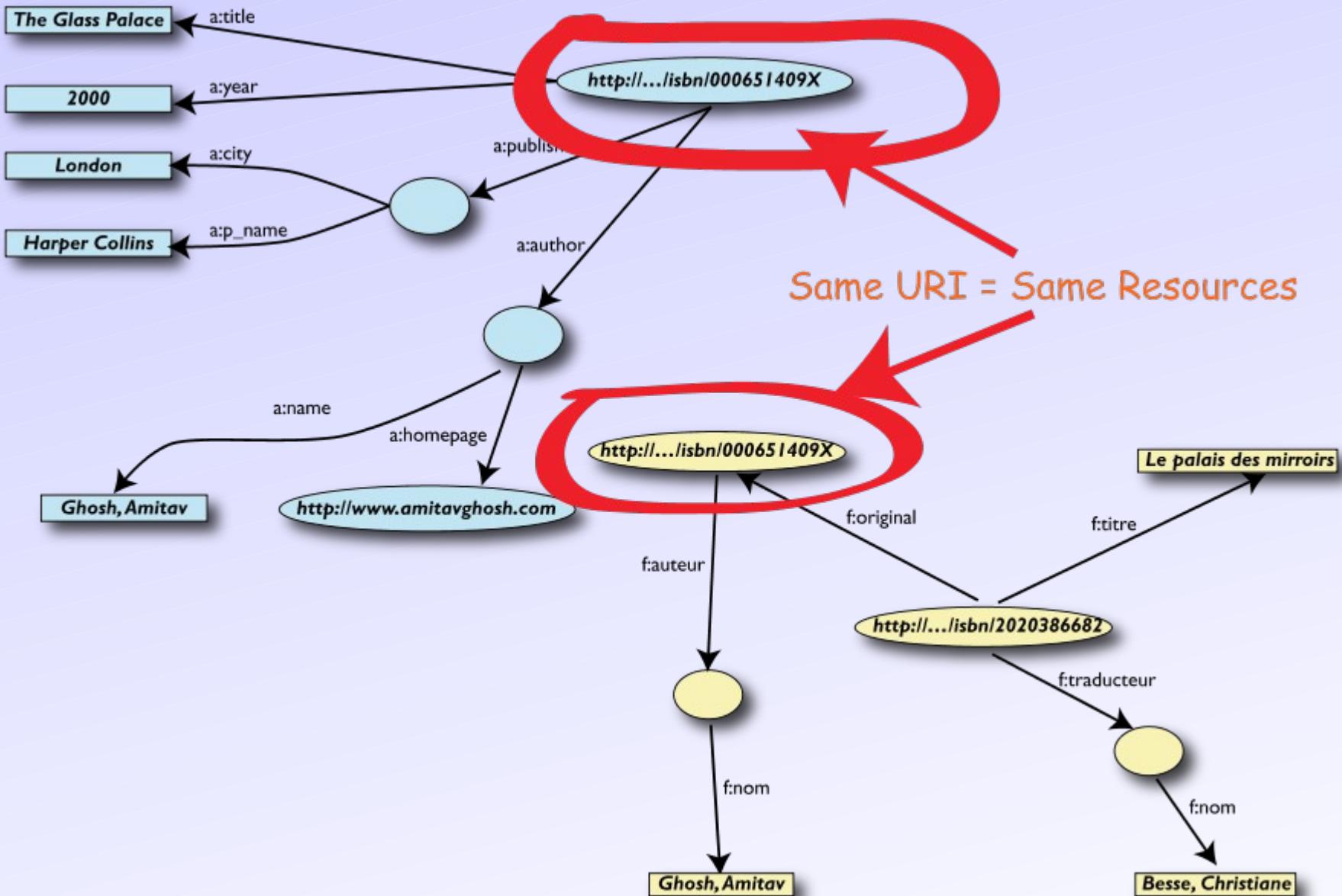
2nd: export your second set of data



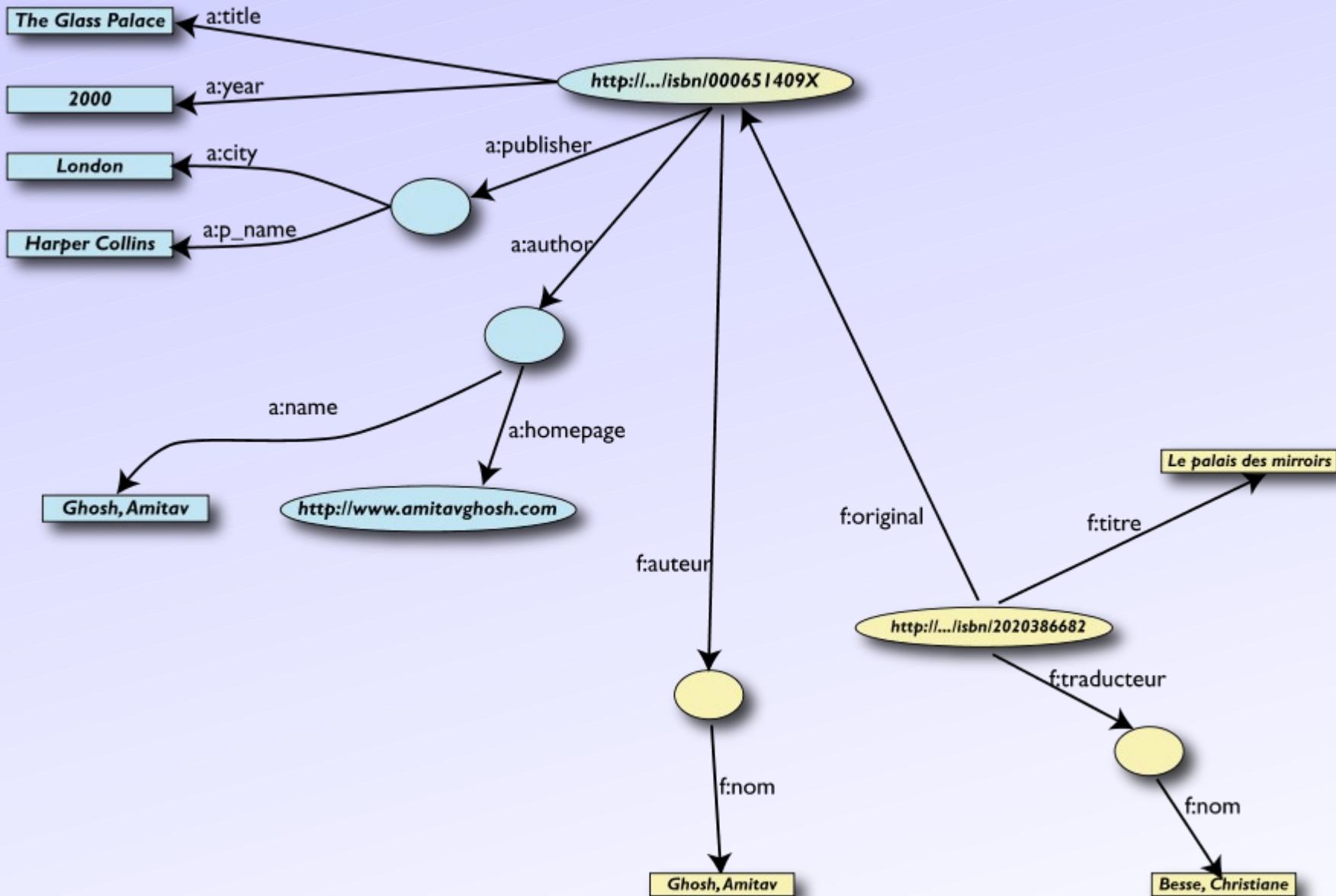
3rd: start merging your data



3rd: start merging your data (cont.)

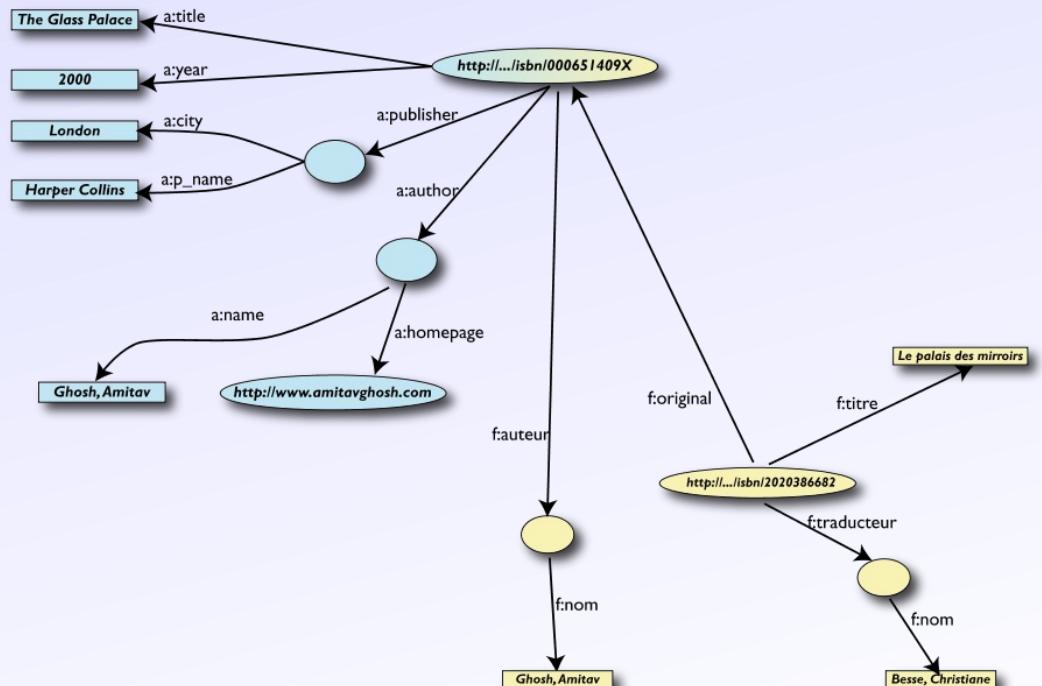


3rd: merge identical resources



Start making queries...

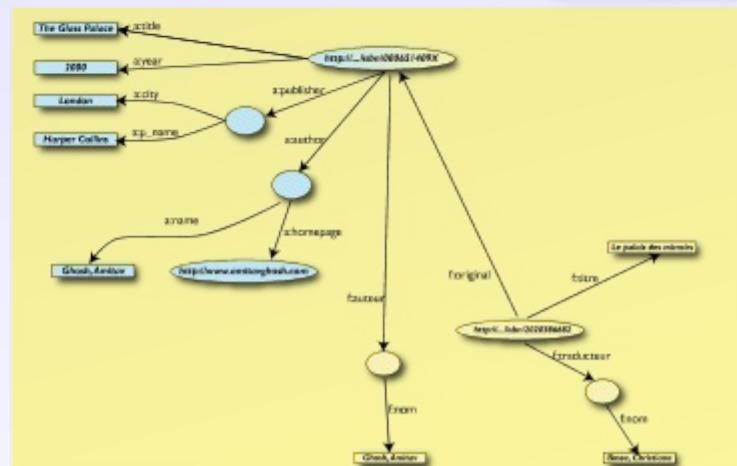
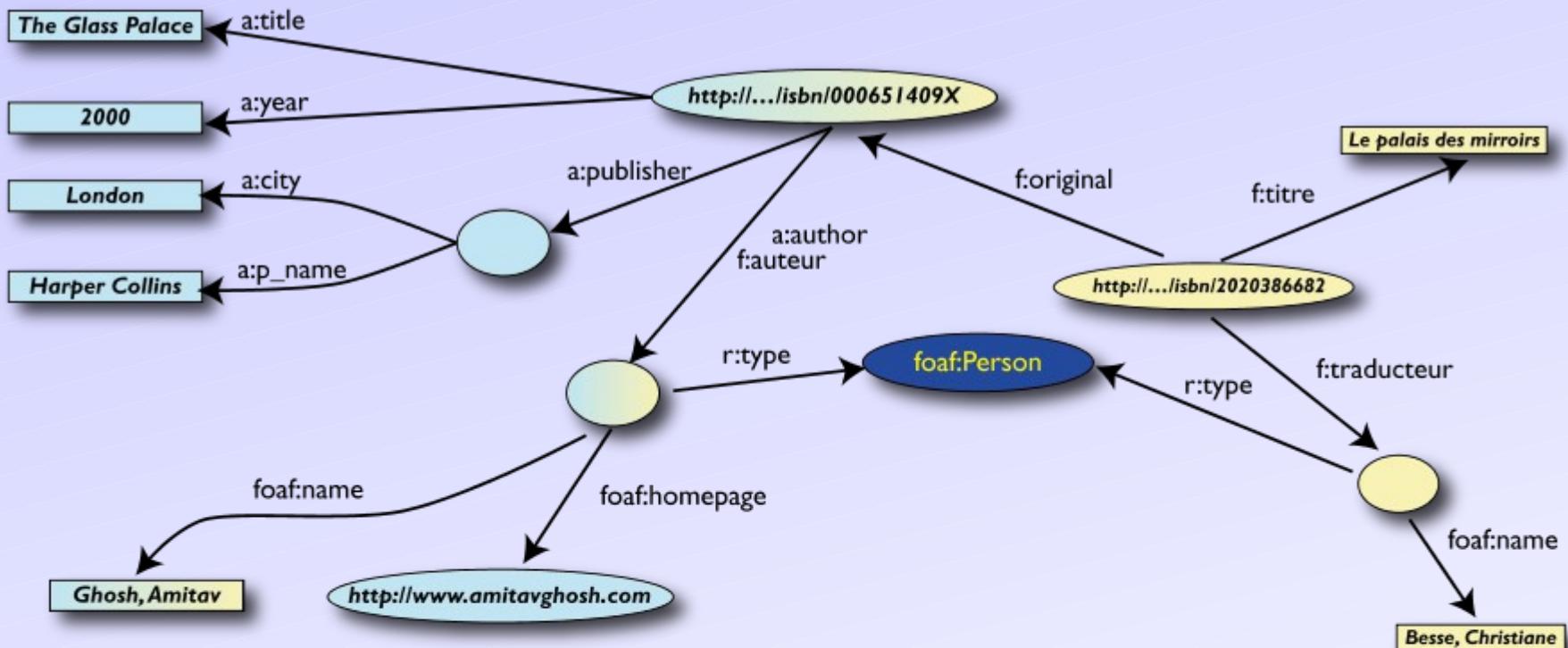
- User of data “F” can now ask queries like:
 - “give me the title of the original”
 - well, ... « donne-moi le titre de l’original »
- This information is not in the dataset “F”...
- ...but can be retrieved by merging with dataset “A”!



However, more can be achieved...

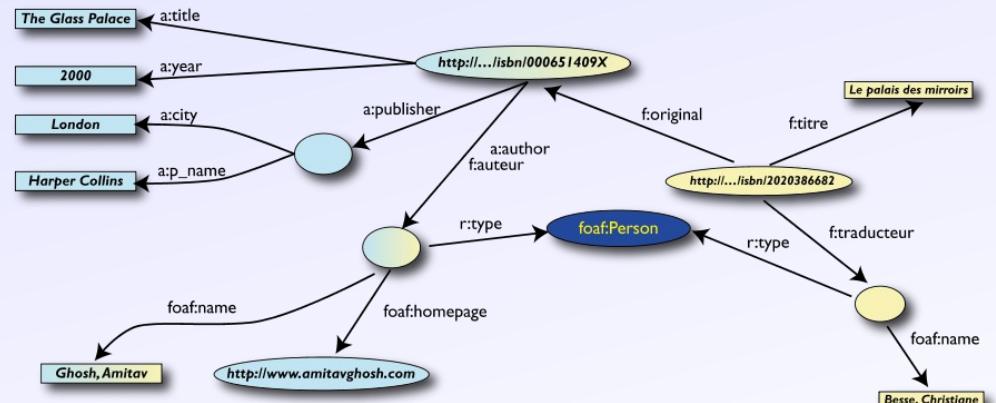
- We “feel” that **a:author** and **f:auteur** should be the same
- But an automatic merge does not know that!
- Let us add some extra information to the merged data:
 - **a:author** same as **f:auteur**
 - both identify a “Person”
 - a term that a community may have already defined:
 - a “Person” is 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!

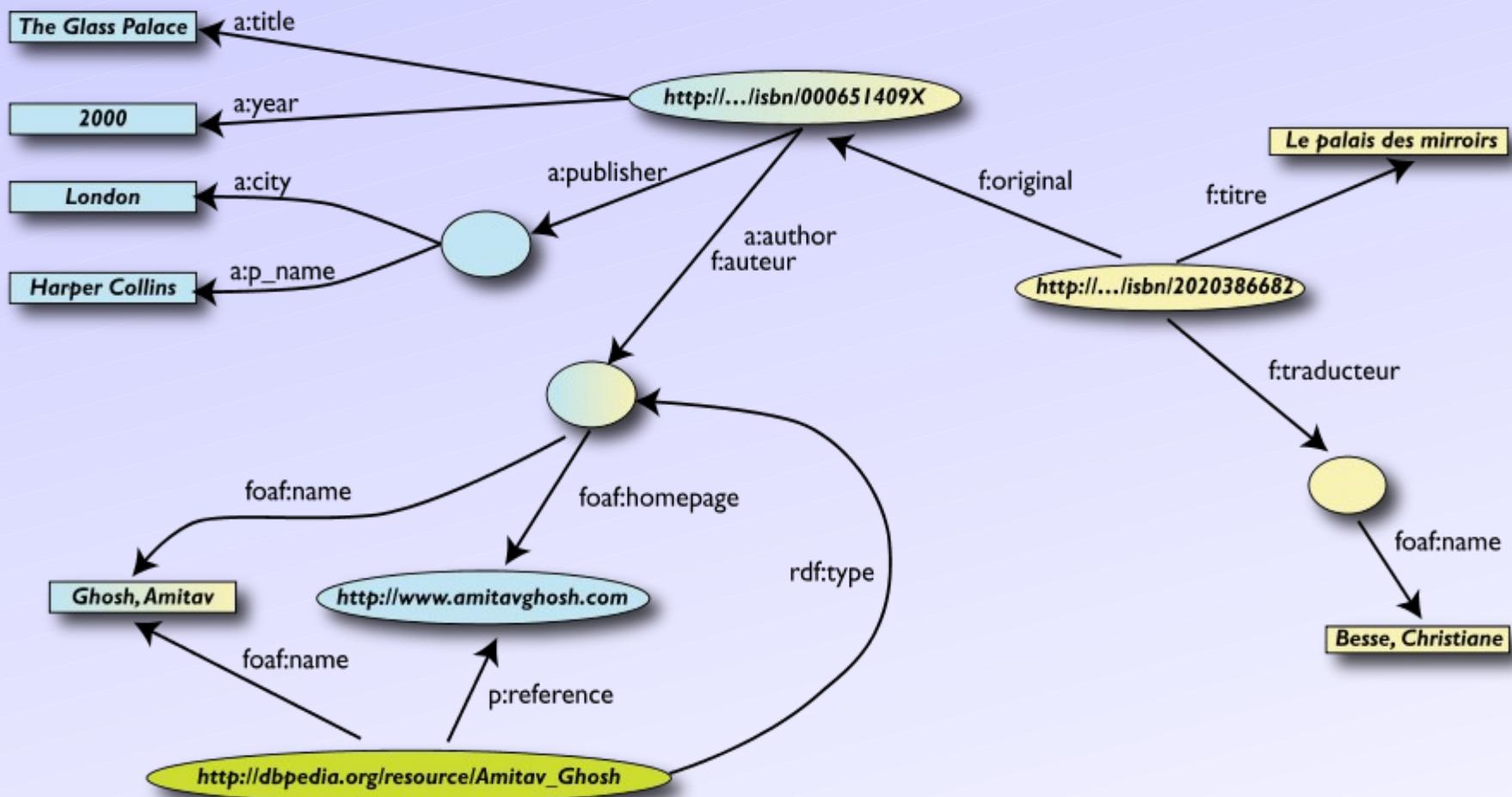
- User of dataset “F” can now query:
 - “donnes-moi la page d'accueil de l'auteur de l'originale”
 - well... “give me the home page of the original’s ‘auteur’”
- The information is not in datasets “F” or “A”...
- ...but was made available by:
 - merging datasets “A” and datasets “F”
 - adding three simple extra statements as an extra “glue”



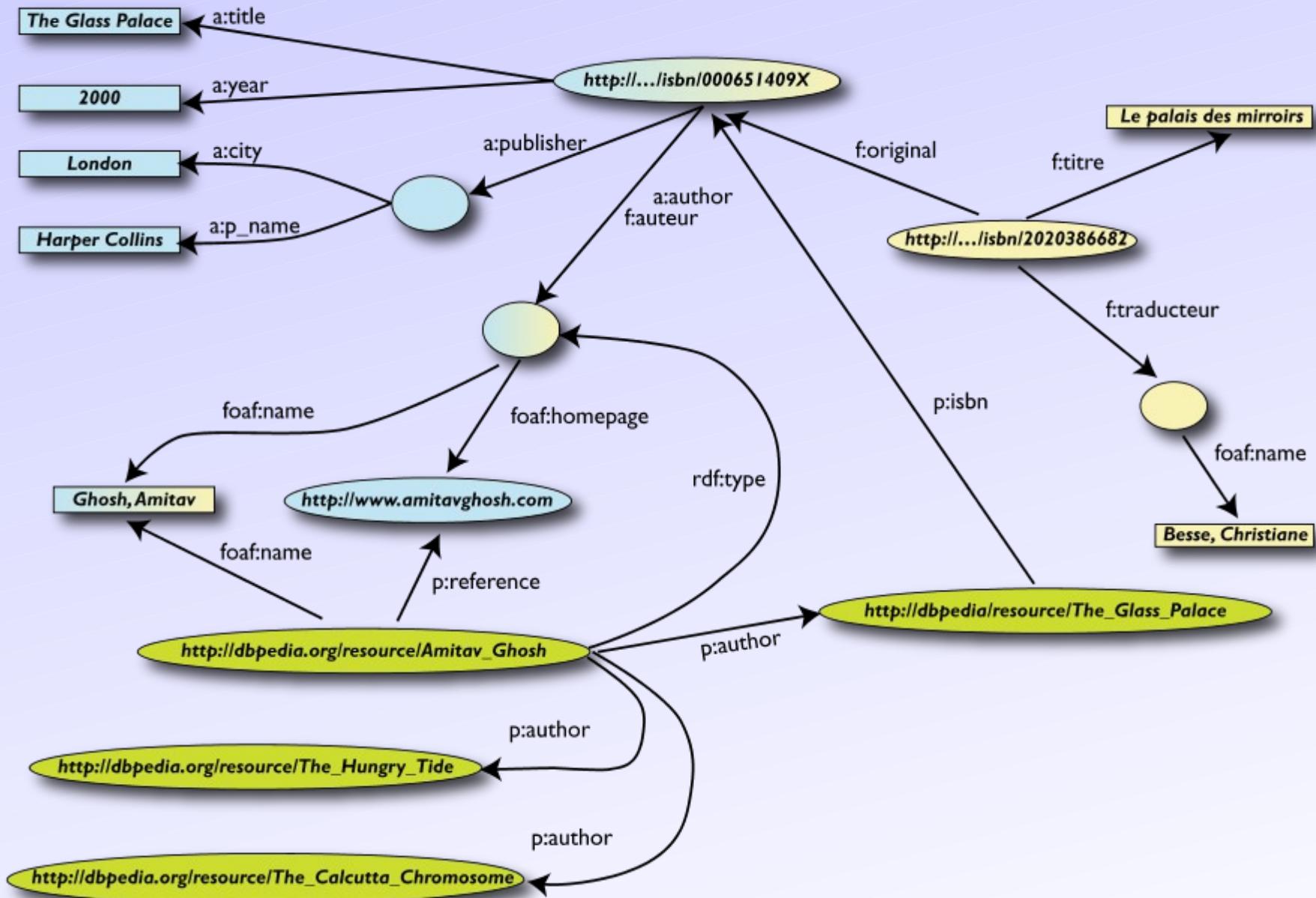
Combine with different datasets

- Using, e.g., the “Person”, the dataset can be combined with other sources
- For example, data in Wikipedia can be extracted using dedicated tools
 - e.g., the “[dbpedia](#)” project can extract the “infobox” information from Wikipedia already...

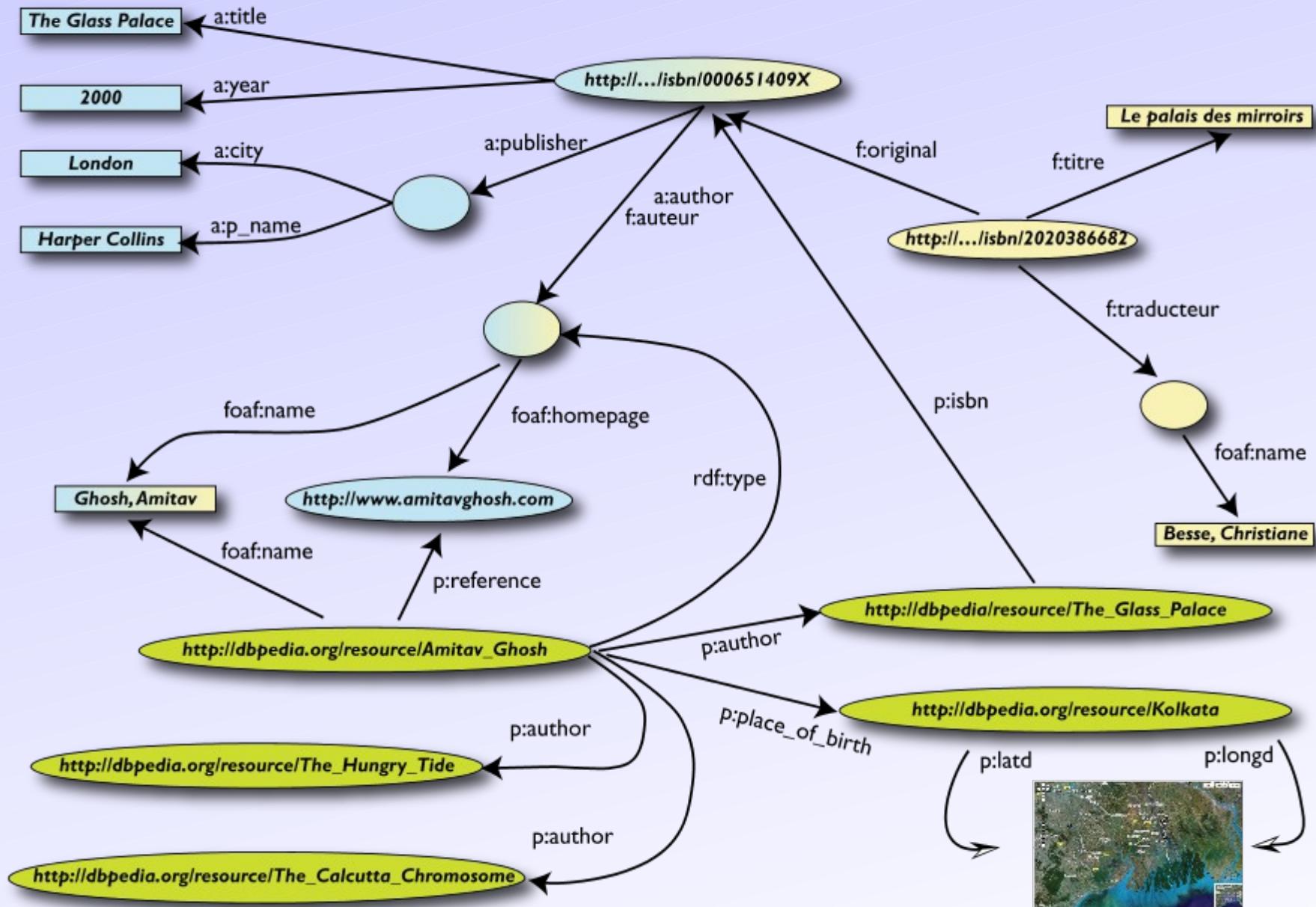
Merge with Wikipedia data



Merge with Wikipedia data



Merge with Wikipedia data



Is that surprising?

- It may look like it but, in fact, it should not be...
- 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

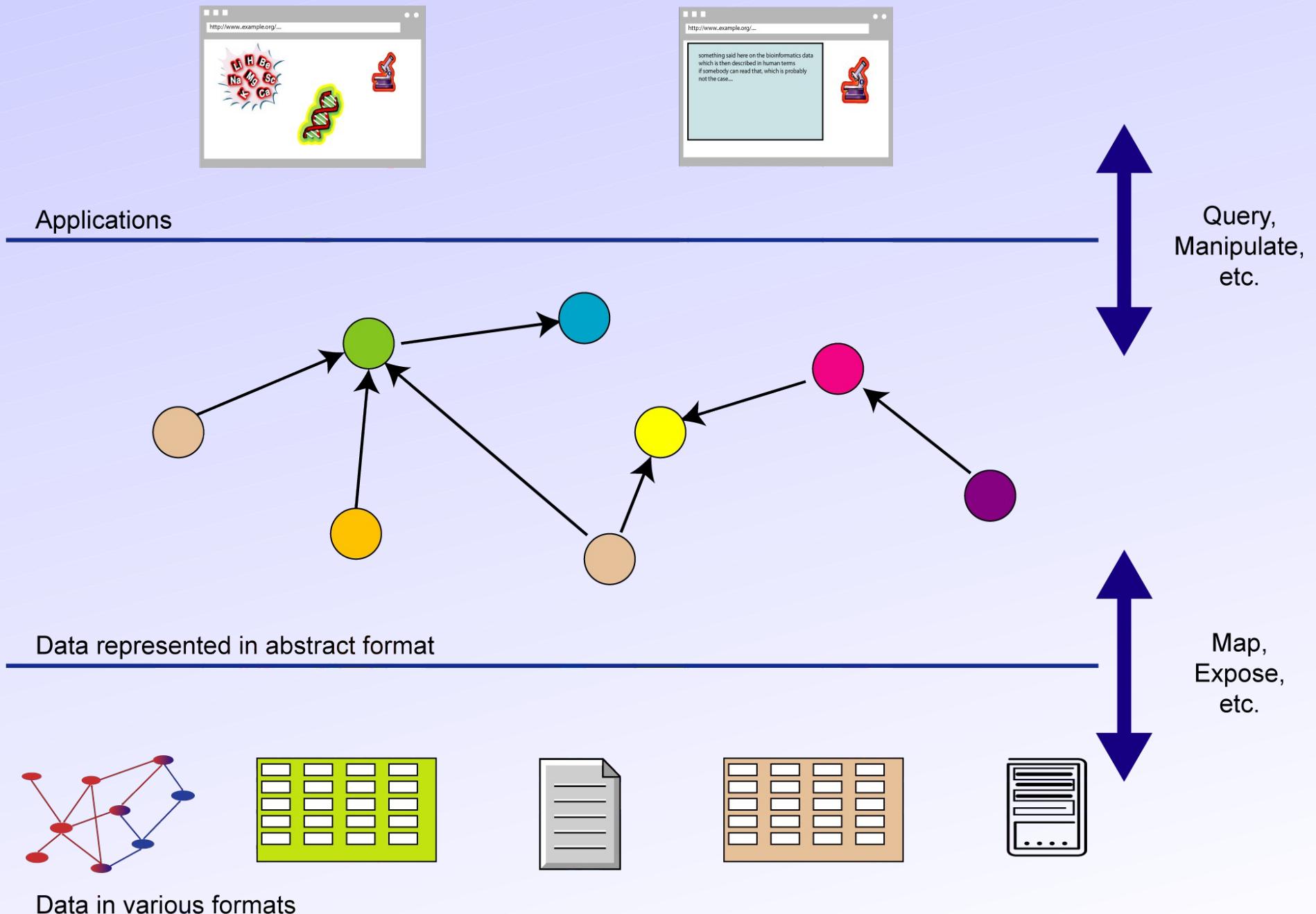
What did we do?

- We combined different datasets that
 - are somewhere on the web
 - are of different formats (mysql, excel sheet, XHTML, etc)
 - have different names for relations
- We could combine the data because some URI-s were identical (the ISBN-s in this case)
- We could add some simple additional information (the “glue”), possibly using common terminologies that a community has produced
- As a result, new relations could be found and retrieved

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)



The Basis: RDF

RDF triples

- Let us begin to formalize what we did!
 - we “connected” the data...
 - but a simple connection is not enough... data 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:

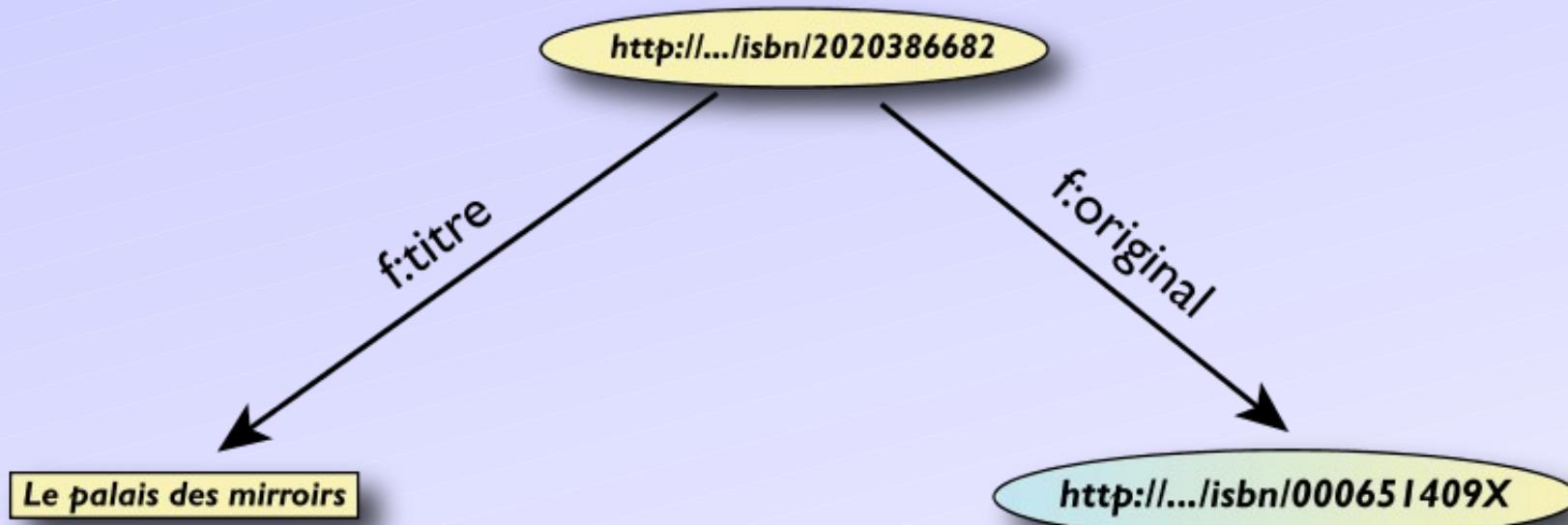
```
(<http://...isbn...6682>, <http://.../original>, <http://...isbn...409x>)
```

- 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, 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])`
- URI-s can also denote non Web entities:
 - `http://www.ivan-herman.net/me` is me
 - not my home page, not my publication list, but me
- RDF triples form a directed, labelled graph

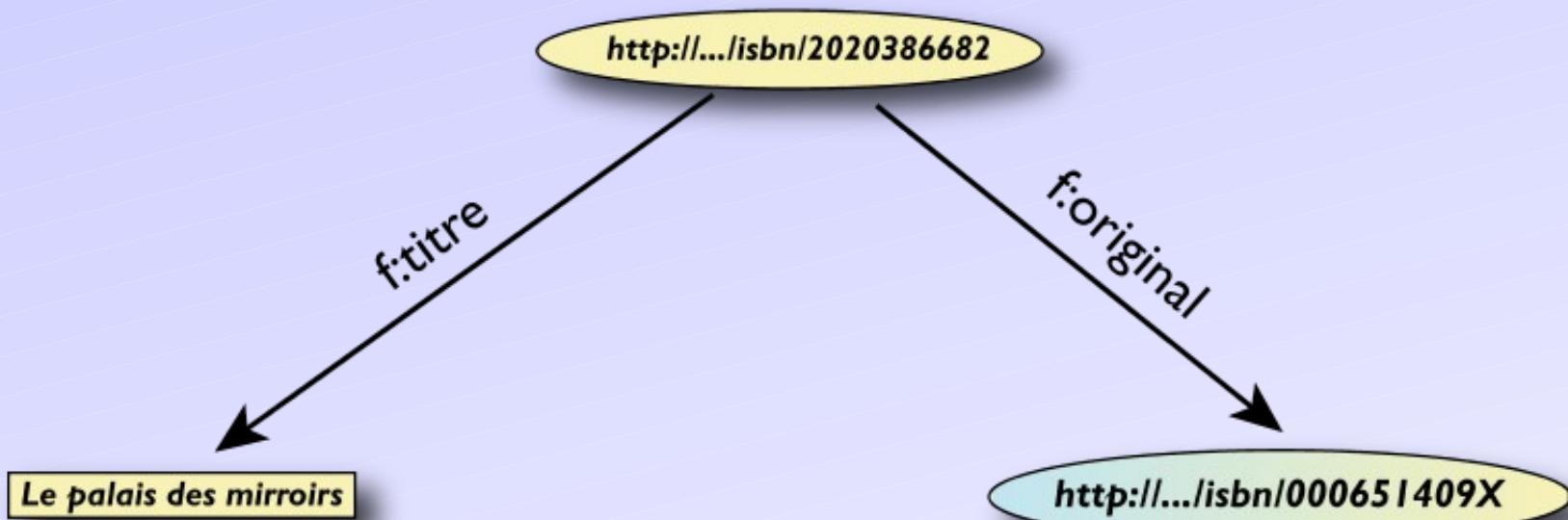
A simple RDF example (in RDF/XML)



```
<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)

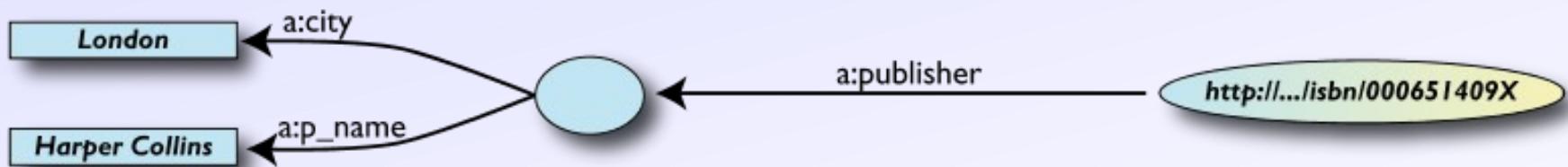
A simple RDF example (in Turtle)



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

“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»?



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>
```

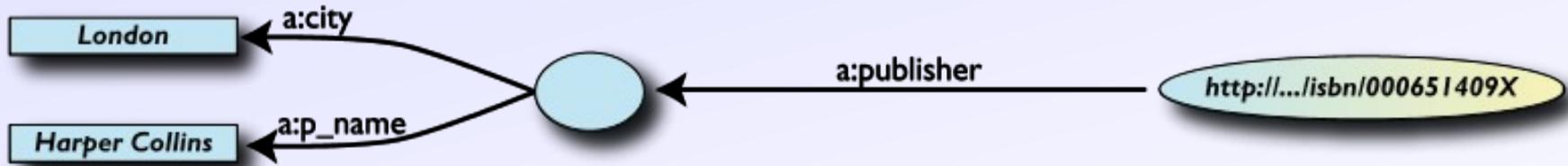
```
<http://.../isbn/2020386682> a:publisher _:A234 .
_:A234 a:p_name "HarpersCollins".
```

- 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

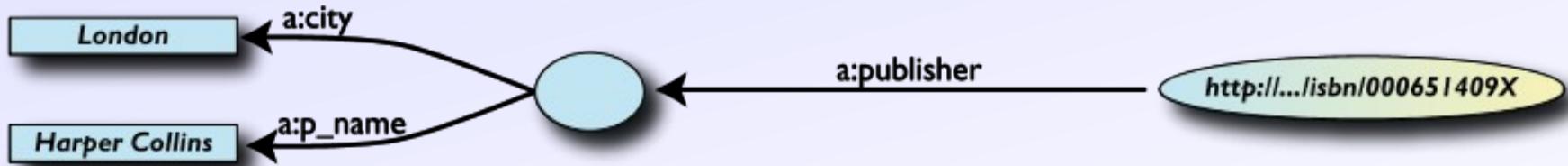
- Let the system create a “nodeID” internally (you do not really care about the name...)

```
<rdf:Description rdf:about="http://.../isbn/000651409X">
  <a:published>
    <rdf:Description>
      <a:p_name>HarpersCollins</a:p_name>
      ...
    </rdf:Description>
  </a:published>
</rdf:Description>
```



Same 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”

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

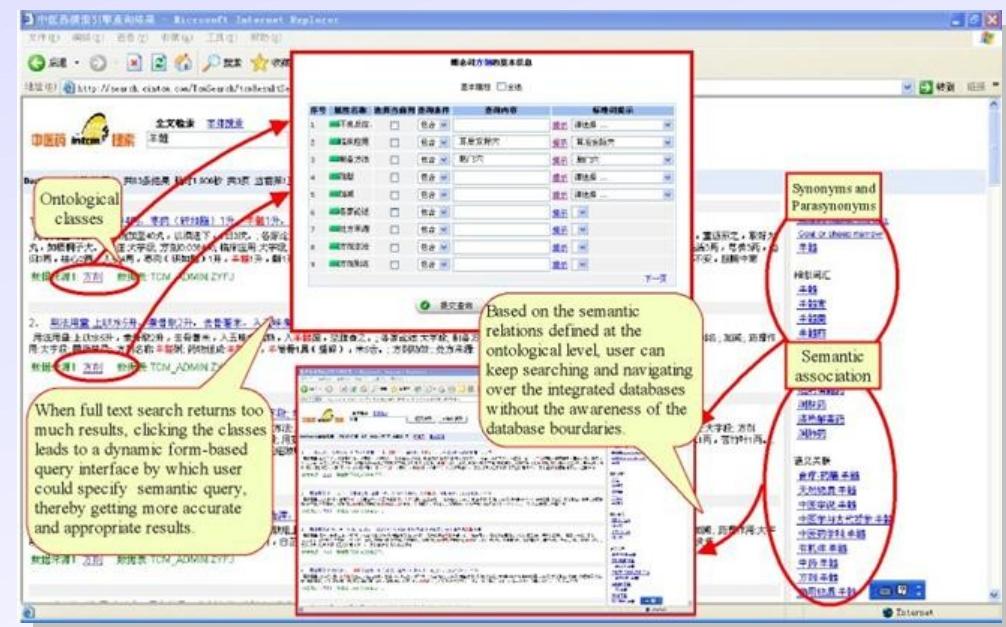
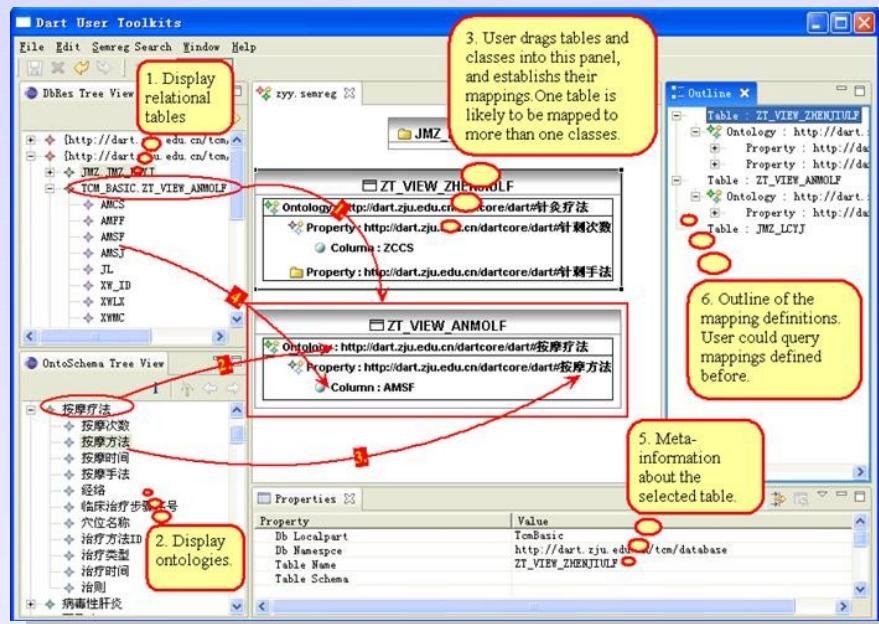
```
// create a model
Model model=new ModelMem();
Resource subject=model.createResource("URI_of_Subject")
// 'in' refers to the input file
model.read(new InputStreamReader(in));
StmtIterator iter=model.listStatements(subject,null,null);
while(iter.hasNext()) {
    st = iter.next();
    p = st.getProperty();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

Integrate knowledge for Chinese Medicine

- Integration of a large number of TCM databases
 - around 80 databases, around 200,000 records each
- Form based query interface for end users



One level higher up

(RDFS, Datatypes)

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...

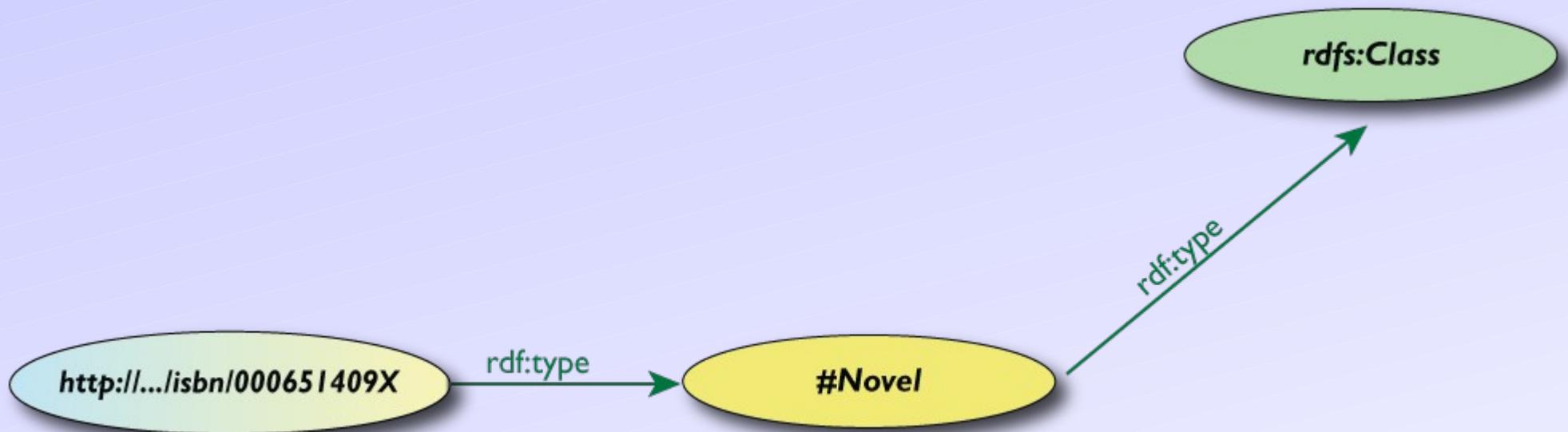
Classes, resources, ...

- 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 and 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>
```

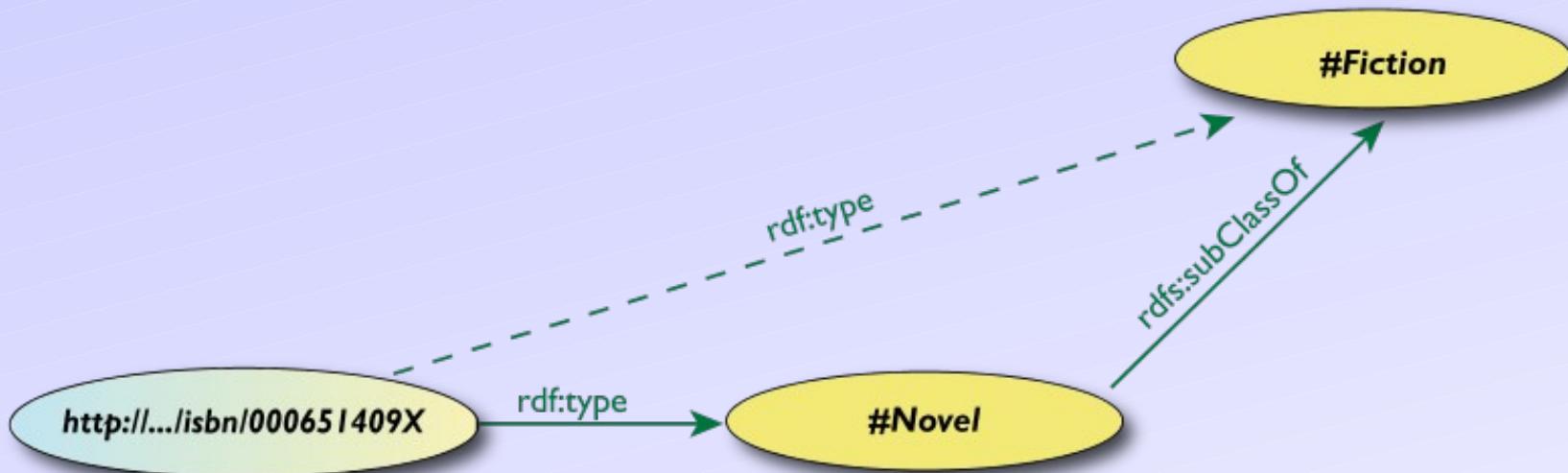
- The RDF data on a specific novel:

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

Further remarks on types

- A resource may belong to several classes
 - rdf:type is just a property...
 - “«The Glass Palace» is a novel, but «The Glass Palace» is also an «inventory item»...”
 - i.e., it is *not* like a datatype!
- 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 (33) 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
- There is also a possibility for a “sub-property”
 - all resources bound by the “sub” are also bound by the other
- Range and domain of properties can be specified
 - i.e., what type of resources serve as object and subject

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 .
```

Literals

- Literals may have a data type
 - floats, integers, booleans, etc, defined in XML Schemas
 - full XML fragments
- (Natural) language can also be specified

Examples for datatypes

```
<http://.../isbn/000651409X>
:page_number "543"^^xsd:integer ;
:publ_date    "2000"^^xsd:gYear ;
:price        "6.99"^^xsd:float .
```

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...)

Another relatively simple 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

Internal Compound Repurposing Example

Welcome, Allergy & Respiratory Team Member

This tool allows you to identify opportunities for additional uses of compounds from other teams within your project. It combines internal data, public data and the results of data mining experiments to provide testable hypotheses.

Control Panel & Item Filtering					
Area	5: Approach	3: Term+Reason	1: Max_Stage_Reached	1: Literature Links	
2: Pain	<input checked="" type="checkbox"/> 7: Antibody	<input type="checkbox"/> 3: ACTIVE	<input type="checkbox"/> 5: Candidate	<input checked="" type="checkbox"/> 0 - 50	
16: Metabolic Disease	<input checked="" type="checkbox"/> 1: Recombinant	<input type="checkbox"/> 12: BIOMARKER	<input type="checkbox"/> 10: Discovery	<input type="checkbox"/>	
3: Cancer	<input type="checkbox"/> 18: SM_Agonist	<input checked="" type="checkbox"/> 5: EFFICACY	<input checked="" type="checkbox"/> 4: Exploratory	<input type="checkbox"/>	

Internal Compound Repurposing Example

Welcome, Allergy & Respiratory Team Member

This tool allows you to identify opportunities for additional uses of compounds from other teams within your project. It combines internal data, public data and the results of data mining experiments to provide testable hypotheses.

Control Panel & Item Filtering					
Area	5: Approach	3: Term+Reason	1: Max_Stage_Reached	1: Literature Links	
2: Pain	<input checked="" type="checkbox"/> 7: Antibody	<input type="checkbox"/> 3: ACTIVE	<input type="checkbox"/> 5: Candidate	<input checked="" type="checkbox"/> 0 - 50	
16: Metabolic Disease	<input checked="" type="checkbox"/> 1: Recombinant	<input type="checkbox"/> 12: BIOMARKER	<input type="checkbox"/> 10: Discovery	<input type="checkbox"/>	
3: Cancer	<input type="checkbox"/> 18: SM_Agonist	<input checked="" type="checkbox"/> 5: EFFICACY	<input checked="" type="checkbox"/> 4: Exploratory	<input type="checkbox"/>	
3: Sexual Health	<input checked="" type="checkbox"/> 12: SM_Antagonist	<input type="checkbox"/> 11: MARKET	<input type="checkbox"/> 19: HTS	<input type="checkbox"/>	
2: Infectives	<input checked="" type="checkbox"/> 21: SM_Inhibitor	<input type="checkbox"/> 11: REORG	<input type="checkbox"/> 11: Phase I	<input type="checkbox"/>	
1: Urogenitals	<input checked="" type="checkbox"/>	<input type="checkbox"/> 10: TOXIC	<input type="checkbox"/> 13: Phase III	<input type="checkbox"/>	
			<input type="checkbox"/> 4: Screening	<input type="checkbox"/>	

51 items filtered from 710 originally (Reset All Filters)

Area	Original + Indication	Target_Name	Approach	Start	Term+Reason	Max_Stage_Reached	Owner	OBIM Lit_All Lit_2007 Lit_Mech IMA GEO Pathway Compounds
Metabolic Disease	Diabetes	Liver glycogen phosphorylase	SM_Inhibitor	2007-Q2	EFFICACY	Candidate	P. Person	SW-030072
Sexual Health	Erectile Dysfunction	Integrin alpha-3 (Glycoprotein 33/Mu3) (CD49c)	SM_Antagonist	2006-Q3	EFFICACY	Candidate	P. Person	SW-029782
Sexual Health	Erectile Dysfunction	Leukotriene C4 synthase	SM_Agonist	2006-Q3	EFFICACY	Candidate	M. Manager	SW-029638
Sexual Health	Erectile Dysfunction	transcription elongation factor A (STF)-like 4	SM_Inhibitor	2005-Q2	EFFICACY	Candidate	P. Person	SW-029926
Infectives	HIV	Putative four-repeat ion channel (ix)	SM_Inhibitor	2006-Q2	EFFICACY	Candidate	L. Leader	SW-029994
Infectives	HIV	Voltage-gated potassium channel protein KV.2 (Ix)	SM_Agonist	2007-Q1	EFFICACY	Candidate	A. Scientist	SW-029653
Urogenital Incontinence		Human RNA binding motif (RBM) gene, partial cdk.	SM_Agonist	2007-Q3	EFFICACY	Candidate	L. Leader	SW-029684
Pain	Migraine	Monocarboxylate transporter homologue294064CD1 (Ix)	SM_Inhibitor	2007-Q3	EFFICACY	Candidate	L. Leader	SW-030085

How to get RDF Data?

(Microformats, GRDDL, RDFa)

Simple approach

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

RDF with XHTML and XML

- Obviously, a huge source of information
- By adding some “meta” information, the same source can be reused for, eg, data integration, better mashups, etc
 - typical example: your personal information, like address, should be readable for humans and processable by machines
- Two solutions have emerged:
 - extract the structure from the page and convert the content into RDF
 - add RDF statements directly into XHTML via RDFa

Extract RDF

- 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)
- GRDDL formalizes the this general scheme

Formalizing the scraper approach: GRDDL

- GRDDL formalizes the scraper approach. For example:

```
<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, through **dc-extract.xsl**:

```
<>
  dc:subject "Some subject";
  dc:date "2006-01-02" .
```

GRDDL with XML

- The approach is very similar to the XHTML case
- The appropriate attributes are added to the XML namespace document
- Otherwise it is identical

Bridge 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, ...)
- A survey on mapping techniques has been published at W3C
- A W3C group has just started to standardize this

Linking Data

Linking Open Data Project

- Goal: “expose” open datasets in RDF
- Set *RDF links among the data items* from different datasets
- Set up query endpoints
- Altogether billions of triples, millions of links...



Example data source: DBpedia

- DBpedia is a community effort to
 - extract structured (“infobox”) information from Wikipedia
 - provide a query endpoint to the dataset
 - interlink the DBpedia dataset with other datasets on the Web



UNIVERSITÄT LEIPZIG



Extracting Wikipedia structured data

Amsterdam	
	
The Keizersgracht at dusk	
Location of Amsterdam	
Coordinates:	52°22'23"N 4°53'32"E
Country	Netherlands
Province	North Holland
Government	
- Type	Municipality
- Mayor	Job Cohen ^[1] (PvdA)
- Aldermen	Lodewijk Asscher Carolien Gehrels Tjeerd Herrema Maarten van Poelgeest Marijke Vos Erik Gerritsen
- Secretary	
Area ^{[2][3]}	
- City	219 km ² (84.6 sq mi)
- Land	166 km ² (64.1 sq mi)
- Water	53 km ² (20.5 sq mi)
- Urban	1,003 km ² (387.3 sq mi)
- Metro	1,815 km ² (700.8 sq mi)
Elevation ^[4]	2 m (7 ft)
Population ^{(1 October 2008)[5][6]}	
- City	755,269
- Density	4,459/km ² (11,548.8/sq mi)
- Urban	1,364,422
- Metro	2,158,372
- Demonym	Amsterdamer
Time zone	CET (UTC+1)
- Summer (DST)	CEST (UTC+2)
Postcodes	1011 – 1109
Area code(s)	020
Website: www.amsterdam.nl	

```
@prefix dbpedia <http://dbpedia.org/resource/>.
```

```
@prefix dbterm <http://dbpedia.org/property/>.
```

dbpedia:Amsterdam

```
dbterm:officialName "Amsterdam" ;
```

```
dbterm:longd "4" ;
```

```
dbterm:longm "53" ;
```

```
dbterm:longs "32" ;
```

```
...
```

```
dbterm:leaderTitle "Mayor" ;
```

```
dbterm:leaderName dbpedia:Job_Cohen ;
```

```
...
```

```
dbterm:areaTotalKm "219" ;
```

```
...
```

dbpedia:ABN_AMRO

```
dbterm:location dbpedia:Amsterdam ;
```

```
...
```

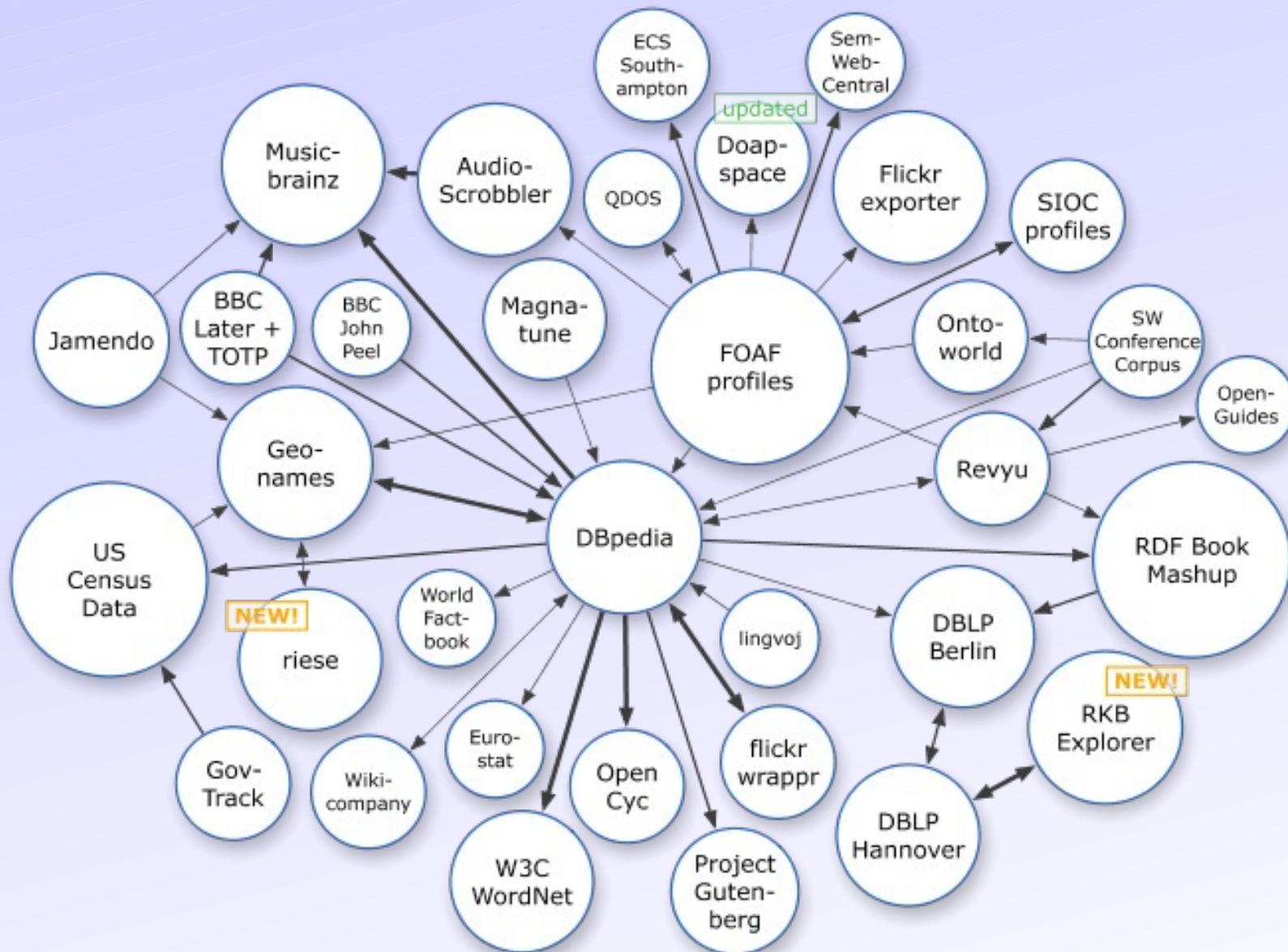
Automatic links among open datasets

```
<http://dbpedia.org/resource/Amsterdam>
owl:sameAs <http://rdf.freebase.com/ns/...> ;
owl:sameAs <http://sws.geonames.org/2759793> ;
...
```

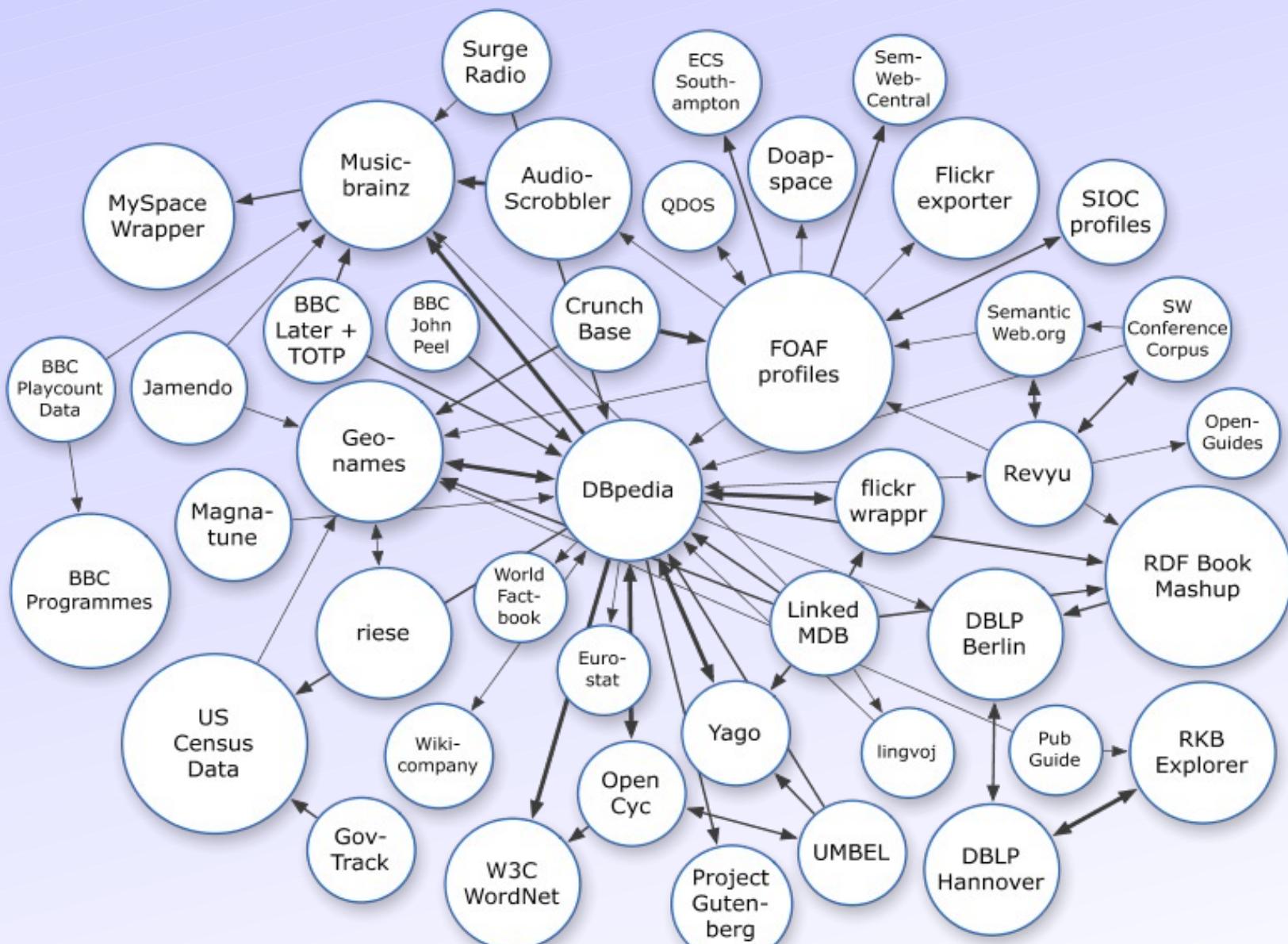
```
<http://sws.geonames.org/2759793>
owl:sameAs <http://dbpedia.org/resource/Amsterdam>
wgs84_pos:lat "52.3666667" ;
wgs84_pos:long "4.8833333" ;
geo:inCountry <http://www.geonames.org/countries/#NL> ;
...
```

Processors can switch automatically from one to the other...

The LOD “cloud”, March 2008

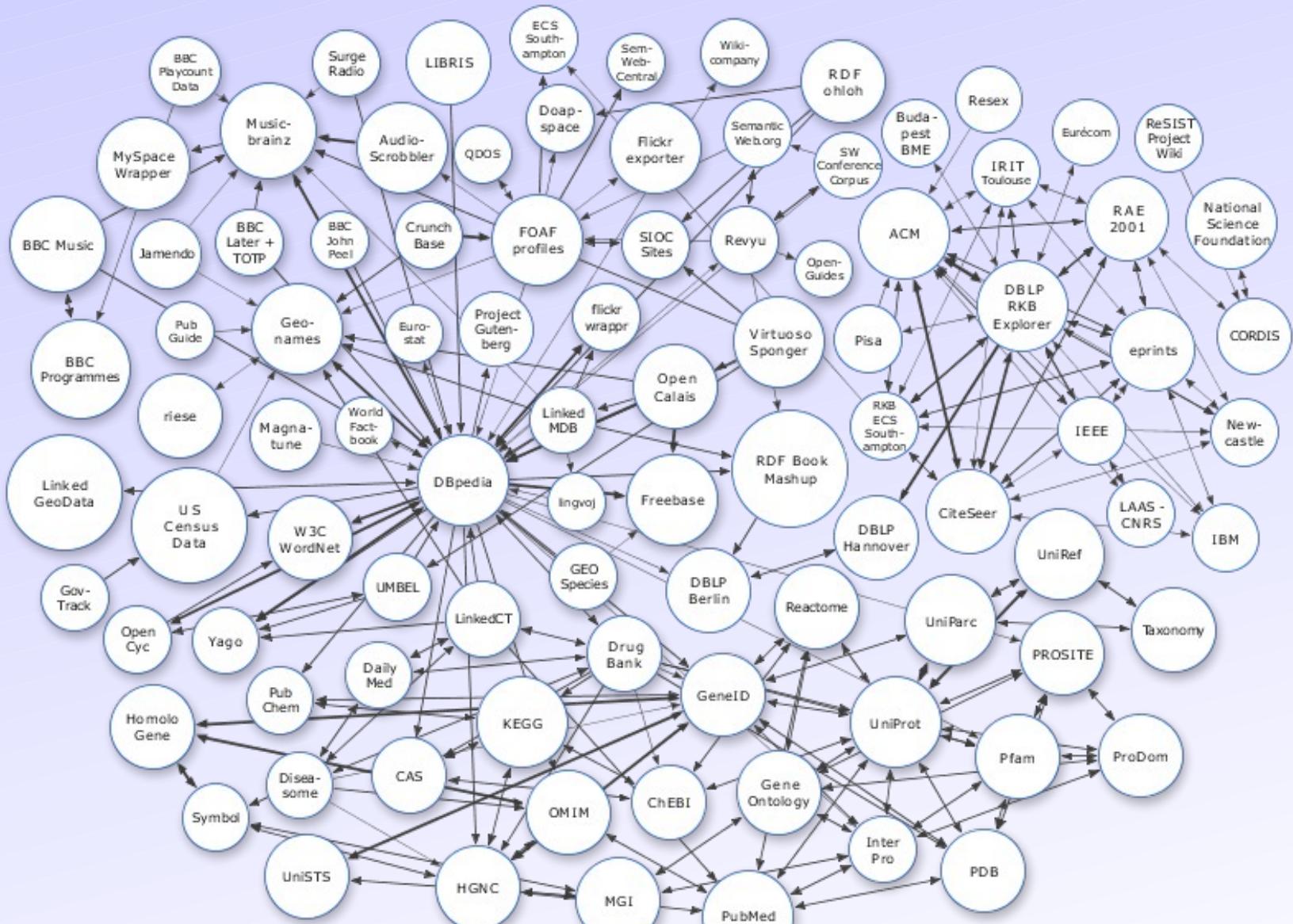


The LOD “cloud”, September 2008



As of September 2008

The LOD “cloud”, July 2009



As of July 2009

Using the LOD to build Web site: BBC

BBC - Music - Eric Clapton - Mozilla Firefox

File Edit View History Bookmarks Tools Help

BBC http://www.bbc.co.uk/music/artists/618b6900-0618-4f1e-b835-bccb17f84294.html Dogpile ABP

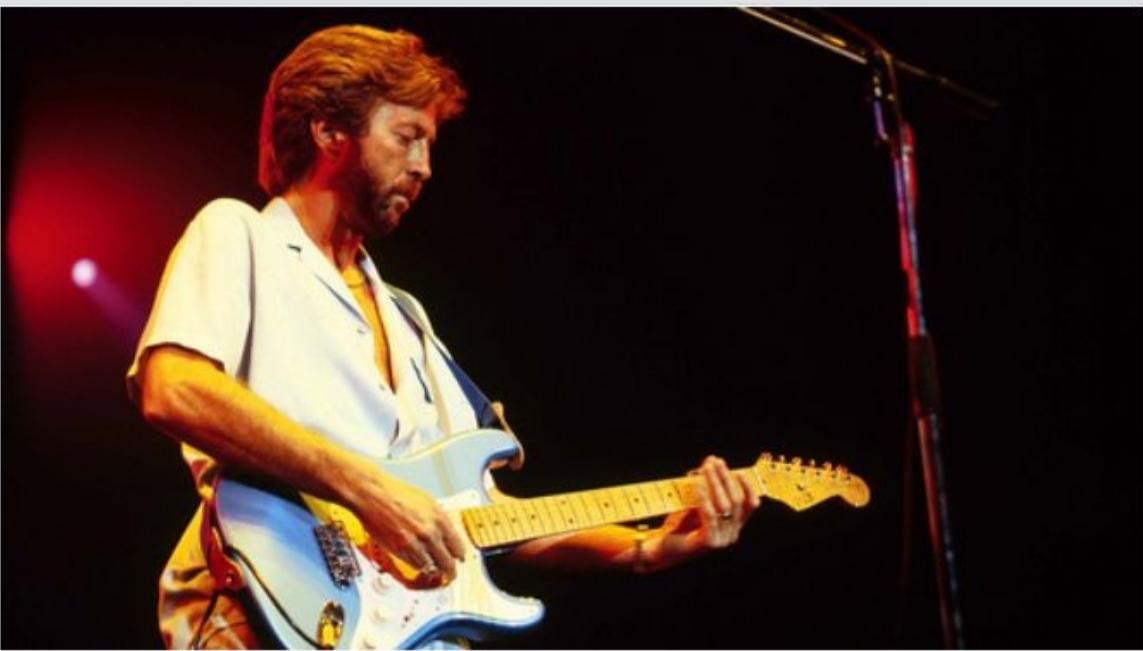
BBC BBC - Music - Eric Clapton

Music BETA

BBC Music > Artists > Eric Clapton

Eric Clapton

Born 30 March 1945.



MOST PLAYED ON BBC RADIO 2

Played By

Since December 2008

Steve Wright in the Afternoon
2 BBC Radio 2

Steve Wright's afternoon show with special guests and other features

Ken Bruce
2 BBC Radio 2

The best in music every weekday with Ken Bruce sessions

Chris Hawkins
6 BBC 6 Music

Join Chris for regular great music and a new show

Done

Using the LOD to build Web site: BBC

BBC - Music - Eric Clapton - Mozilla Firefox

File Edit View History Bookmarks Tools Help

<http://www.bbc.co.uk/music/artists/618b6900-0618-4f1e-b835-bccb17f84294.html>

BBC BBC - Music - Eric Clapton



“ “ ...this hits the spot nicely and underlines just why Clapton is so revered amongst...

More Eric Clapton Releases »

Credits

ROLE	ARTIST	RELEASE
Instrument	George Harrison	All Things Must Pass (1987)
Instrument	Roger Waters	The Pros and Cons of Hitch Hiking (1984)
Performer	T.D.F.	Retail Therapy

Credits comes from MusicBrainz. You can add or edit information about Eric Clapton at musicbrainz.org. Find out more about our use of this data .

More Eric Clapton Credits »

Blind Faith

Cream

Derek and the Dominos

John Mayall & The Bluesbreakers

The Yardbirds

Information about group members comes from MusicBrainz. You can add or edit information about Eric Clapton at musicbrainz.org. Find out more about our use of this data .

Connected Artists

COLLABORATED ON

J.J. Cale & Eric Clapton

Eric Clapton & The Immediate All Stars

Eric Clapton & The Impressions

Connected artists information comes from MusicBrainz. You can add or edit information about Eric Clapton at musicbrainz.org. Find out more about our use of this data .

More Eric Clapton Connected Artists

Links

- Official homepage at ericclapton.com
- Fanpage at whereseric.com
- Wikipedia article on Eric Clapton
- MySpace at myspace.com/ericclapton
- Last.fm page on Eric Clapton
- MusicBrainz entry on Eric Clapton

Done

Using the LOD to build Web site: BBC

Mozilla Firefox

File Edit View History Bookmarks Tools Help

BBC http://www.bbc.co.uk/music/artists/618b6900-0618-4f1e-b835-bccb17f84294.rdf

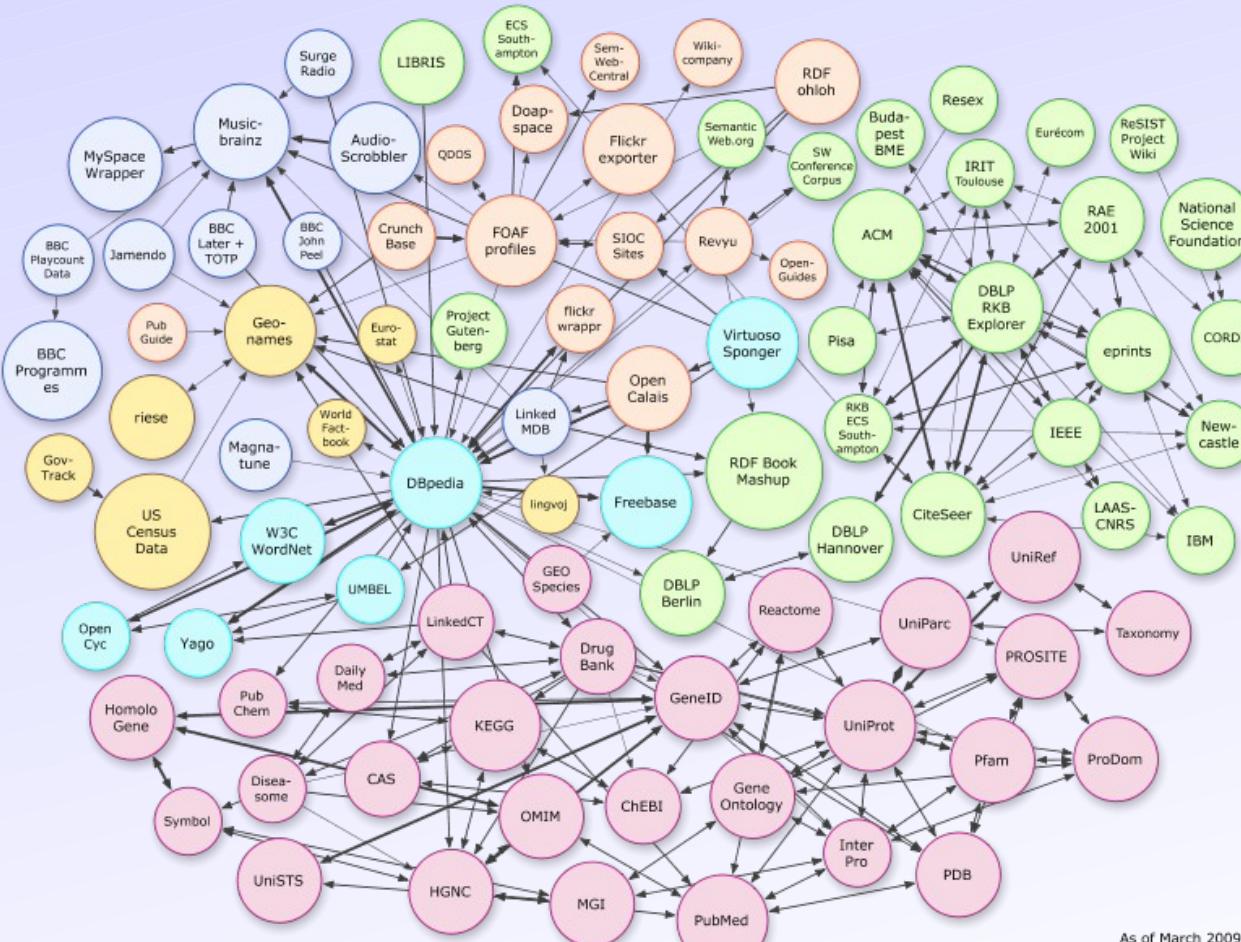
```

<mo:MusicBrainz rdf:resource="http://musicbrainz.org/artist/618b6900-0618-4f1e-b835-bccb17f84294.html">
  <foaf:homepage rdf:resource="http://www.ericclapton.com"/>
  <mo:fanpage rdf:resource="http://www.whereseric.com"/>
  <mo:wikipedia rdf:resource="http://en.wikipedia.org/wiki/Eric_Clapton"/>
  <mo:myspace rdf:resource="http://www.myspace.com/ericclapton"/>
  <mo:member_of rdf:resource="/music/artists/53fa91ca-a2b9-463d-b78e-daca9894082a#artist"/>
  <mo:member_of rdf:resource="/music/artists/04cd0cf0-bfd1-4c36-bc38-95c35e2c045f#artist"/>
  <mo:member_of rdf:resource="/music/artists/2155a81a-f0c6-417a-9b16-2f86f98bb8bc#artist"/>
  <mo:member_of rdf:resource="/music/artists/4756395c-57ed-4a63-afb2-011117f14dff6#artist"/>
  <mo:member_of rdf:resource="/music/artists/191de76f-a224-445d-b041-54df16d65bf7#artist"/>
  - <foaf:made>
    - <mo:Record>
      <dc:title>Me and Mr. Johnson</dc:title>
      <mo:musicbrainz rdf:resource="http://musicbrainz.org/release/cf83ac25-374f-4cd4-9872-c6c00aaced92.html"/>
      <rev:hasReview rdf:resource="/music/reviews/5dqv#review"/>
    </mo:Record>
  </foaf:made>
  - <foaf:made>
    - <mo:Record>
      <dc:title>Martin Scorsese Presents the Blues: Eric Clapton</dc:title>
      <mo:musicbrainz rdf:resource="http://musicbrainz.org/release/a36c1d21-5669-44d6-969c-179fb6039359.html"/>
      <rev:hasReview rdf:resource="/music/reviews/6jxm#review"/>
    </mo:Record>
  </foaf:made>
</mo:MusicArtist>
- <mo:MusicArtist rdf:about="/music/artists/53fa91ca-a2b9-463d-b78e-daca9894082a#artist">
  <foaf:name>Blind Faith</foaf:name>
</mo:MusicArtist>
- <mo:MusicArtist rdf:about="/music/artists/04cd0cf0-bfd1-4c36-bc38-95c35e2c045f#artist">
  <foaf:name>Cream</foaf:name>
</mo:MusicArtist>
- <mo:MusicArtist rdf:about="/music/artists/2155a81a-f0c6-417a-9b16-2f86f98bb8bc#artist">

```

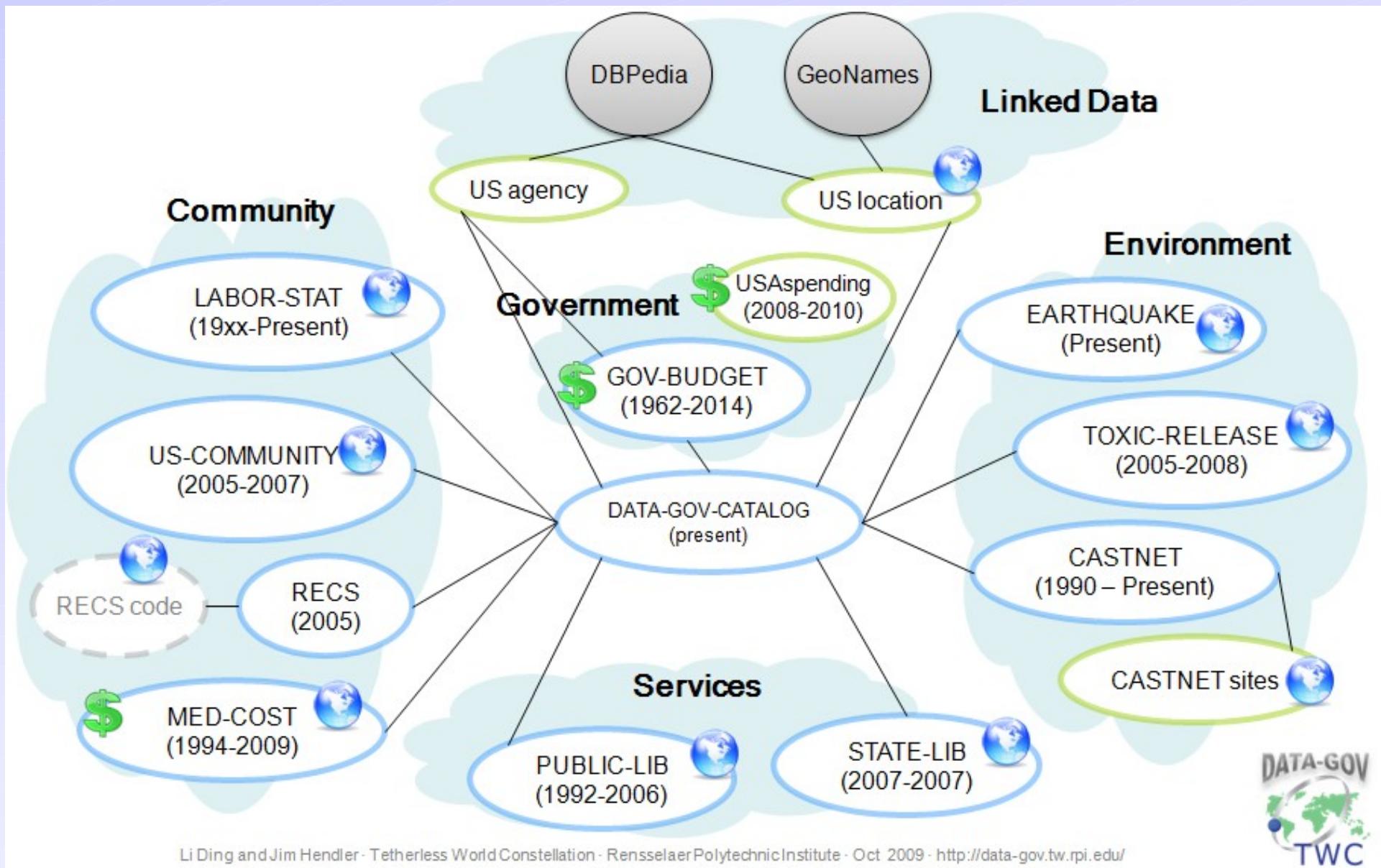
Application specific portions of the cloud

- Eg, “bio” related datasets
 - done, partially, by the “Linking Open Drug Data” task force of the HCLS IG at W3C



As of March 2009

Linked Open eGov Data



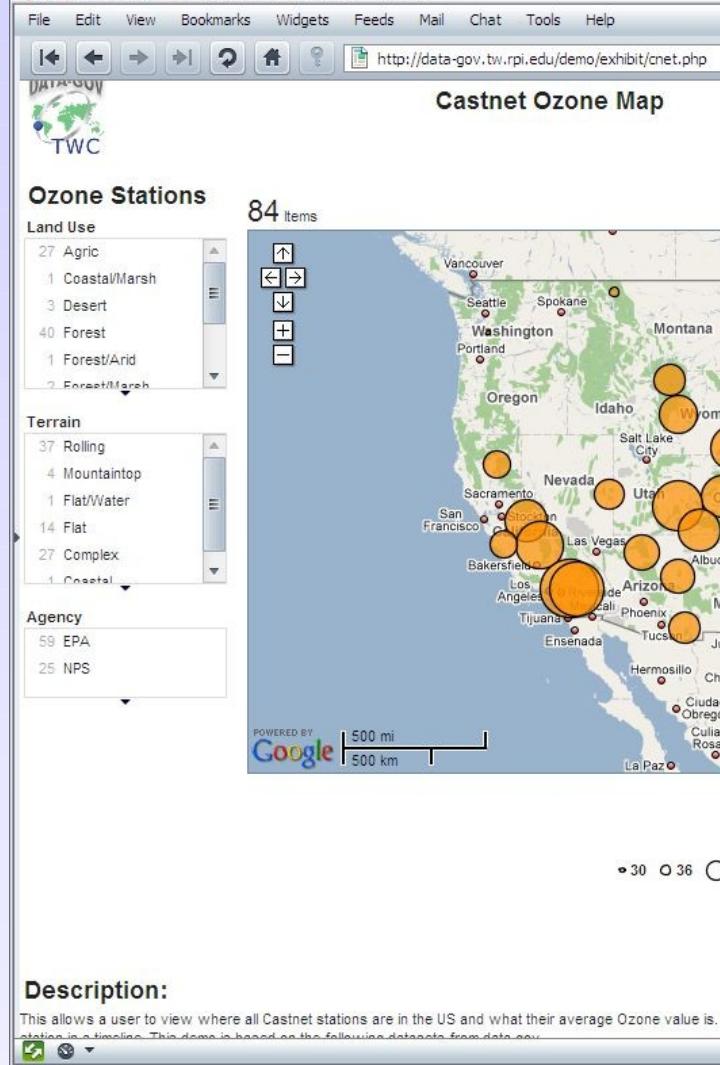
You publish the raw data, we use it...

Individual and Corporate Tax Receipts

From [Dataset 403](#)



Castnet Ozone Map: stations and readings - Opera



Earthquakes From the Past Day

From [Dataset 33](#)



Query RDF Data (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:

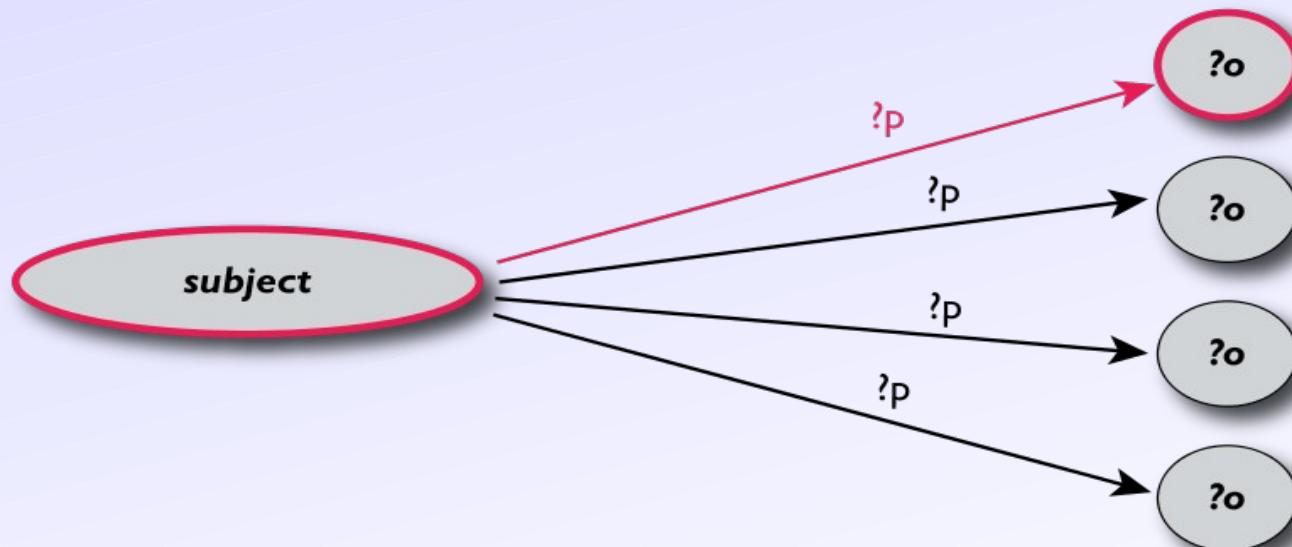
```
StmtIterator iter=model.listStatements(subject,null,null);  
while(iter.hasNext()) {  
    st = iter.next();  
    p = st.getProperty(); o = st.getObject();  
    do_something(p,o);
```

- 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)

Analyse 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 $(\text{subject}, ?p, ?o)$ is a *pattern* for what we are looking for (with $?p$ and $?o$ as “unknowns”)



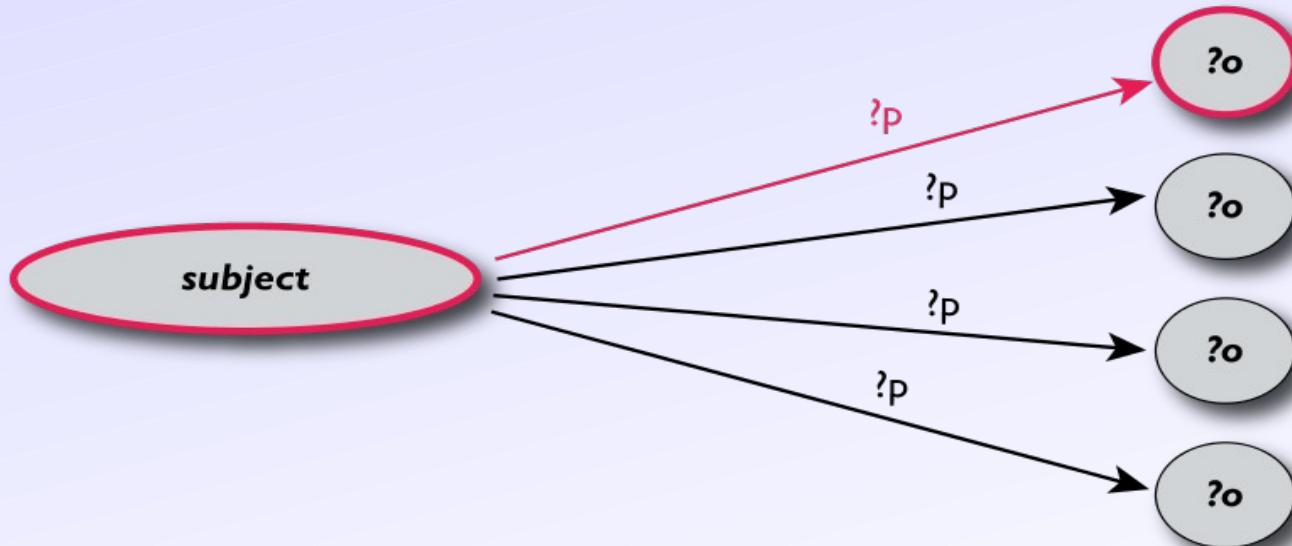
General: graph patterns

- 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

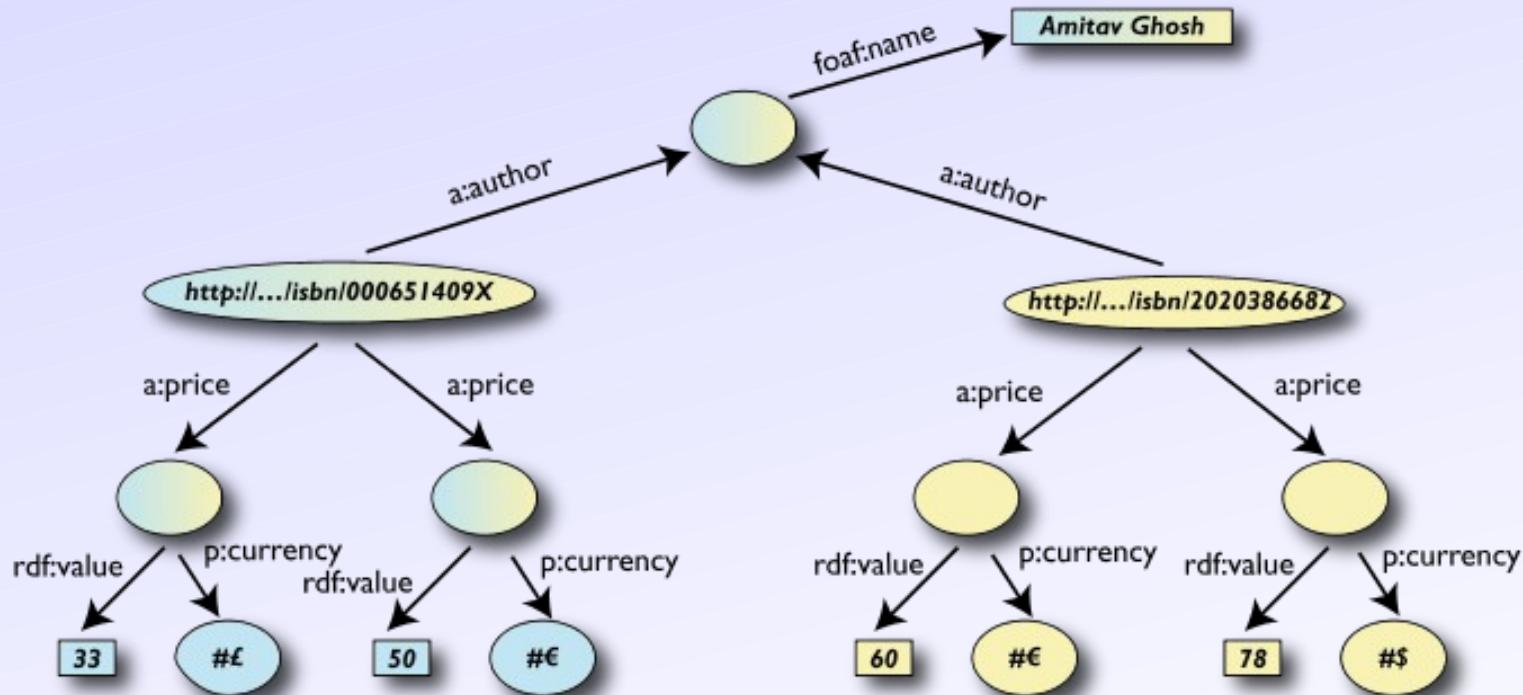
```
SELECT ?p ?o  
WHERE {subject ?p ?o}
```

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



Simple SPARQL example

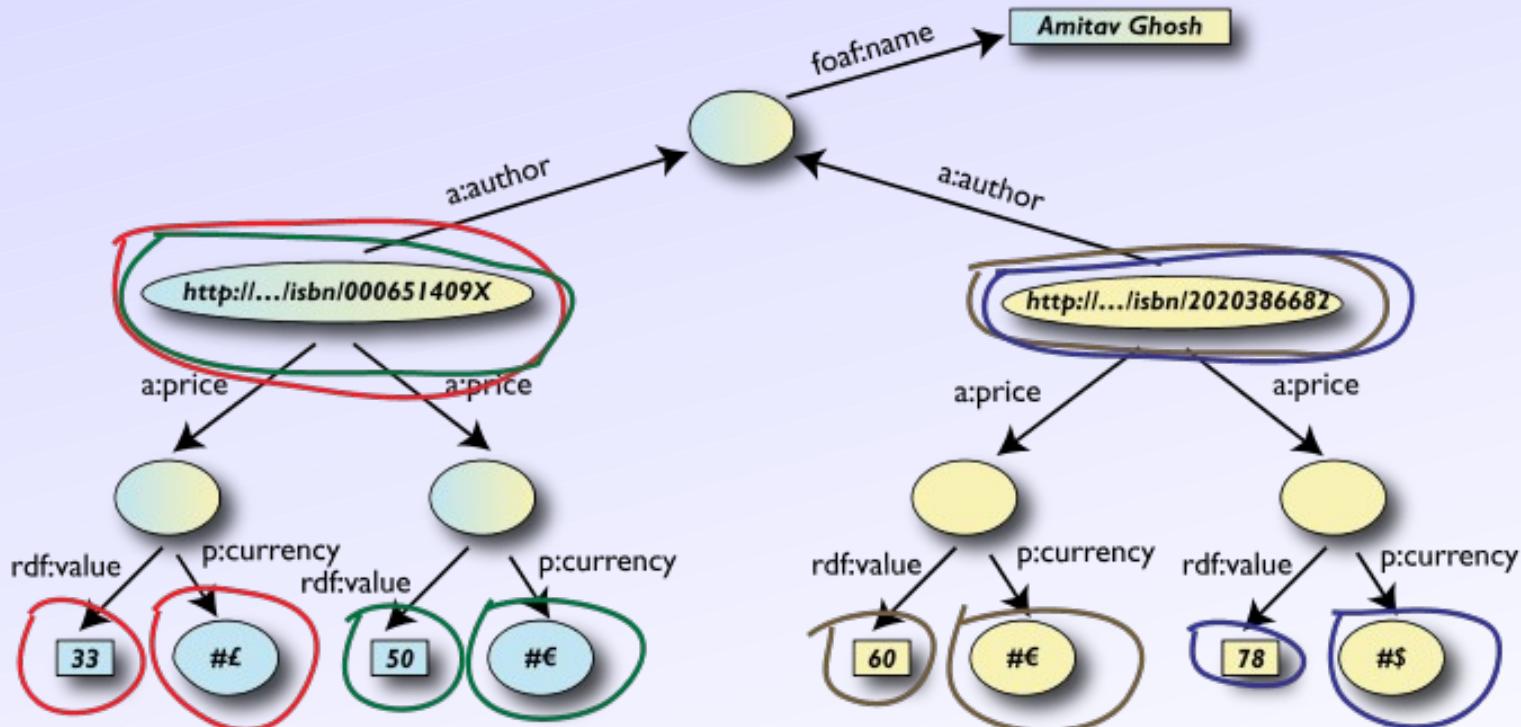
```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



Simple SPARQL example

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

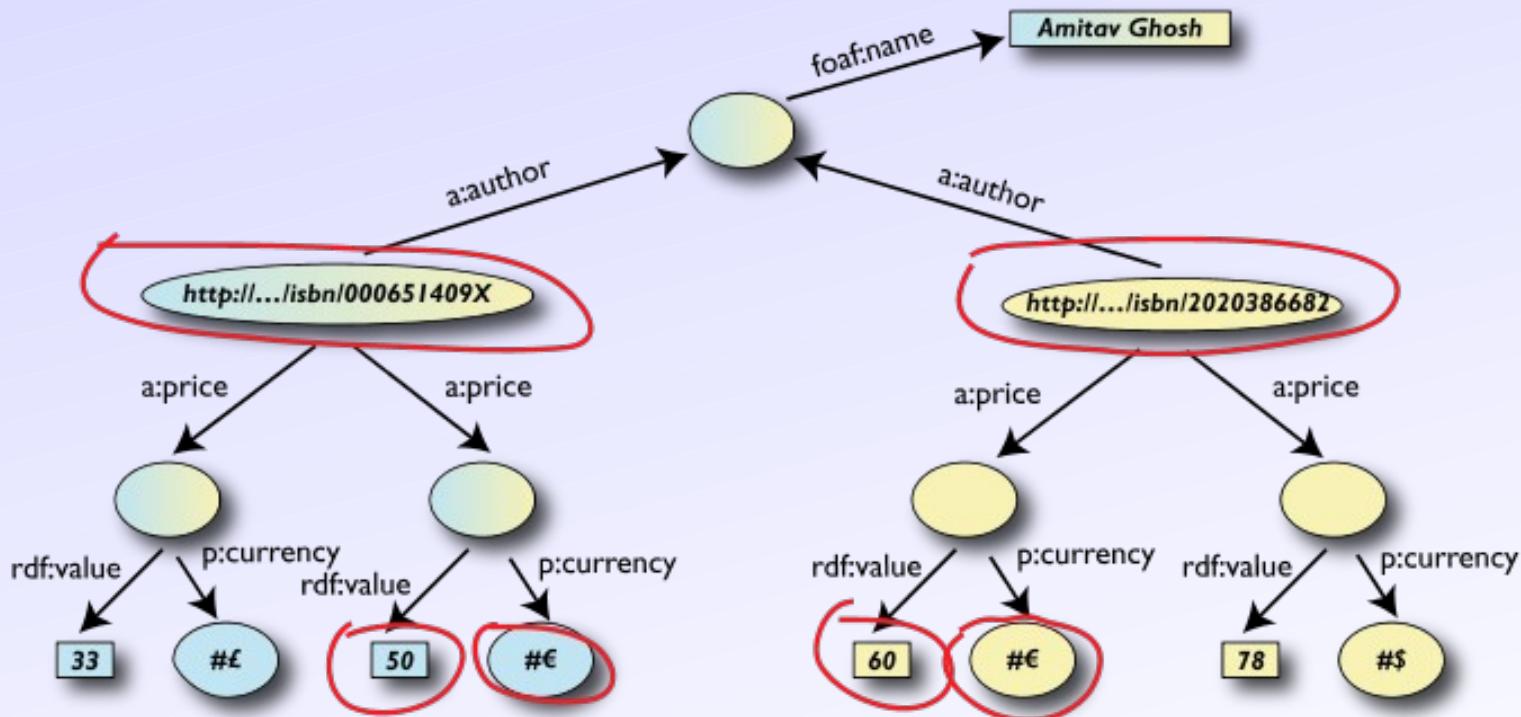
- Returns:
[[<..49X>,33,£], [<..49X>,50,€], [<..6682>,60,€],
[<..6682>,78,\$]]



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,€]]



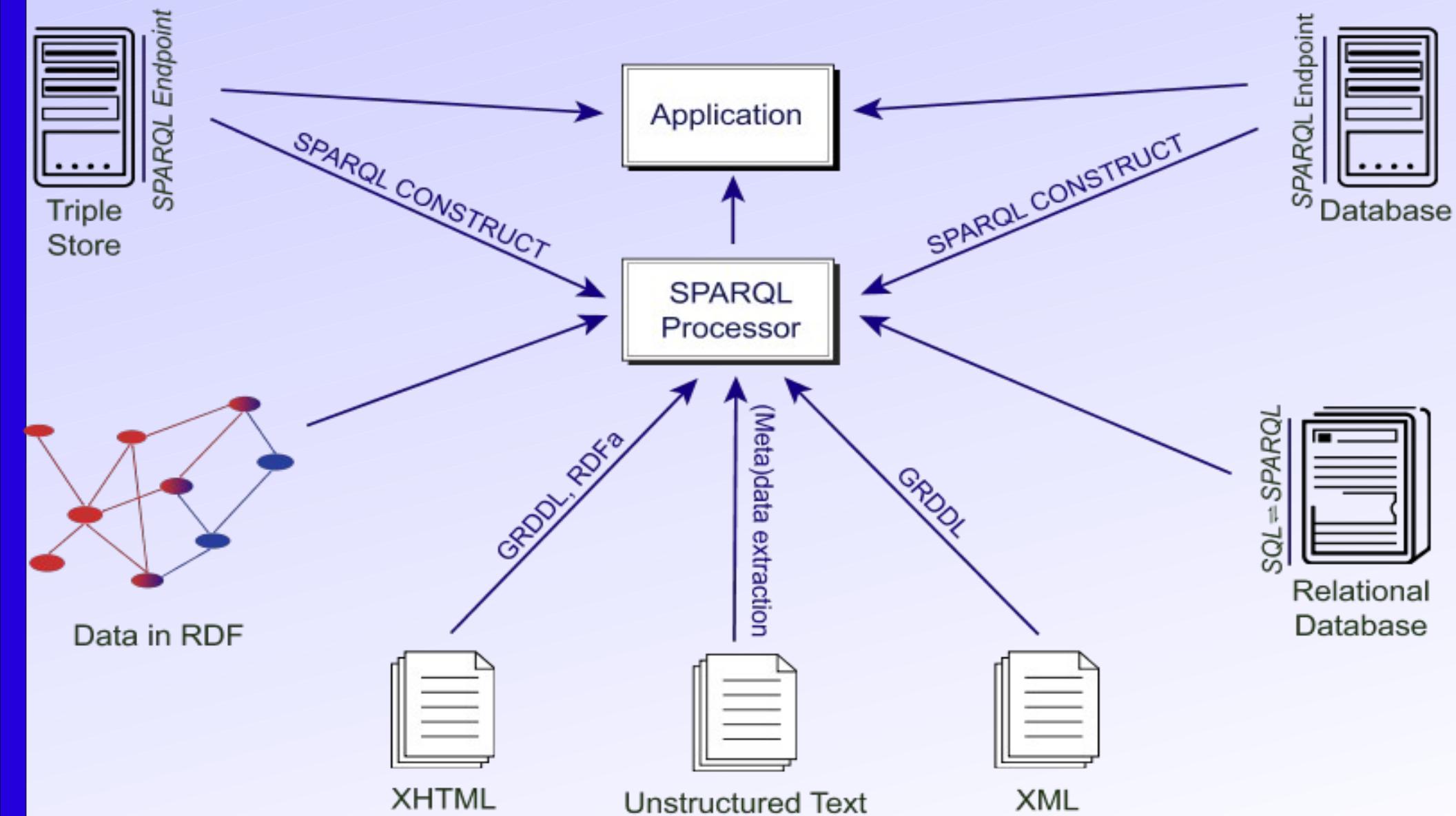
Other SPARQL features

- Limit the number of returned results; remove duplicates, sort them, ...
- Optional branches in the query
- Specify several data sources (via URI-s) within the query (essentially, a merge!)
- Construct a graph combining a separate pattern and the query results
- Use datatypes and/or language tags when matching a pattern

SPARQL usage in practice

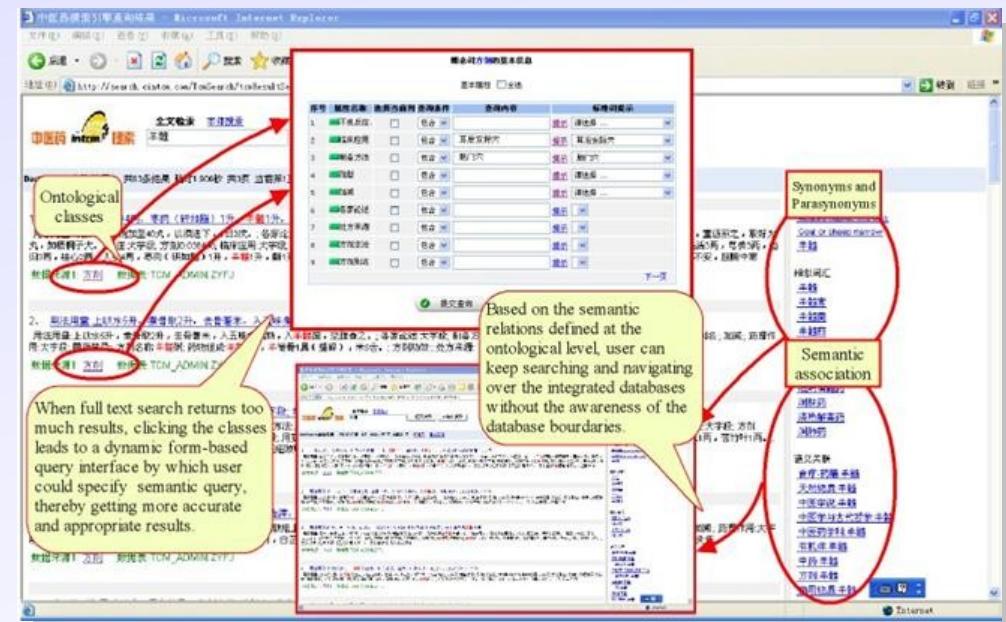
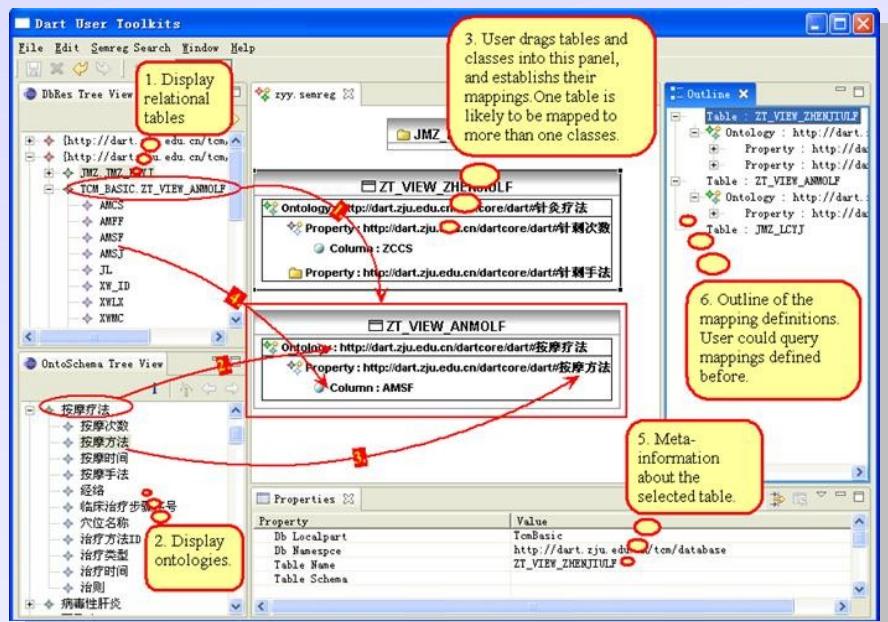
- SPARQL is usually used over the network
 - separate documents define the protocol and the result format
 - SPARQL Protocol for RDF with HTTP and SOAP bindings
 - SPARQL results in XML or JSON formats
- Big datasets usually offer “SPARQL endpoints” using this protocol
 - typical example: SPARQL endpoint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

SPARQL as a unifying point



Remember this example?

- The access to all the data is based on SPARQL queries



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
 - 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”

- 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
 - actually... there is a 2004 version of OWL (“OWL 1”)
 - and there is an update (“OWL 2”) to be published in 2009

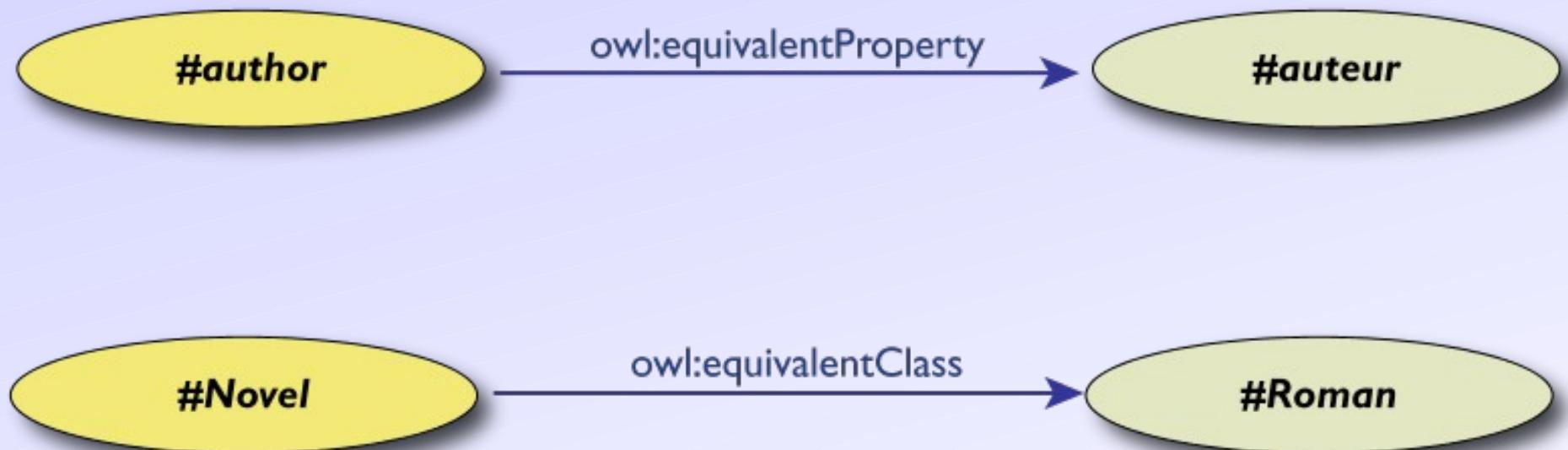
OWL is complex...

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

Term equivalences

- 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**
 - **owl:propertyDisjointWith**
- For individuals:
 - **owl:sameAs**: two URIs refer to the same concept (“individual”)
 - **owl:differentFrom**: negation of **owl:sameAs**

Connecting to French...



Typical usage of owl:sameAs

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

```
<http://dbpedia.org/resource/Amsterdam>
owl:sameAs <http://sws.geonames.org/2759793>;
```

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

Property characterization

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

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, processed through OWL, the following holds, too:

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

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

Property chains (OWL 2)

- Properties, when applied one after the other, may be subsumed by yet another one:
 - “if a person «P» was born in city «A» and «A» is in country «B» then «P» was born in country «B»”
 - more formally:

```
ex:born_in_country owl:propertyChainAxiom  
    (ex:born_in_city ex:city_in_country) .
```

- More than two constituents can be used
- There are some restrictions to avoid “circular” specifications

Keys (OWL 2)

- Inverse functional properties are important for identification of individuals
 - think of the email examples
- But... identification based on *one* property may not be enough

Keys (OWL 2)

“if two persons have the same emails and the same homepages then they are identical”

- Identification is based on the identical values of *two* properties
- The rule applies to persons only

Previous rule in OWL 2

```
:Person rdf:type owl:Class;  
owl:hasKey (:email :homepage) .
```

What it means is...

If:

```
<A> rdf:type :Person ;
  :email    "mailto:a@b.c" ;
  :homepage "http://www.ex.org".
```

```
<B> rdf:type :Person ;
  :email    "mailto:a@b.c" ;
  :homepage "http://www.ex.org".
```

then, processed through OWL 2, the following holds, too:

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

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

Classes in OWL (cont)

- OWL makes a stronger conceptual distinction between classes and individuals
 - there is a separate term for `owl:Class`, to make the difference (a specialization of the RDFS class)
 - individuals are separated into a special class called `owl:Thing`
- Eg, a precise classification would be:

```
ex:Person rdf:type owl:Class.
```

```
<uri-for-Amitav-Ghosh>
  rdf:type owl:Thing;
  rdf:type owl:Person .
```

Classes contents can be enumerated

```
:£ rdf:type owl:Thing.  
:€ rdf:type owl:Thing.  
:¥ rdf:type owl:Thing.  
:Currency  
    rdf:type owl:Class;  
    owl:oneOf (:€ :£ :¥).
```

- I.e., the class consists of exactly of those individuals

Union of classes can be defined

```
:Novel          rdf:type owl:Class.  
:Short_Story   rdf:type owl:Class.  
:Poetry        rdf:type owl:Class.  
:Literature    rdf:type owl:Class;  
               owl:unionOf (:Novel :Short_Story :Poetry).
```

- Other possibilities: `complementOf`, `intersectionOf`, ...

For example...

If:

```
:Novel          rdf:type owl:Class.  
:Short_Story    rdf:type owl:Class.  
:Poetry         rdf:type owl:Class.  
:Literature     rdf:type owl:Class;  
               owl:unionOf (:Novel :Short_Story :Poetry).
```

```
<myWork> rdf:type :Novel .
```

then the following holds, too:

```
<myWork> rdf:type :Literature .
```

It can be a bit more complicated...

If:

```
:Novel          rdf:type owl:Class.  
:Short_Story   rdf:type owl:Class.  
:Poetry        rdf:type owl:Class.  
:Literature    rdf:type owlClass;  
               owl:unionOf (:Novel :Short_Story :Poetry).
```

```
fr:Roman owl:equivalentClass :Novel .
```

```
<myWork> rdf:type fr:Roman .
```

then, through the *combination* of different terms,
the following still holds:

```
<myWork> rdf:type :Literature .
```

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.
- Many inferred relationships can be found using a traditional rule engine

However... that may not be enough

- Very large vocabularies might require even more complex features
 - typical usage example: definition of all concepts in a health care environment
 - a major issue: 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 its individuals
- 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 be of one of those values...
 - ...thereby defining the class of “listed price”

Restrictions formally

- Defines a class of type **owl:Restriction** with a
 - reference to the property that is constrained
 - definition of the constraint itself
- One can, e.g., subclass from this node when defining a particular class

```
:Listed_Price rdfs:subClassOf [  
    rdf:type owl:Restriction;  
    owl:onProperty p:currency;  
    owl:allValuesFrom :Currency.  
].
```

Possible usage...

If:

```
:Listed_Price rdfs:subClassOf [  
    rdf:type owl:Restriction;  
    owl:onProperty p:currency;  
    owl:allValuesFrom :Currency.  
].  
  
:price rdf:type :Listed_Price .  
  
:price p:currency <something> .
```

then the following holds:

```
<something> rdf:type :Currency .
```

Other restrictions

- **allValuesFrom** could be replaced by:
 - **someValuesFrom**
 - e.g., I could have said: there should be a price given in at least one of those currencies
 - **hasValue**, when restricted to one specific value
- Cardinality restrictions: instead of looking at the values of properties, their number is considered
 - eg, a specific property should occur exactly once

Datatypes in OWL

- RDF Literals can have a datatypes, OWL adopts those
- But more complex vocabularies require datatypes “restrictions”; eg, numeric intervals
 - “I am interested in a price range between €5 and €15”
- RDF allows any URI to be used as datatypes
 - ie, one could use XML Schemas to define, eg, numeric intervals
 - but it is very complex, and reasoners would have to understand a whole different syntax

Datatype restrictions (OWL 2)

- For each datatype, XML Schema defines possible restriction “facets”: min and max for numeric types, length for strings, etc
- OWL uses these facets to define *datatype ranges* for its own use

Definition of a numeric interval in OWL 2

```
:AllowedPrice rdf:type rdfs:Datatype;
  owl:onDatatype xsd:float;
  owl:withRestriction (
    [ xsd:minInclusive 5.0 ]
    [ xsd:maxExclusive 15.0 ]
) .
```

- The possible facets depend on the datatype:
xsd:pattern, **xsd:length**, **xsd:maxLength**,
...

Typical usage of OWL 2 datatype restrictions

```
:Affordable_book rdf:type owl:Class;
  rdfs:subClassOf [
    rdf:type owl:Restriction;
    owl:onProperty p:price_value;
    owl:allValuesFrom [
      rdf:type rdfs:Datatype;
      owl:onDatatype xsd:float;
      owl:withRestriction (
        [ xsd:minInclusive 5.0 ]
        [ xsd:maxExclusive 15.0 ]
      )
    ]
  ].
```

ie: an affordable book has a price between 5.0 and 15.0

But: OWL is hard!

- The combination of class constructions with various restrictions is extremely powerful
- What we have so far follows the same logic as before
 - extend the basic RDF and RDFS possibilities with new features
 - define their semantics, ie, what they “mean” in terms of relationships
 - expect to infer new relationships based on those
- However... a full inference procedure is hard 🤯
 - not implementable with simple rule engines, for example

OWL “species”

- OWL species 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
- They reflect compromises: expressibility vs. implementability

Unrestricted OWL (a.k.a. “OWL Full”)

- No constraints on any of the constructs
 - `owl:Class` is just syntactic sugar for `rdfs:Class`
 - `owl:Thing` is equivalent to `rdfs:Resource`
 - 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, apply properties on them
 - etc
 - etc.
- Extension of RDFS in all respects
- But: no system may exist that infers everything one might expect

OWL Full usage

- Nevertheless OWL Full is essential
 - it gives a generic framework to express many things with precise semantics
 - some application actually just need to express and interchange terms (even with possible scruffiness)
- Applications may control what terms are used and how
 - in fact, they may define their own sub-language via, eg, a vocabulary
 - thereby ensuring a manageable inference procedure

OWL DL

- A number of restrictions are defined
 - classes, individuals, object and datatype properties, etc, are fairly strictly separated
 - object properties must be used with individuals
 - i.e., properties are really used to create relationships between individuals
 - no characterization of datatype properties
 - ...
- But: well known inference algorithms exist!

Examples for restrictions

- The following is not “legal” OWL DL:

```
<q> rdf:type <A>.           # A is a class, q is an individual  
  
<r> rdf:type <q>.           # error: q cannot be used for a class, too  
  
<A> ex:something <B>.       # error: properties are for individuals only  
  
<q> ex:something <s>.         # error: same property cannot be used as  
<p> ex:something "54".       #   object and datatype property
```

OWL DL usage

- Abiding to the restrictions means that very large ontologies can be developed that require precise procedures
 - eg, in the medical domain, biological research, energy industry, financial services (eg, XBRL), etc
 - the number of classes and properties described this way can go up to the many thousands
- OWL DL has become a language of choice to define and manage formal ontologies in general
 - even if their usage is not necessarily on the Web

OWL 2 defines further species a.k.a. “profiles”

- Further restrictions on how terms can be used and what inferences can be expected

OWL 2 profiles: EL

- Goal: classification and instance queries in polynomial time
- Suitable for
 - very large number of classes and/or properties
 - not require complex expressions
 - eg: SNOMED
- Some excluded features
 - no cardinality restrictions, fewer property restrictions
 - no inverse, reflexive, disjoint, symmetric, asymmetric, functional or inverse functional properties
 - class disjunction
 - ...

OWL 2 profiles: QL

- Goal: conjunctive queries on top of relational databases (essentially: query rewriting to SQL)
- Suitable for
 - lightweight ontologies, but large data
- Some excluded features
 - functional and inverse functional properties, sameAs, keys
 - fewer property restrictions
 - no cardinality restrictions
 - transitive properties, property chains
 - ...

OWL 2 profiles: RL

- Goal: polynomial reasoning on top of rule engines
- Suitable for
 - relatively lightweight ontologies, but large data
- Some excluded features
 - fewer property restrictions
 - fewer cardinality restrictions (at most 0/1)
 - constraints on class expressions (union, intersections, etc) when used in subclass expressions
 - no datatype restrictions
 - ...

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 import

Must I use large ontologies?

- NO!!!
- Many applications are possible with RDFS and a just a little bit of OWL
 - a few terms, whose meaning is defined in OWL, and that application can handle directly
 - OWL RL is a step to create such a generic OWL level
- Big ontologies can be expensive (both in time and money); use them only when really necessary!

Ontologies examples

- 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

Example: improved search via ontology

- Search results are re-ranked using ontologies
- Related terms are highlighted, usable for further search

The screenshot shows the GoPubMed interface in Mozilla Firefox. The search term 'tinnitus' is entered in the search bar, resulting in 1,000 articles. The left sidebar displays a tree view of 'Top categories' under 'what', with a red oval highlighting the 'Diseases [985]' section. The main content area lists four search results, each with a thumbnail, author information, PMID, title, and a brief abstract. A blue arrow points from the third result back to the 'Diseases' category in the sidebar.

Result Number	Title	Author(s)	PMID	Abstract
5	Pros and cons of tinnitus retraining therapy.	Hatanaka A et al., Acta Otolaryngol	18368566	A significant reduction in the tinnitus Handicap Inventory (THI) was obtained as early as 1 month after implementation of tinnitus retraining therapy (TRT).
1	Gabapentin effectiveness on the sensation of subjective idiopathic tinnitus : a pilot study.	Bakhshaei M et al., Eur Arch Otorhinolaryngol	17960408	Pure-tone audiograms, laboratory test and personal histories were used to exclude any particular etiology of tinnitus .
3	Algorithm for evaluation of pulsatile tinnitus.	Mattox DE et al., Acta Otolaryngol	18368578	Among patients with venous tinnitus , sigmoid sinus diverticulum was the most common finding.
4	Functional imaging of unilateral tinnitus using fMRI.	Lanting CP et al., Acta Otolaryngol	18368576	The response to sound in the inferior colliculus was elevated in tinnitus patients compared with controls without tinnitus .

Example: improved search via ontology

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

Screenshot of the Go3R - Mozilla Firefox interface showing search results for "tinnitus".

The search bar shows "tinnitus" and the result count is "1,000 articles".

The left sidebar displays a tree view of categories under "what", with several nodes circled in red:

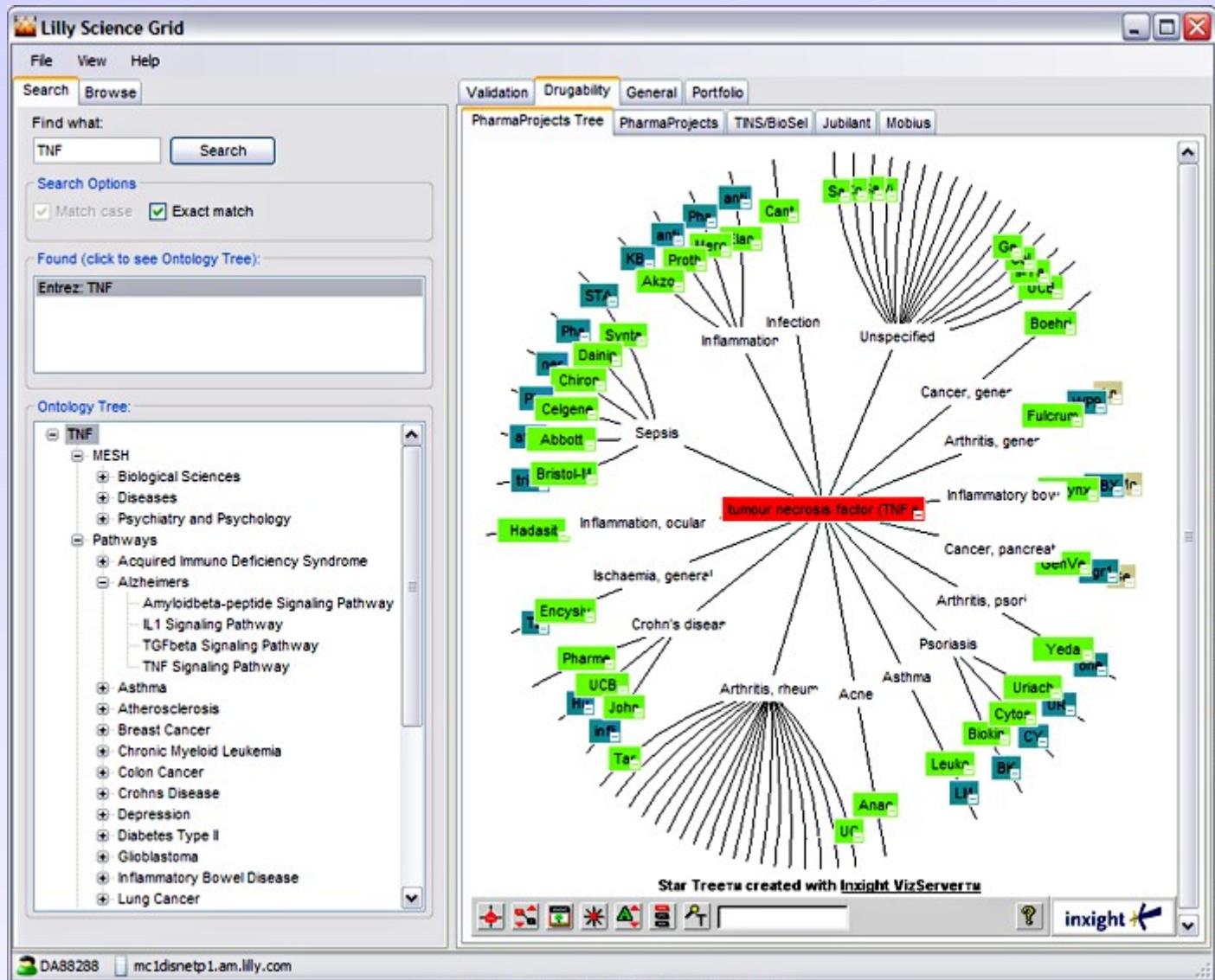
- 3R Relevance Filters (Beta)
- Top categories
 - Diseases & Symptoms [601]
 - Tinnitus [547]
 - Hearing Loss [248]
 - Vertigo [98]
 - Disease [118]
 - Hearing Loss, Sensorineural [95]
 - more
 - Methodology [408]
 - Life Sciences [503]
 - Body Systems & Structures [401]
 - Bioethics [102]
 - Reduction [90]
 - more
 - Statistics [125]
 - Substances, Preparations & Products [277]
 - Biological Material & Organisms for Animal U
 - Method Specification [36]
 - Animal Species [40]
 - Product Properties & Effects [62]
 - Product Testing & Assessment [20]
 - 3Rs Methods in the Life Sciences [6]
 - Animal Experiment [61]
 - 3Rs Relevant [5]
 - In Vitro Experimental Design [20]
 - In Vivo Experimental Design [5]
 - Animal Condition, Physiological or Psycholog
 - Animal Care & Handling [3]
 - Toxic Actions of Substances [7]
 - Unclassified [390]
 - Find related categories ...
 - My last 5 queries
 - Clipboard [0]

The main content area displays six search results:

- 2: Microvascular decompression of cochleovestibular nerve.**
Yap L et al., Eur Arch Otorhinolaryngol, 2008
This report provides a review of all the published studies on MVD of the eighth (8th) nerve in alleviating cochleovestibular symptoms and presents three additional patients who underwent MVD of the eighth nerve for **tinnitus** or vertigo.
- 3: Algorithm for evaluation of pulsatile tinnitus.**
Mattox DE et al., Acta Otolaryngol, 128 (4): 427-31, 2008
Among patients with arterial **tinnitus**, carotid atherosclerotic disease was the most common.
- 4: Functional imaging of unilateral tinnitus using fMRI.**
Lanting CP et al., Acta Otolaryngol, 128 (4): 415-21, 2008
This article shows that the inferior colliculus plays a key role in unilateral subjective **tinnitus**.
- 5: Pros and cons of tinnitus retraining therapy.**
Hatanaka A et al., Acta Otolaryngol, 128 (4): 365-8, 2008
A significant reduction in the **Tinnitus** Handicap Inventory (THI) was obtained as early as 1 month after implementation of **tinnitus** retraining therapy (TRT).
- 6: Mass casualty incident management triage injury distribution of casualties and**

Eli Lilly's Target Assessment Tool

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



Help for deep sea drilling operations

- Integration of experience and data in the planning of deep sea drilling processes
- Discover relevant experiences
 - uses an ontology backed search engine

The screenshot shows a search interface for 'leak in barrier elements' using the AKSIO search engine. The interface includes search filters for discipline, operation, equipment, state, keywords_ref, wellbore_id_ref, and field_id. The results section displays 7 items, each with a title, date, description, and annotation status (55%, 50%, 47%, 39%).

Result ID	Title	Date	Description	Annotation (%)
1.	Top plug/20" EHSV	2002-06-26T10:00:00Z	Description: Experience: In a "standard" OPR design, the upper cement plug would cover the 13 3/8" cut as well as... EXPLORATION NO 6406/1-1 PA PLUGBACK/KICK-OFF Cementing Network Sementeringsnettverk Directional Drilling Network Directional Drilling Network Brønnintegritet Well Integrity Casing Faringor Deep set tubing plug Deep set tubing plug Liner top packer Liner top packer Mechanical tubular plugs Well Integrity Problem Well Integrity Problem	55% annotate comment
2.	RTH with drill stem teststring	2002-06-13T10:00:00Z	Description: RTH with drill stem teststring. Took weight when entering 7" liner with test string. Worked same pas... RIMFAKS NO 34/10-3-4 H DST DRILL STEM TEST Bronnintegritet Well Integrity Snubbing Seabbling Completion string component Completion string component Downhole tester valve Downhole tester valve Borestreng Drillstring Subsea production tree Subsea test tree Subsea test tree Surface test tree Surface test tree Well test packer Well test string Well test string Well test string components Erosion Erosion Lack Of Maintenance Lack Of Maintenance Leak in barrier elements Leak in barrier elements Well Integrity Problem	50% annotate comment
3.	Flowing well	2003-02-20T11:00:00Z	Description: The well was temporary handed back to production during changeover from slick line to 5 1/2" cable to... HEIDRUN NO 6507/7-A-20 WIREL Holi Trond OTHER Snubbing safety head Snubbing safety head USD none return valve USD none return valve Corrosion Corrosion Erosion Erosion Lack Of Maintenance Lack Of Maintenance Leak in barrier elements Leak in barrier elements Scale Deposition Scale Deposition Too High Mud Density Too high mud density Well Integrity Problem Well Integrity Problem	47% annotate comment
4.	Fill drop sub assy prior to making up packer for barrier assy to avoid possible trapped pressure	2002-06-24T10:00:00Z	Description: Fill drop sub assy prior to making up packer for barrier assy to avoid possible trapped pressure... HULDRA NO 30/2-A-6 8 1/2" Rodvett Knut T/A PLUGS & MECH. PLUGS Cementing Network Sementeringsnettverk Brønnintegritet Well Integrity Deep set tubing plug Deep set tubing plug Leak in barrier elements Leak in barrier elements Scale Deposition Scale Deposition Well Integrity Problem Well Integrity Problem	39% annotate comment

Rules (RIF)

Rules

- There is a long history of rule languages and rule-based systems
 - eg: logic programming (Prolog), production rules
- Lots of small and large rule systems (from mail filters to expert systems)
- Hundreds of niche markets

Why rules on the Semantic Web?

- There are conditions that ontologies (ie, OWL) cannot express
 - a well known example is Horn rules: $(P_1 \wedge P_2 \wedge \dots) \rightarrow C$
 - (though OWL 2 property chains cover some cases)
- A different way of thinking — people may feel more familiar in one or the other

Things you may want to express

- An example from our bookshop integration:
 - “a novel with over 500 pages and costing less than €5 is a cheap book”
 - something like (in an ad-hoc syntax):

```
If { ?x rdf:type p:Novel;
      p:page_number ?p;
      p:price [
          p:currency p:€;
          rdf:value ?z
      ].
      ?p > "500"^^xsd:integer.
      ?z < "5.0"^^xsd:double. }
then { ?x rdf:type p:CheapBook }
```

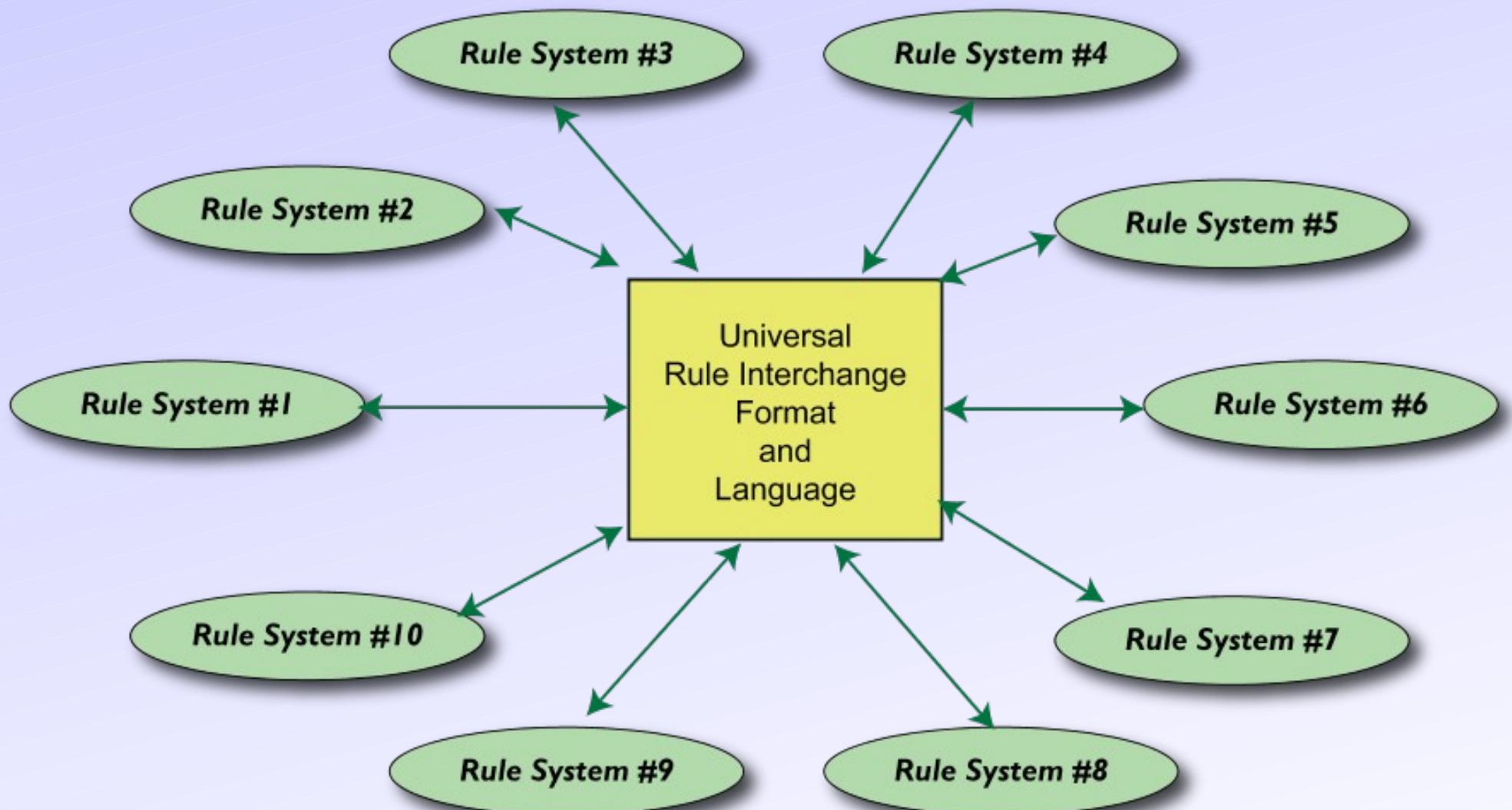
A new requirement: exchange of rules

- Applications may want to exchange their rules:
 - negotiate eBusiness contracts across platforms: supply vendor-neutral representation of your business rules so that others may find you
 - describe privacy requirements and policies, and let clients “merge” those (e.g., when paying with a credit card)
- Hence the name of the working group: Rule Interchange Format
 - goal is a language that
 - expresses the rules a bit like a rule language
 - can be used to exchange rules among engines

Notes on RIF (cont)

- RIF does not concentrate on RDF only
 - ie, certain constructions go beyond what RDF can express
- But there is a “subset” that is RDF and also OWL related
- For the coming few slides, forget about RDF 😊
 - we will come back to it. Promise!

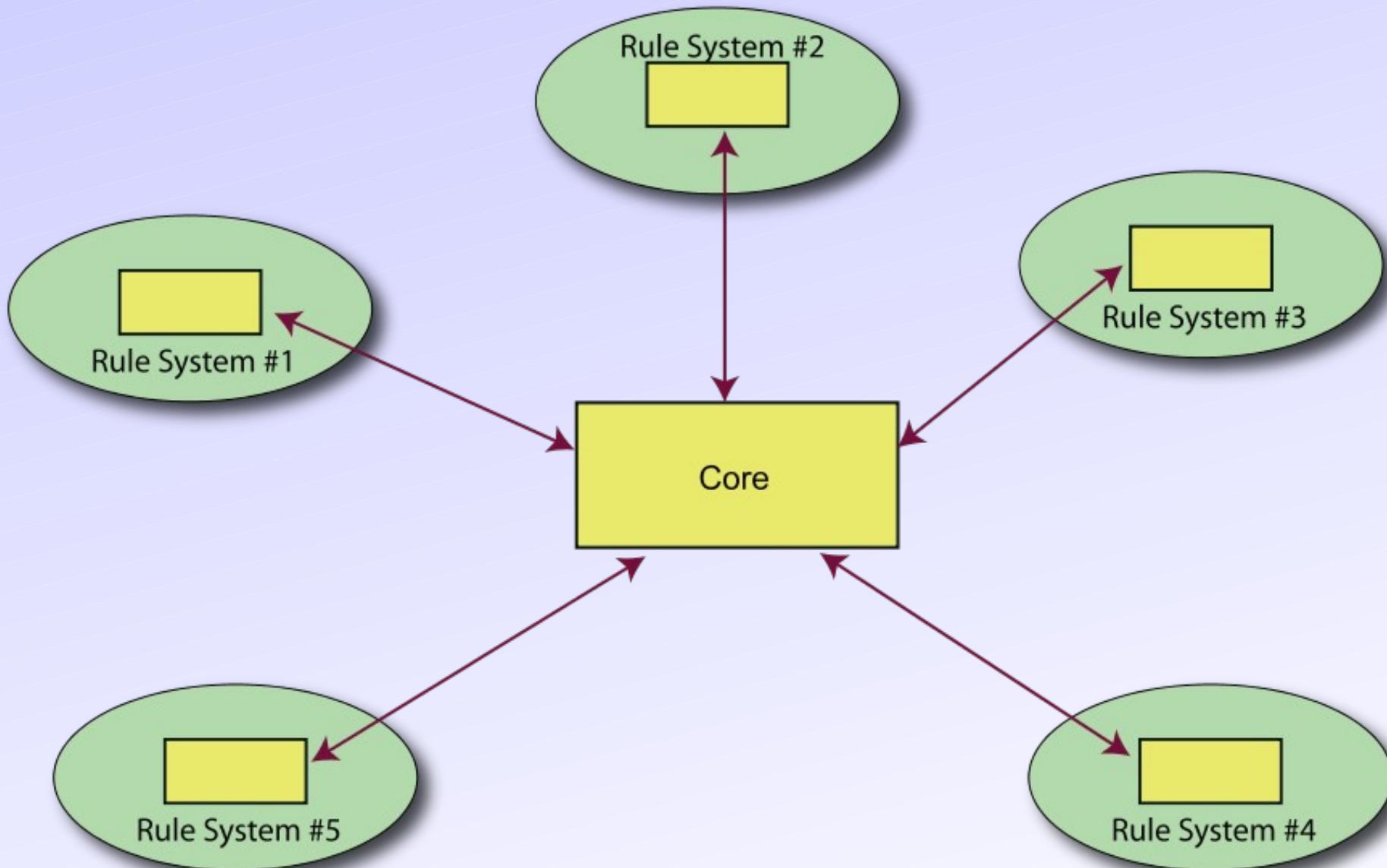
In an ideal World



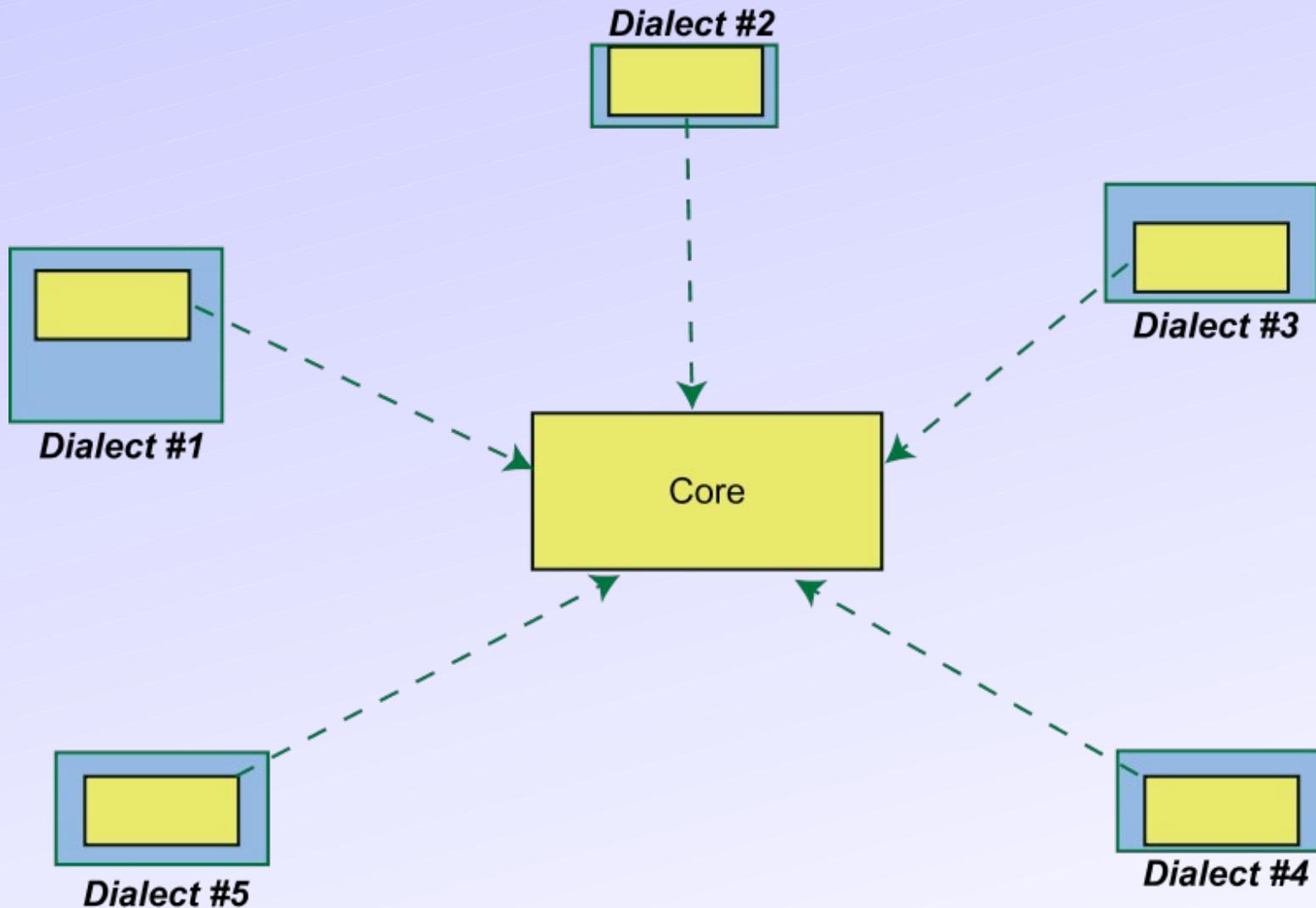
In the real World...

- Rule based systems can be very different
 - different rule semantics (based on various type of model theories, on proof systems, etc)
 - production rule systems, with procedural references, state transitions, etc
- Such universal exchange format is not feasible
- The idea is to define “cores” for a family of languages with “variants”

RIF “core”: only partial interchange

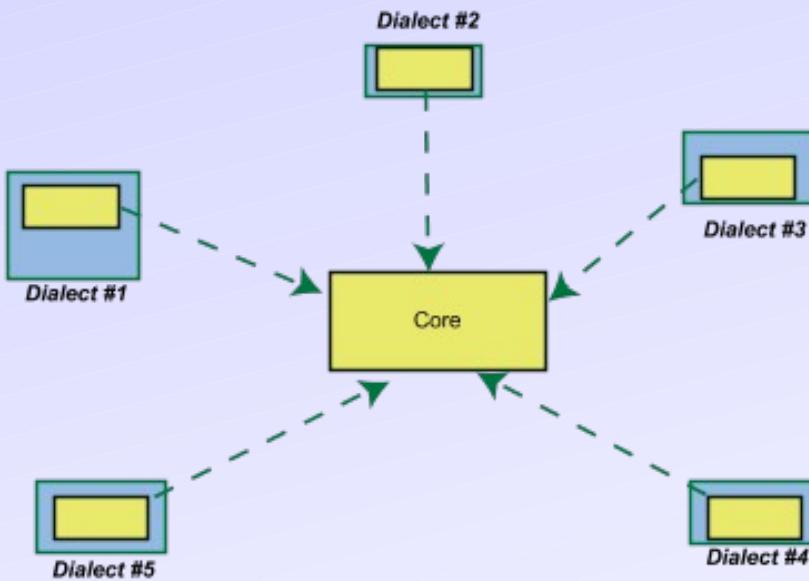


RIF “dialects”

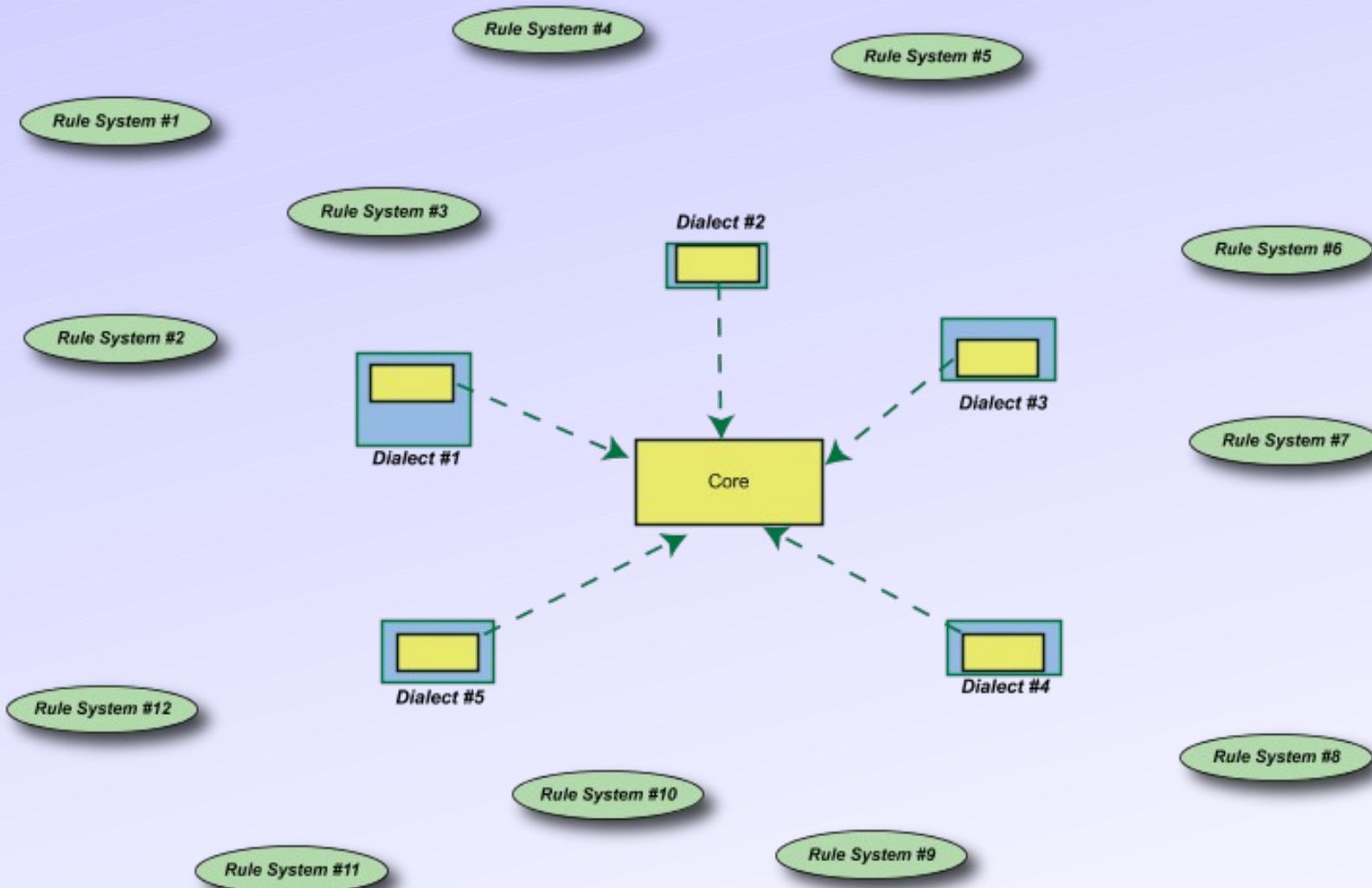


- Possible dialects: F-logic, production rules, fuzzy or probabilistic logic, ...

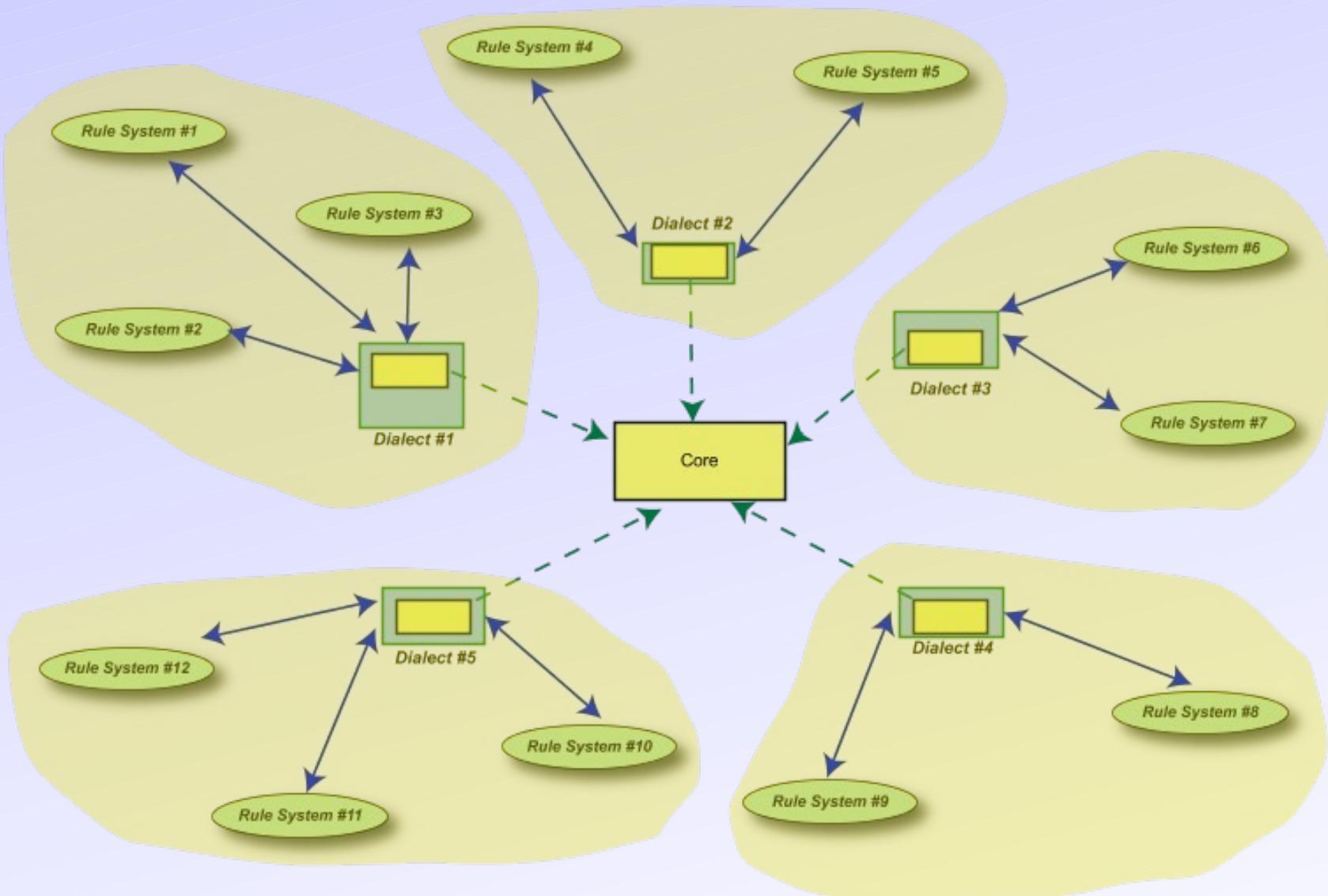
Role of dialects



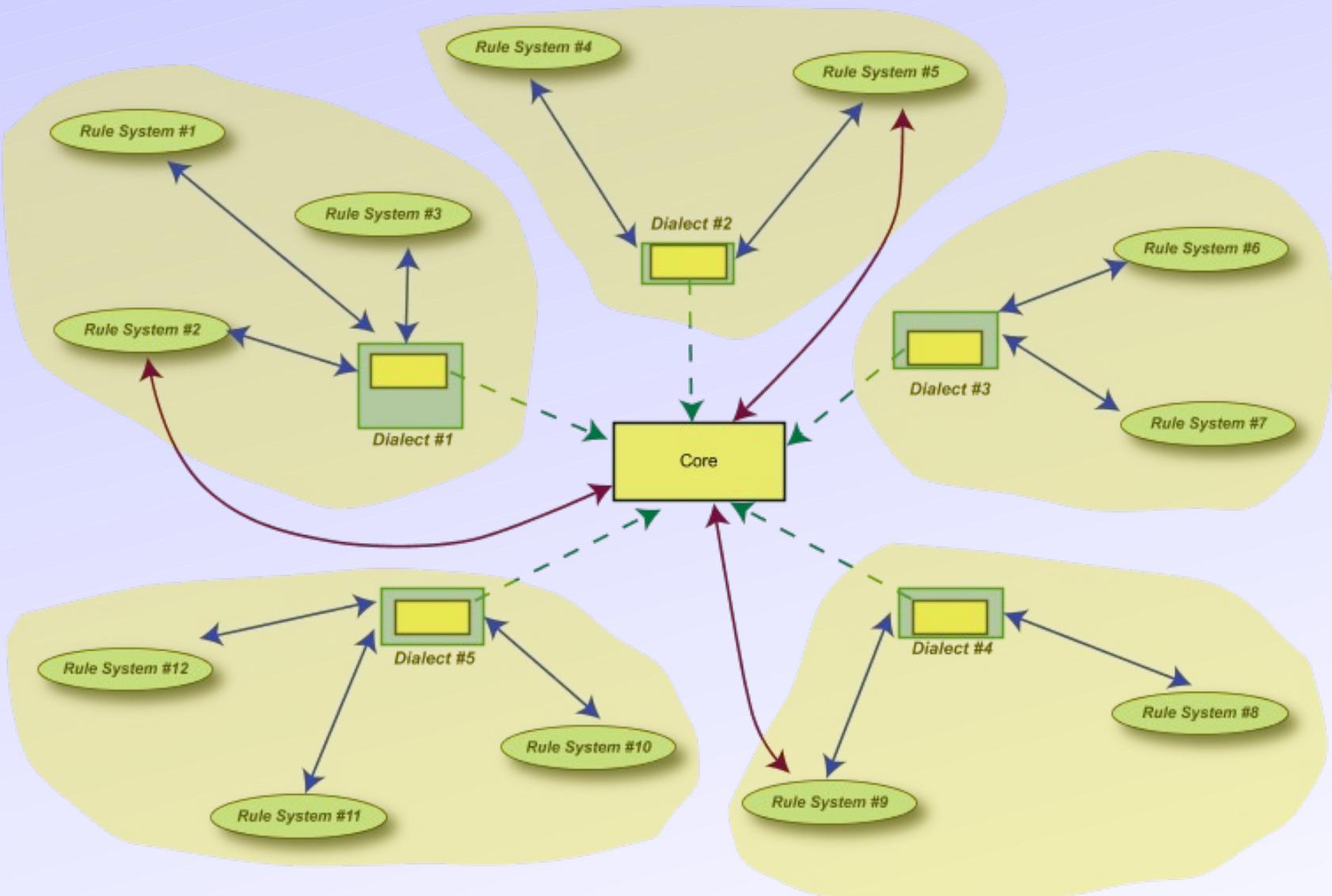
Role of dialects



Role of dialects



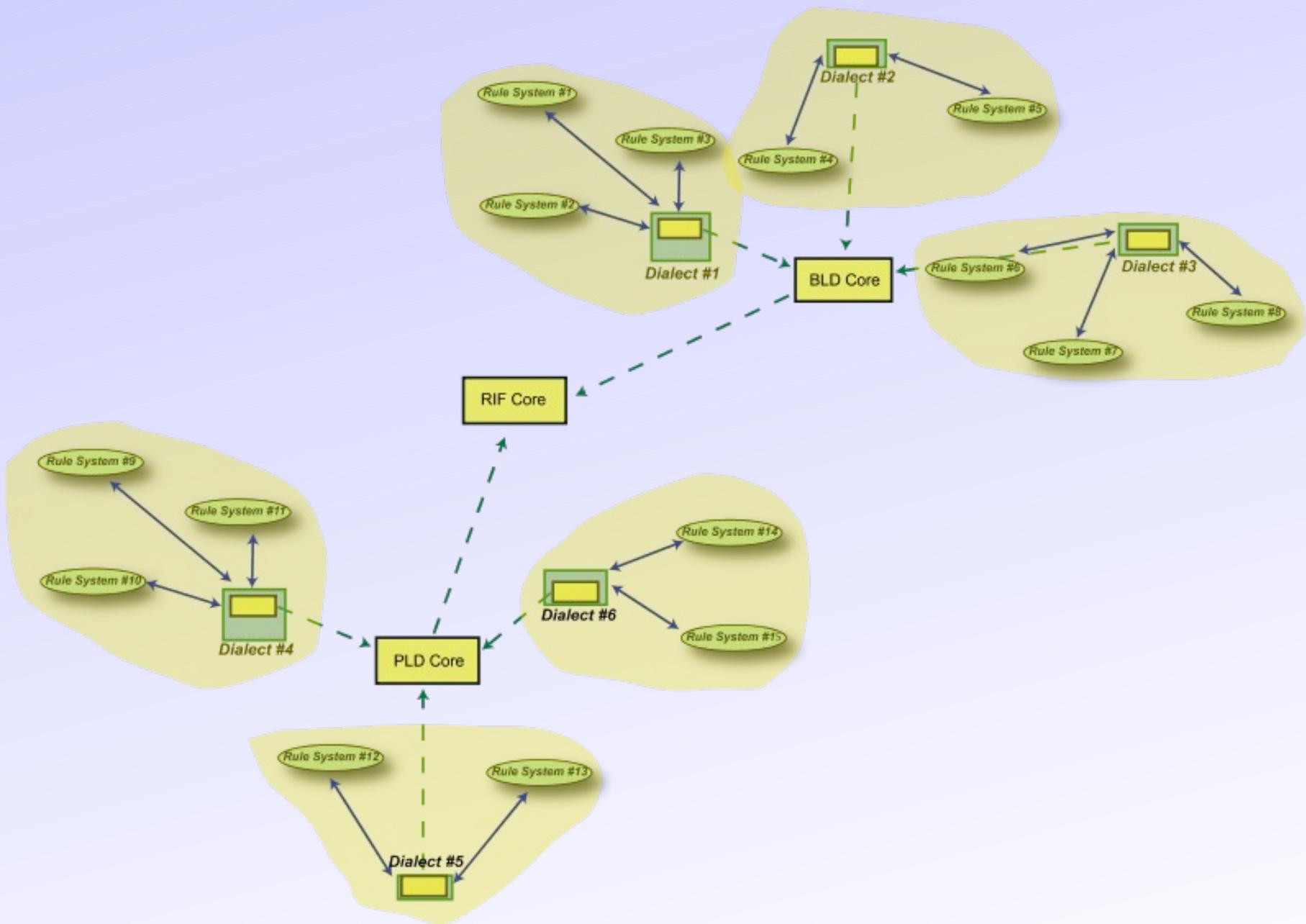
Role of dialects



However...

- Even this model does not completely work
- The gap between production rules and “traditional” logic systems is too large
- A hierarchy of cores is necessary:
 - a *Basic Logic Dialect* and *Production Rule Dialect* as “cores” for families of languages
 - a common *RIF Core* binding these two

Hierarchy of cores



Current status

- Candidate Recommendation published in October 2009
 - what this means: technical work is done, cross-checked against implementations

RIF Core

- Core defines
 - a “presentation syntax”, which is really to... present the constructions (is not necessarily implemented in tools)
 - a formal XML syntax to encode and exchange the rules
- A Core document is
 - some directives like import, prefix settings for URI-s, etc
 - a sequence of implications, possibly involving built-in predicates on datatypes

RIF Core example

```
Document(  
  Prefix(cpt http://example.com/concepts#)  
  Prefix(ppl http://example.com/people#)  
  Prefix(bks http://example.com/books#)  
  
  Group  
  (  
    Forall ?Buyer ?Item ?Seller (  
      cpt:buy(?Buyer ?Item ?Seller) :- cpt:sell(?Seller ?Item ?Buyer)  
    )  
    cpt:sell(ppl:John bks:LeRif ppl:Mary)  
  )  
)
```

infers the following relationship:

```
cpt:buy(ppl:Mary bks:LeRif ppl:John)
```

Additional RIF Core features

- RIF Core includes some extra features
 - built-in datatypes and predicates
 - notion of “local names”, a bit like RDF’s blank nodes
 - “classification”, like typing in RDFS and OWL
 - $p \# T$

What about RDF(S), OWL, and RIF?

- Typical scenario: applications exchange rules that refer to RDF data
- To make that work:
 - RDF facts/triples have to be representable in Core
 - harmonization on the concepts is necessary
 - the formal semantics of the two worlds should also be aligned
- There is a separate document that brings these together

Rules vs OWL?

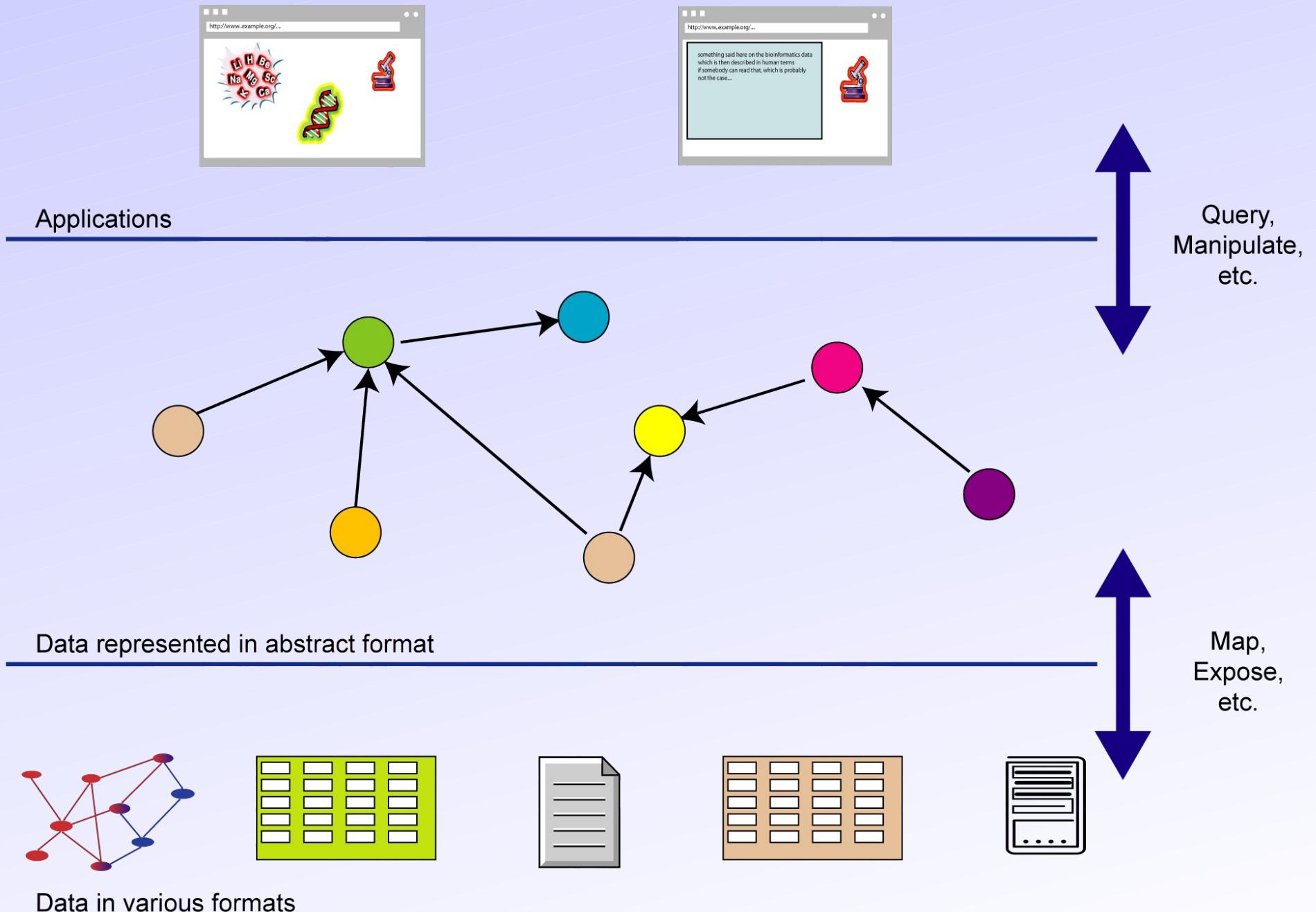
- In a SW application, should I use RIF, OWL, or both?
- The two approaches are complimentary
 - there are things that rules cannot really express or infer
 - eg, inferencing complex relationships among classes
 - there are things that ontologies cannot really express or in only a very complicated manner
 - eg, complex Horn rules
- Often, applications require both

What have we achieved? (putting all this together)

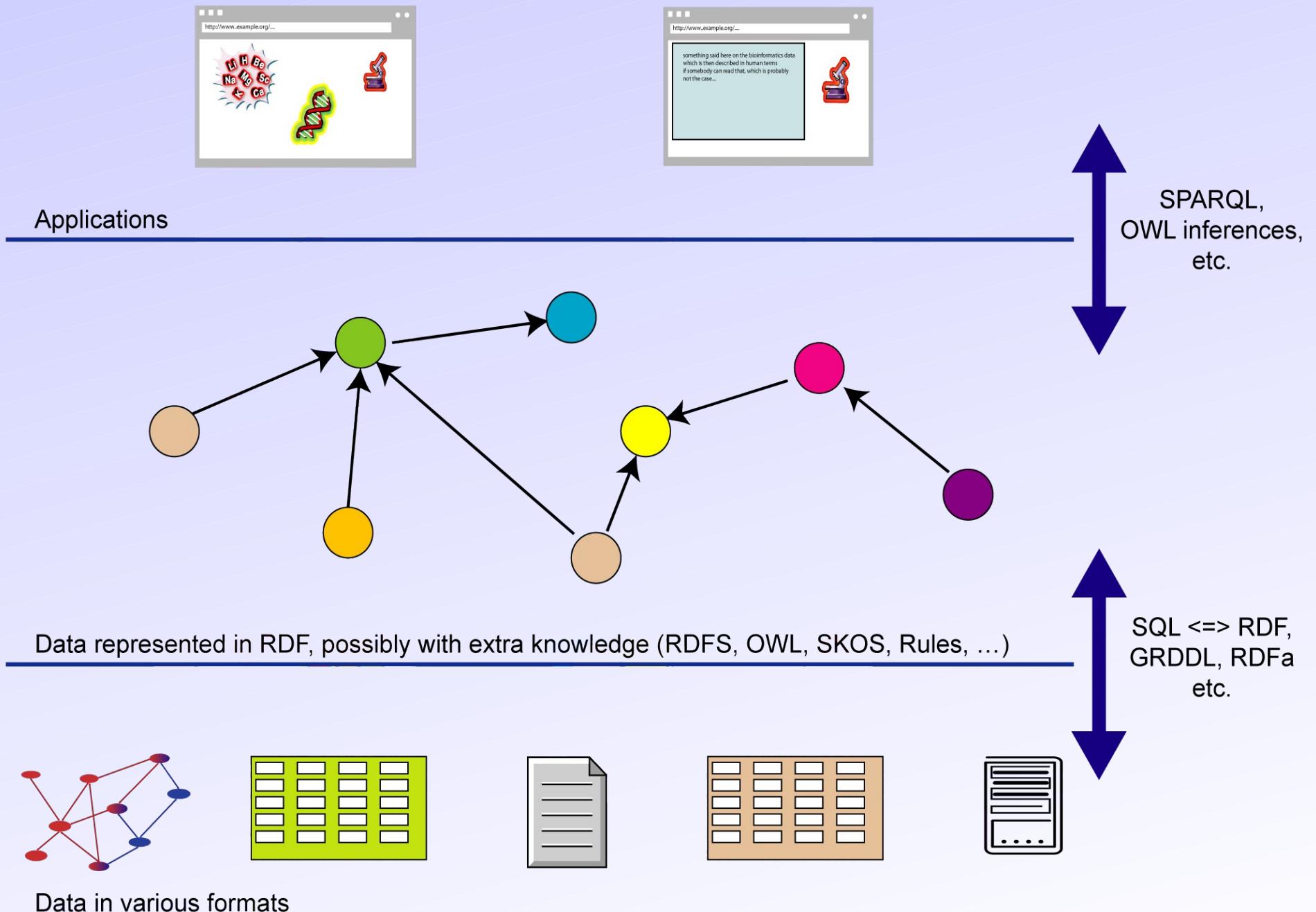
Other SW technologies

- There are other technologies that we do not have time for here
 - find RDF data associated with general URI-s: POWDER
 - bridge to thesauri, glossaries, etc: SKOS

Remember the integration example?



Same with what we learned



Example: personalized tourist itinerary

The screenshot shows a personalized tourist itinerary for Zaragoza on June 17, 2008. The itinerary is divided into Morning and Afternoon sessions. The Morning session includes visits to the Basilica of the Pilar, Ibercaja Camón Aznar Museum, Cathedral of San Salvador or La Seo, Caesaraugusta Forum Museum, Caesaraugusta River Port Museum, and Molins house. The Afternoon session includes visits to various churches and a central market. A map of Zaragoza shows the locations of these sites with numbered pins. A detailed description of the Basilica of the Pilar is provided, mentioning its history and accessibility.

Integration of relevant data in Zaragoza (using RDF and ontologies)

Use rules on the RDF data to provide a proper itinerary

Available documents, resources

Available specifications: Primers, Guides

- The “RDF Primer” or “OWL 2 Primer” give a formal introduction to RDF(S) and OWL
- GRDDL and RDFa Primers have also been published
- The W3C Semantic Web Activity Homepage has links to all the specifications and guides:
 - <http://www.w3.org/2001/sw/>

“Core” vocabularies

- There are also a number widely used “core vocabularies”
 - 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

- J. Pollock: Semantic Web for Dummies, 2009
- 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
- P. Hitzler, R. Sebastian, M. Krötzsch: Foundation of Semantic Web Technologies, 2009
- ...

See the separate Wiki page collecting book references:
<http://esw.w3.org/topic/SwBooks>

Further information and Fora

- Planet RDF aggregates a number of SW blogs:
 - <http://planetrdf.com/>
- 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:
 - <http://www.w3.org/2001/sw/interest/>
 - there are also similar list for Linked Open Data, OWL developers, etc
 - contact me for details if you cannot find them

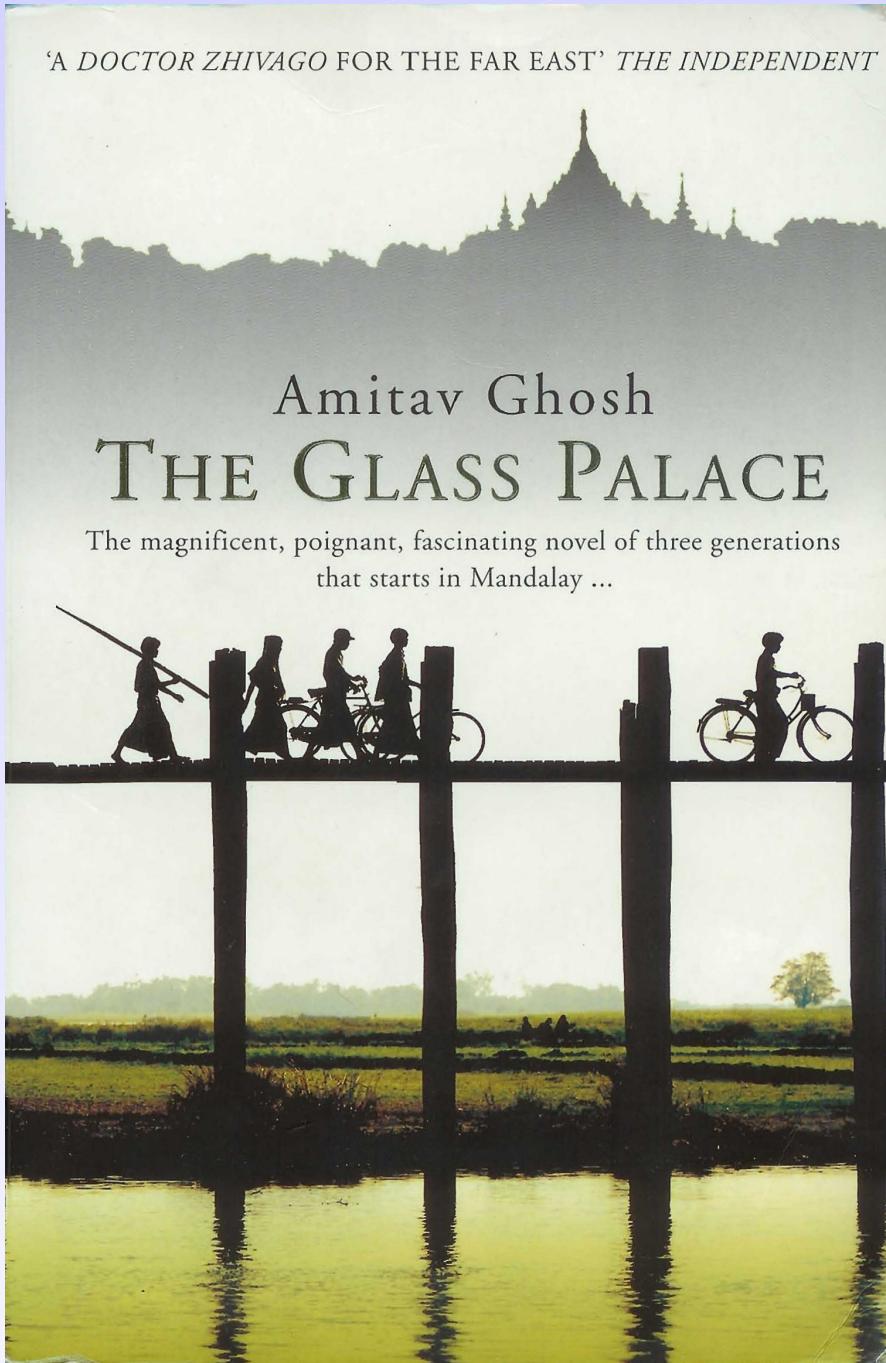
Lots of Tools (not an exhaustive list!)

- 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...
 - ...

Conclusions

- The Semantic Web is about creating a Web of Data
- There is a great and very active user and developer community, with new applications

By the way: the book is real 😊



Thank you for your attention!

These slides are also available on the Web:

<http://www.w3.org/2009/Talks/1030-Philadelphia-IH/>

