Tutorial on Semantic Web

Ivan Herman, W3C

Last update: 2012-04-09

WARNING TO THE READER!

- This is an evolving slide set. This means:
  - it changes frequently
  - there may be bugs, inconsistencies
  - it may try to reflect the latest view of technology evolution but that is often a moving target
- “Frozen” versions are instantiated for a specific presentation, and those become stable

Introduction

Eric Clapton

Born 30 March 1945.

MOST PLAYED ON BBC RADIO 2

Biography

Eric Patrick Clapton, CBE (born 30 March 1945) is an English blues-rock guitarist, singer, songwriter and composer. Clapton has been inducted into the Rock and Roll Hall of Fame as a solo performer, as a member of rock bands; the Yardbirds and Cream. Clapton is the only person ever to be inducted three times. Often viewed by critics and fans alike as one of the most important and influential guitarists of all time, Clapton was ranked fourth in Rolling Stone magazine's list of the "100 Greatest Guitarists of All Time" and #53 on their list of the Immortals: 100 Greatest Artists of All Time

Latest Tracks Played On The BBC

Promises
BBC Radio 2 | Ken Bruce 22/02/2010

Bad Love
BBC Radio 2 | Alex Lester 22/02/2010

Lay Down Sally
BBC Radio 2 | Chris Evans Breakfast 18/02/2010

I Ain't Gonna Stand For It
BBC Radio 2 | Alex Lester 15/02/2010

Wonderful Tonight
BBC Radio 2 | Ken Bruce 10/02/2010

Audio Previews From Latest Album Review

Me And Mr Johnson

8 Milkcow's Calf Blues

10 Come on in My Kitchen

this focus, he is credited as an innovator in a wide variety of genres. These include blues-rock (with John Mayall & the Bluesbreakers and The Yardbirds) and psychedelic rock (with Cream). Clapton's chart success was not limited to the blues, with chart-toppers in Delta Blues (Me and Mr. Johnson), pop ("Change the World") and reggae (Bob Marley's "I Shot the Sheriff") (He is often credited for bringing reggae and Bob Marley to the mainstream.) Two of his most successful recordings were the hit love song "Layla", which he played with the band Derek and the Dominos, and Robert Johnson's "Crossroads", which has been his staple song since his days with Cream.

Read more at Wikipedia...

Links & Information

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

MEMBER OF Derek and the Dominos, Blind Faith (1968-1969), Cream (1966-1968), John Mayall & The Bluesbreakers (1965-1966), The Yardbirds (1963-1965)

COLLABORATED ON J.J. Cale & Eric Clapton, Eric Clapton & The Immediate All Stars, Eric Clapton & The Impressions, Eric Clapton & Jimmy Page, Eric Clapton & David Sanborn, Eric Clapton & Stan Webb's Chicken Shack, Eric Clapton & The Powerhouse, Eric Clapton & Stevie Ray Vaughan, Eric Clapton & Marc Shaiman, The Dirty Mac, Bob Dylan, Roger McGuinn, Tom Petty, Neil Young, Eric Clapton & George Harrison, Jimmie Vaughan, Eric Clapton, Bonnie Raitt, Robert Cray, B.B. King, Buddy Guy, Dr. John & Art Neville, Elton John & Eric Clapton, Michael Kamen, Eric Clapton and David Sanborn, B.B. King & Eric Clapton, Mark Knopfler & Eric Clapton, Paul McCartney & Eric Clapton, Sting with Eric Clapton, Steve Winwood & Eric Clapton

Links & information come from MusicBrainz. You can add or edit information about Eric Clapton at musicbrainz.org. Find out more about our use of this data. The BBC is not responsible for the content of external sites

Latest News Stories

Played By

Since December 2008

Alex Lester
BBC Radio 2

Steve Wright in the Afternoon
BBC Radio 2

Sarah Kennedy
BBC Radio 2

Wake Up to Wogan
BBC Radio 2

Ken Bruce
BBC Radio 2

Steve Wright's Sunday Love Songs
BBC Radio 2

Jeremy Vine
BBC Radio 2

Information displayed about artists played on BBC programmes is incomplete
out more about this artist play count information.

How to build such a site 1.

- Site editors roam the Web for new facts
  - may discover further links while roaming
- They update the site manually
- And the site gets soon out-of-date ☹

How to build such a site 2.

- Editors roam the Web for new data published on Web sites
- “Scrape” the sites with a program to extract the information
  - Ie, write some code to incorporate the new data
- Easily get out of date again… ☹

How to build such a site 3.

- Editors roam the Web for new data via API-s
- Understand those…
  - input, output arguments, datatypes used, etc
- Write some code to incorporate the new data
- Easily get out of date again… ☹

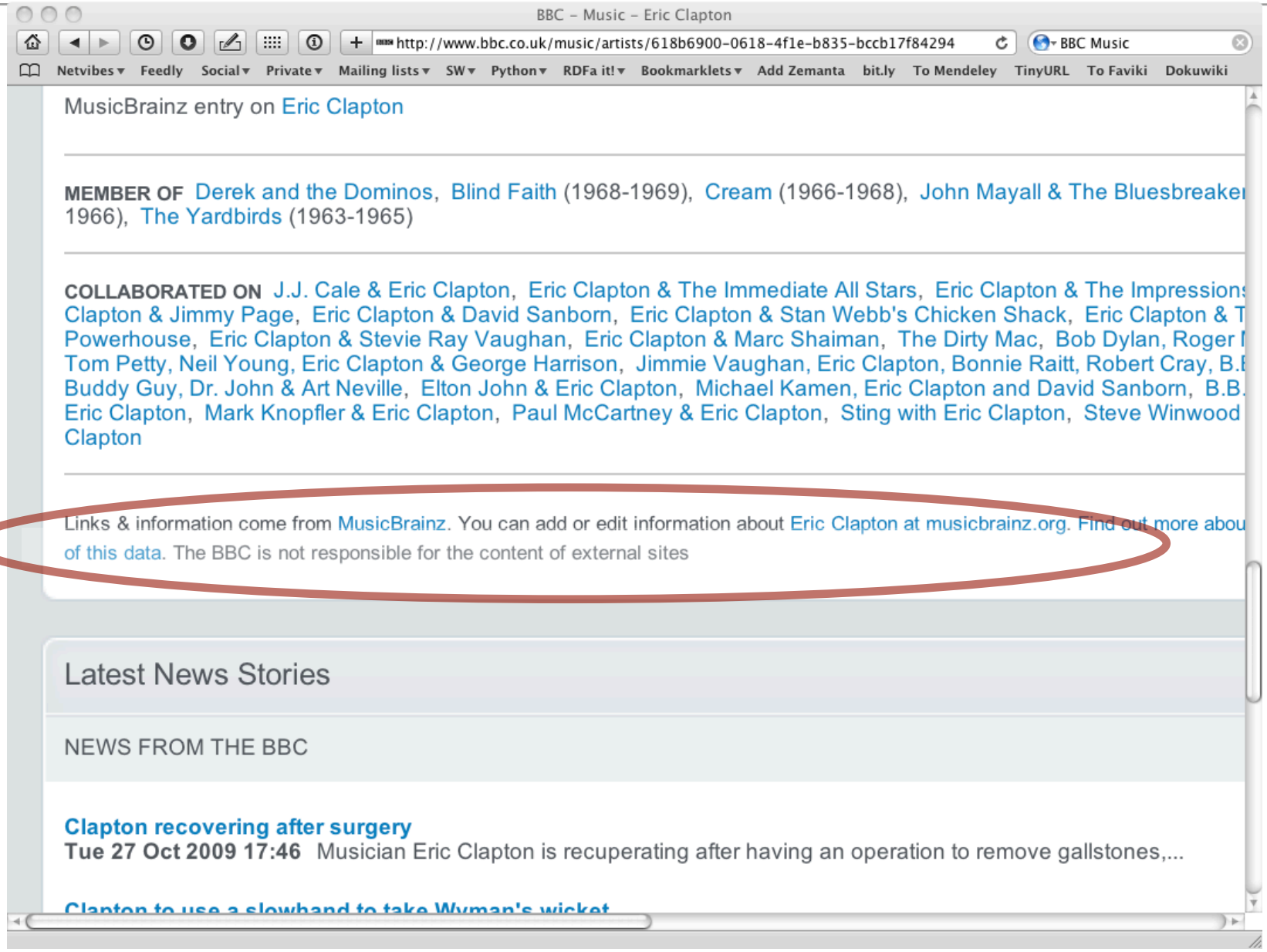
The choice of the BBC

- Use external, public datasets
  - Wikipedia, MusicBrainz, …
- They are available *as data*
  - not API-s or hidden on a Web site
  - data can be extracted using, e.g., HTTP requests or standard queries

In short…

- Use the Web of Data as a Content Management System
- Use the community at large as content editors

And this is no secret…



Data on the Web

- There are more an more data on the Web
  - government data, health related data, general knowledge, company information, flight information, restaurants,…
- More and more applications rely on the availability of that data

Photo credit “nepatterson”, Flickr

Imagine…

- A “Web” where
  - documents are available for download on the Internet
  - but there would be no hyperlinks among them

CoCoDat - Collation of Cortical Data - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.cocom

CoCoMac DATABASES ORT EXAMPLES

CoCoDat: Collation of Cortical [s
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 u
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/ind

Cell Centered Database - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://ccdb.ucsd.edu/sand/main?event=gallery&action=show&dpl=y

red database

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

y 23
D:
e 071202tjd
e:
D: P1188
ct Brain Maps
e:
ct Tom Deerinck
r:
nt BioRad RTS 2000MP
d: Multiphoton
n: montage of cerebellar cortex triple labeled for Hoescht stain (blue), IP3 receptor (green) and GFAP (red)

Z PIP logged out

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

File Edit View History Bookmarks Tools Help

http://senselab.med.yale.edu

senselab

NeuronDB

Back

Thalamic relay neuron

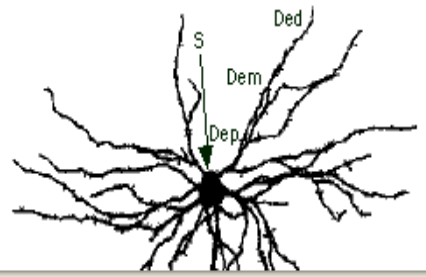
Mode: **Overview** | Data/Search | plus Connectivity | plus Classical References/Notes | Models

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

Z PIP logged out

Data on the Web is not enough…

- We need a proper infrastructure for a real *Web of Data*
  - data is available on the Web
    - accessible via standard Web technologies
  - data are interlinked over the Web
  - ie, data can be *integrated* over the Web
- This is where Semantic Web technologies come in

Photo credit "kxlly", Flickr

In what follows…

- We will use a simplistic example to introduce the main Semantic Web concepts

The rough structure of data integration

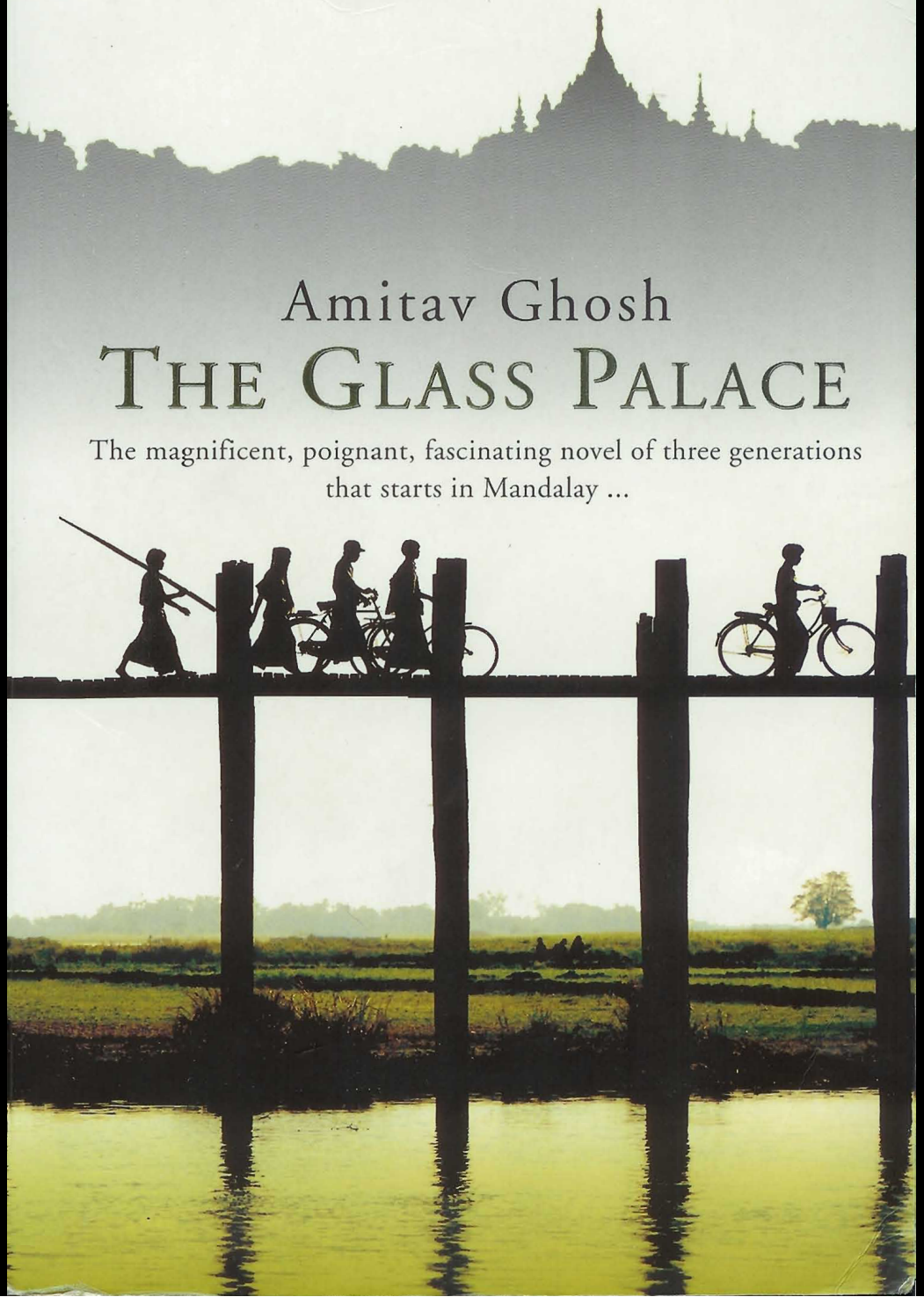
- Map the various data onto an abstract data representation
  - make the data independent of its internal representation…
- Merge the resulting representations
- Start making queries on the whole!
  - queries not possible on the individual data sets

'A *DOCTOR ZHIVAGO* FOR THE FAR EAST' *THE INDEPENDENT*

Amitav Ghosh

THE GLASS PALACE

The magnificent, poignant, fascinating novel of three generations that starts in Mandalay ...



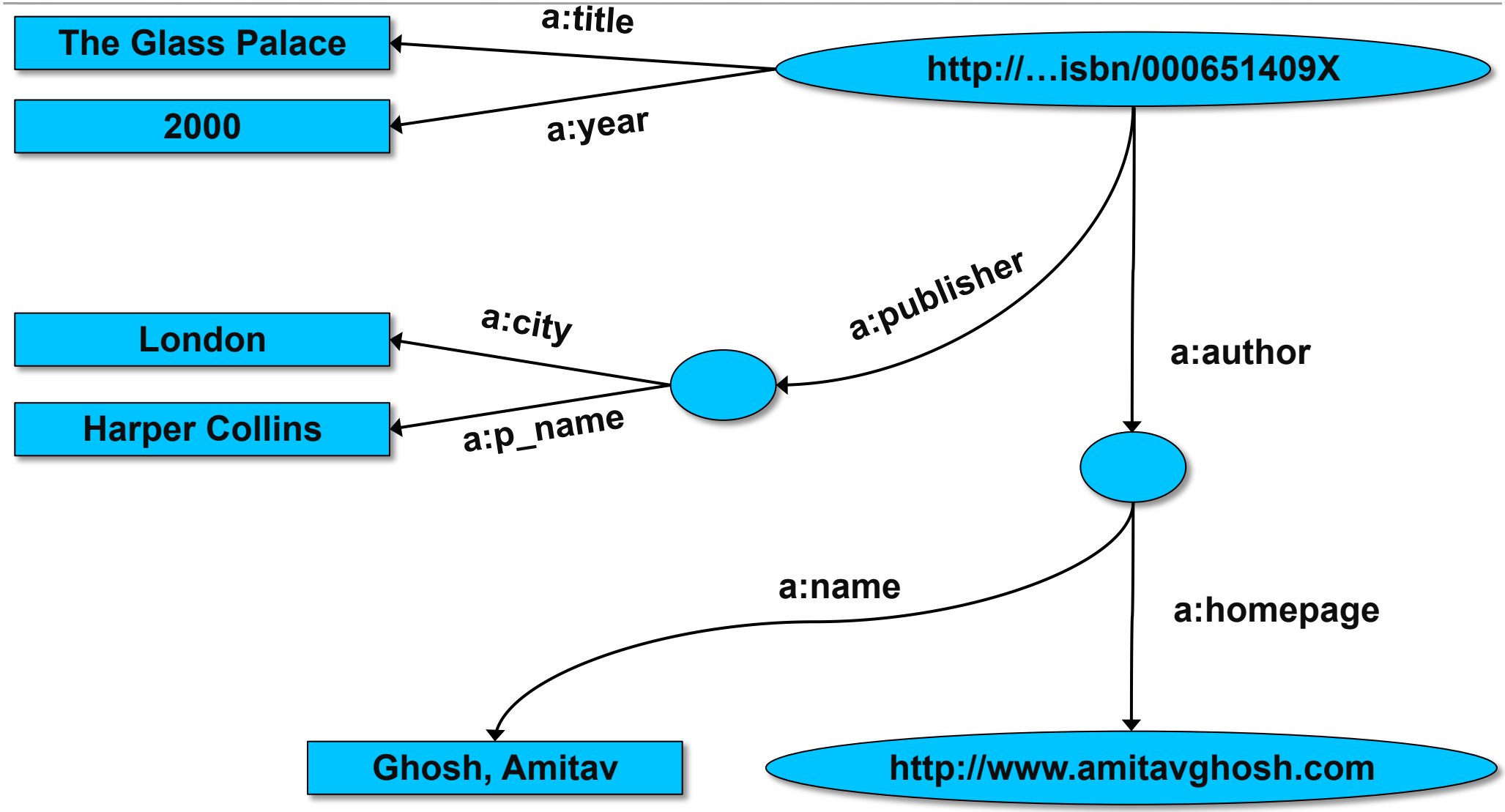
A simplified bookstore data (dataset “A”)

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

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

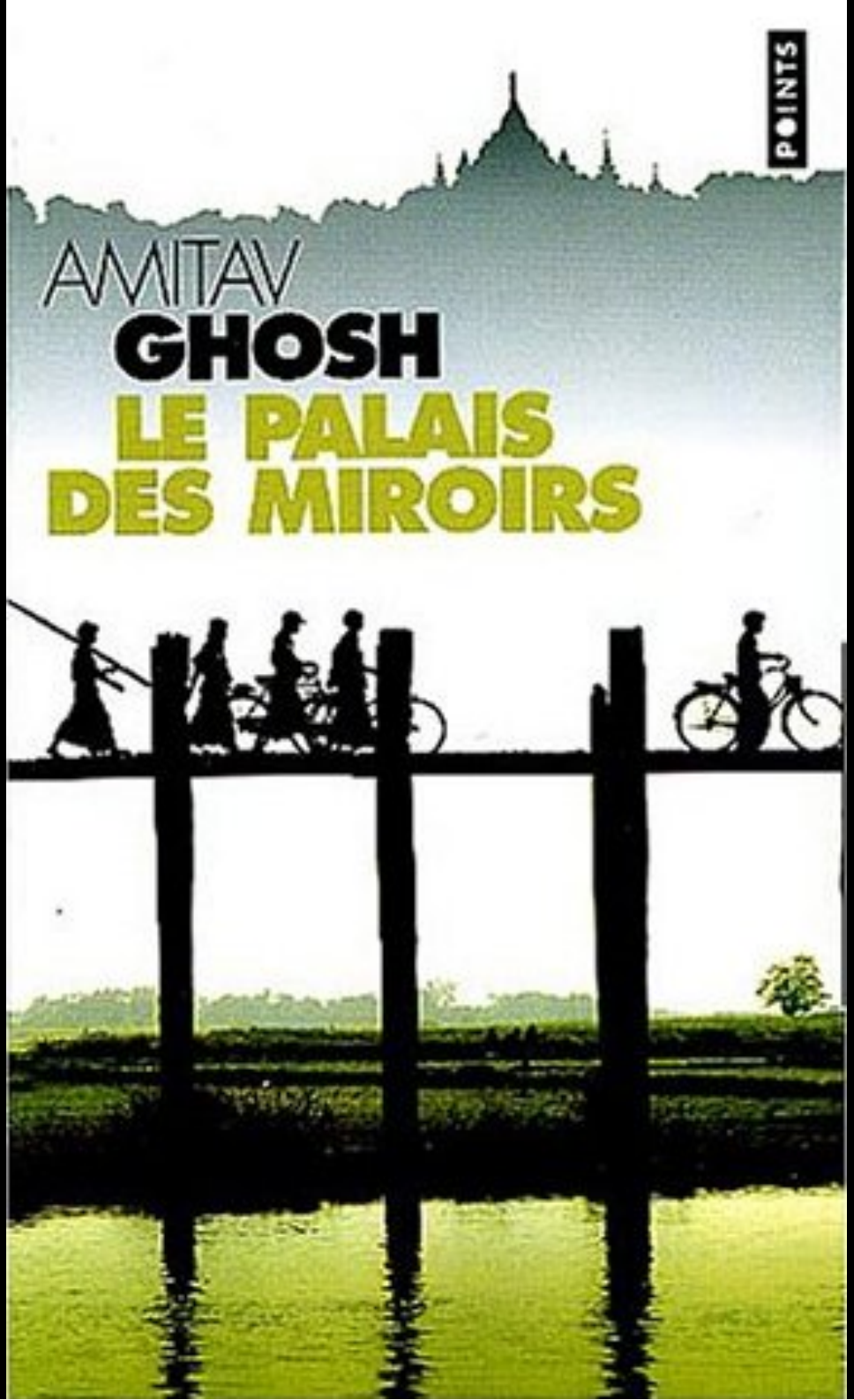
ID	Publisher’s name	City
id_qpr	Harper Collins	London

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



Some notes on the exporting the data

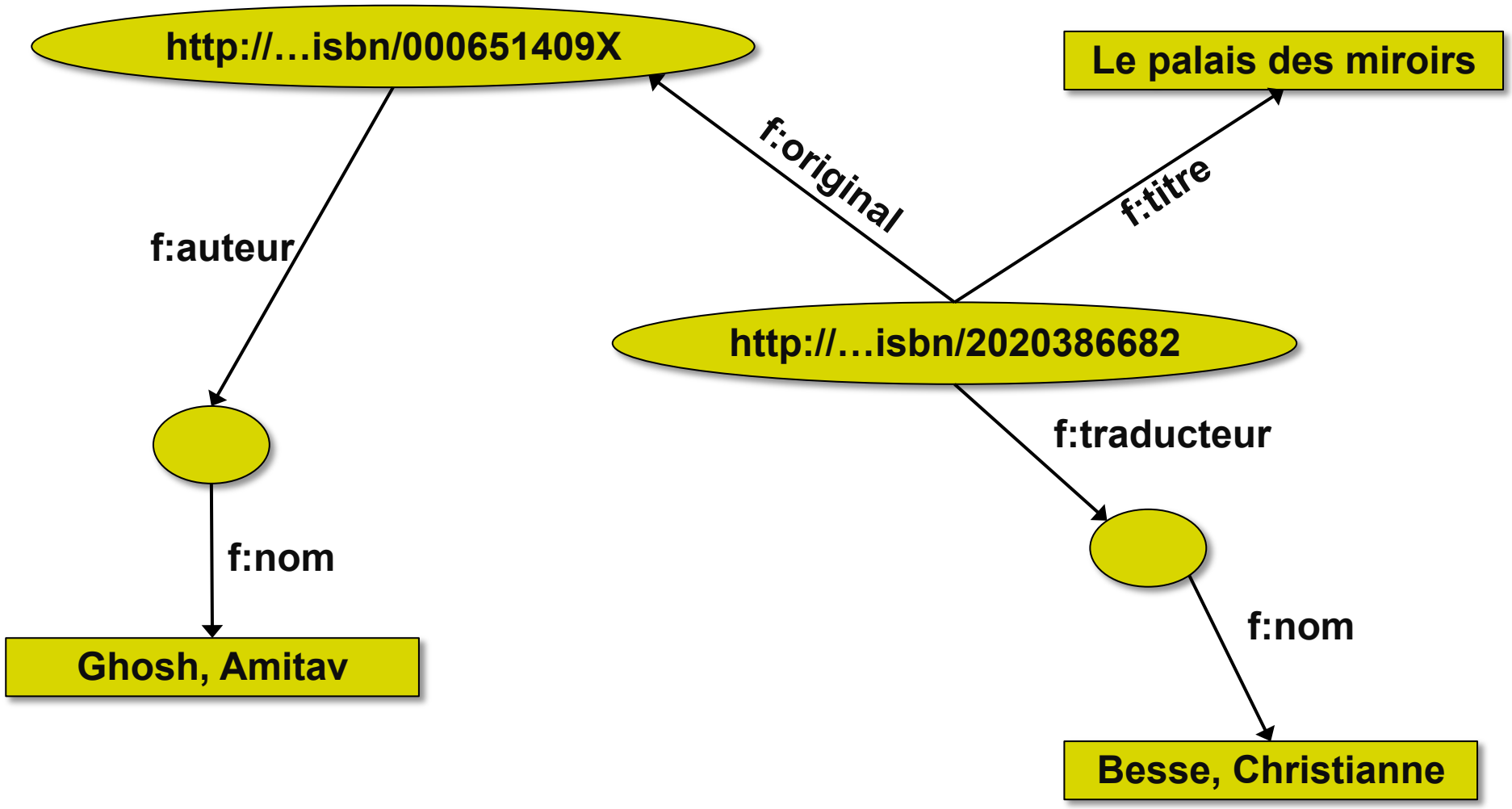
- Relations form a graph
  - the nodes refer to the “real” data or contain some literal
  - how the graph is represented in machine is immaterial for now


POINTS
AMITAV
GHOSH
LE PALAIS
DES MIROIRS

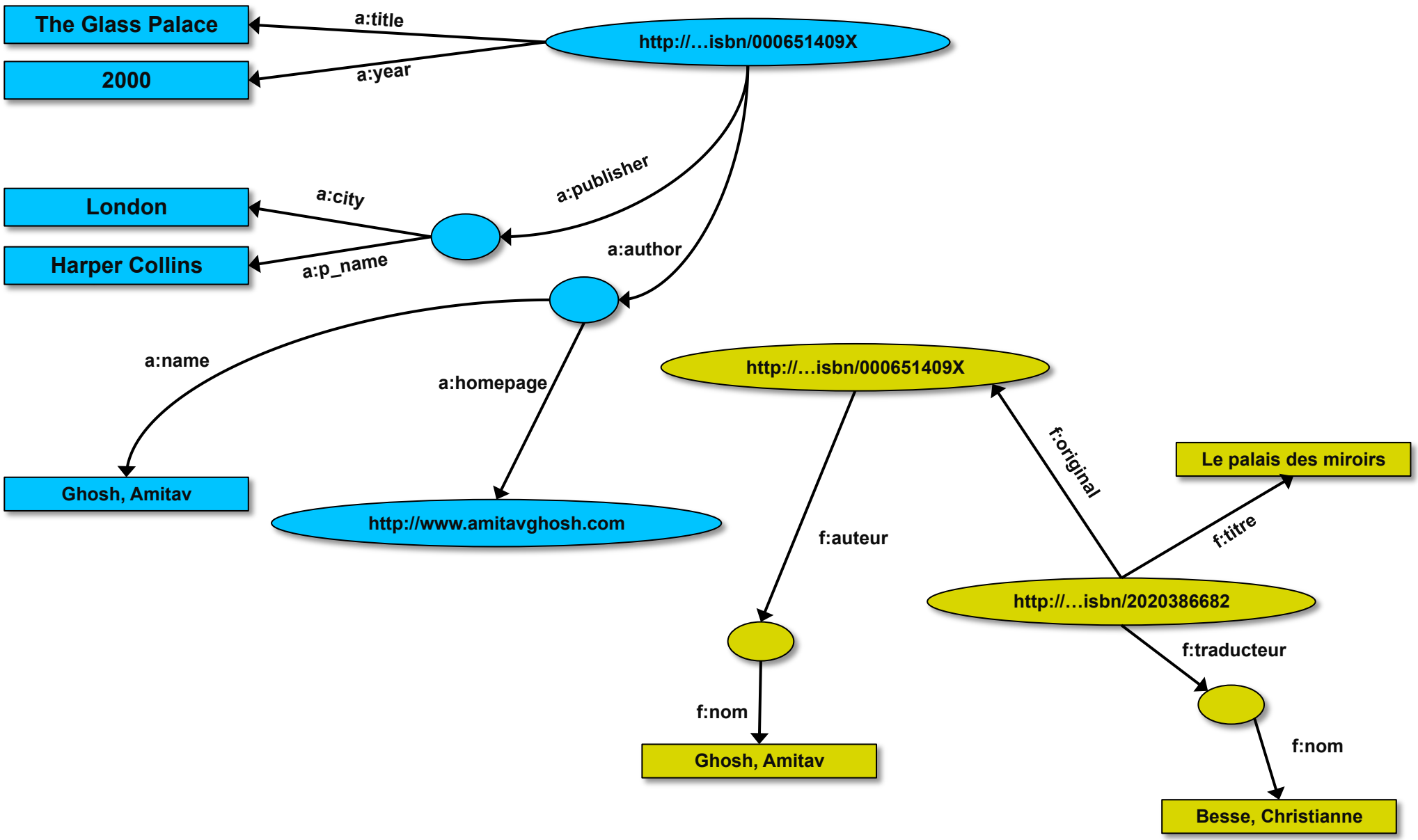
Another bookstore data (dataset “F”)

	A	B	C	D
1	ID	Titre	Traducteur	Original
2	ISBN 2020286682	Le Palais des Miroirs	$A12$	ISBN 0-00-6511409-X
3				
4				
5				
6	ID	Auteur		
7	ISBN 0-00-6511409-X	$A11$		
8				
9				
10	Nom			
11	Ghosh, Amitav			
12	Besse, Christianne			

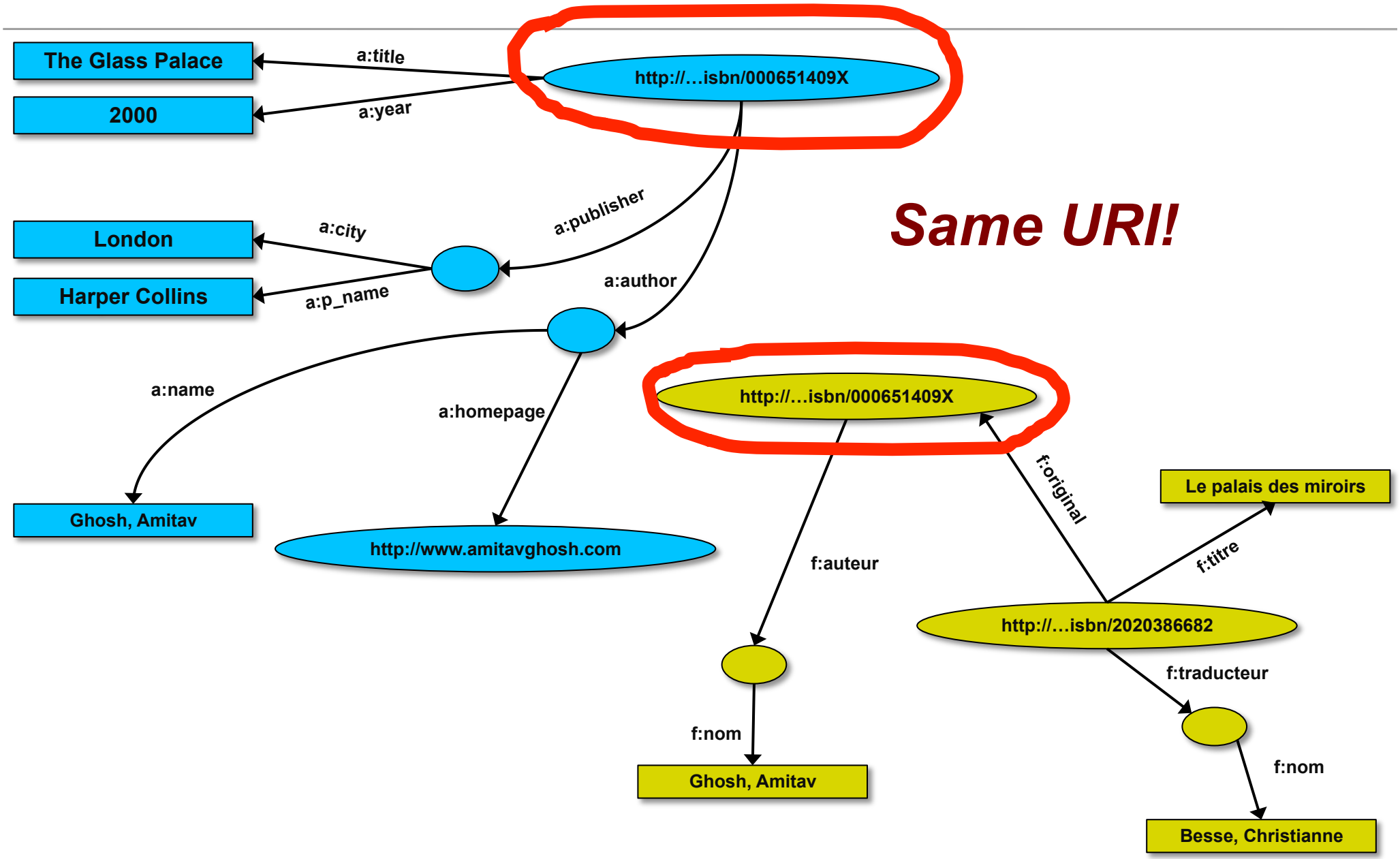
2nd: export your second set of data



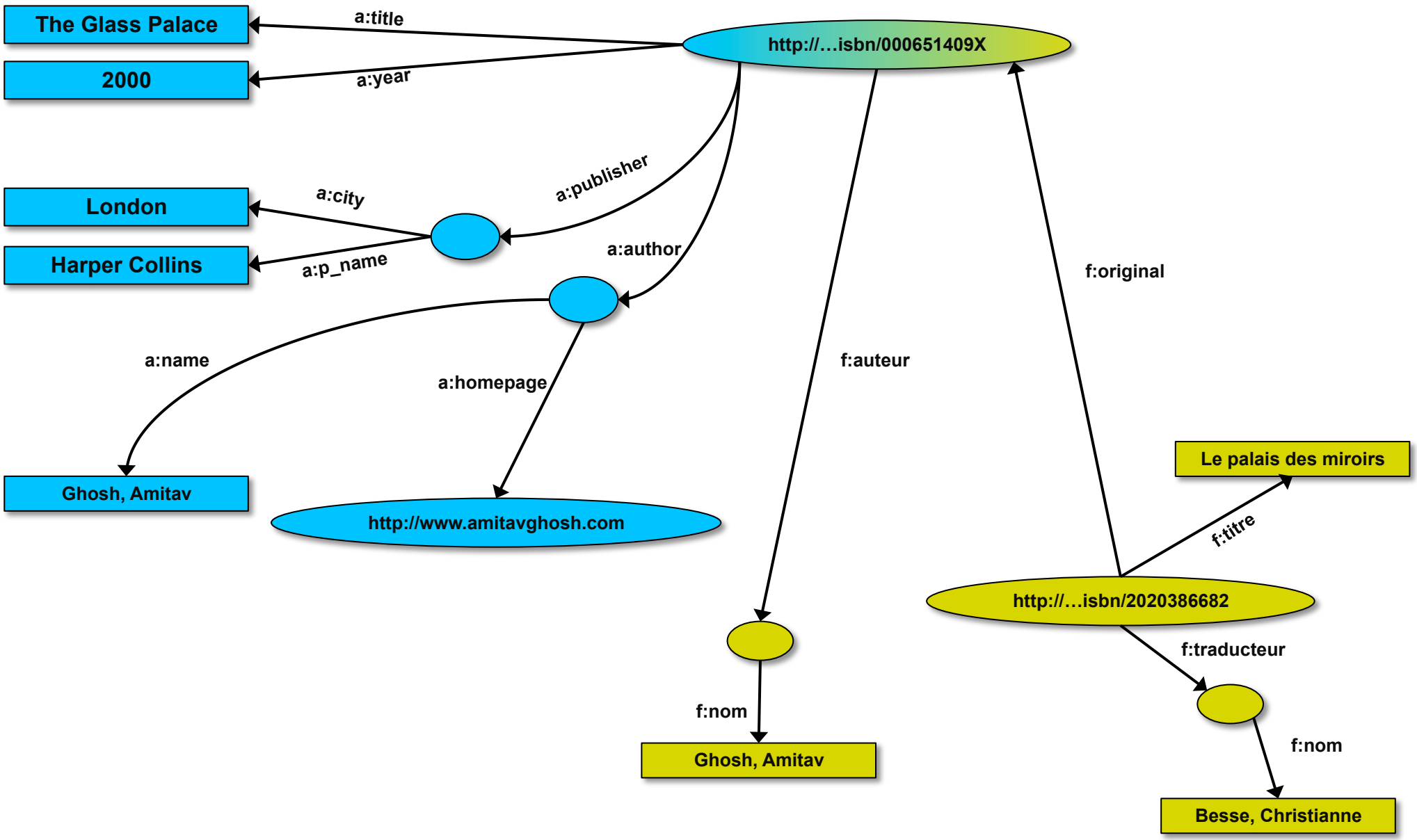
3rd: start merging your data



3rd: start merging your data (cont)

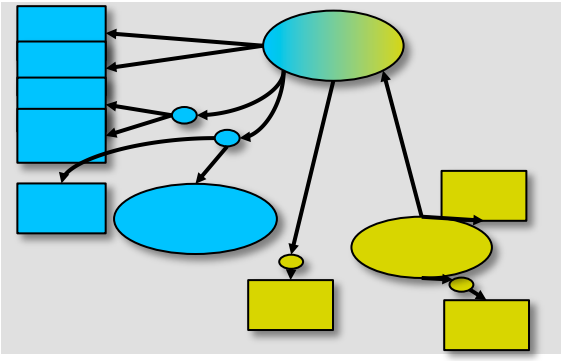


3rd: start merging your data



Start making queries…

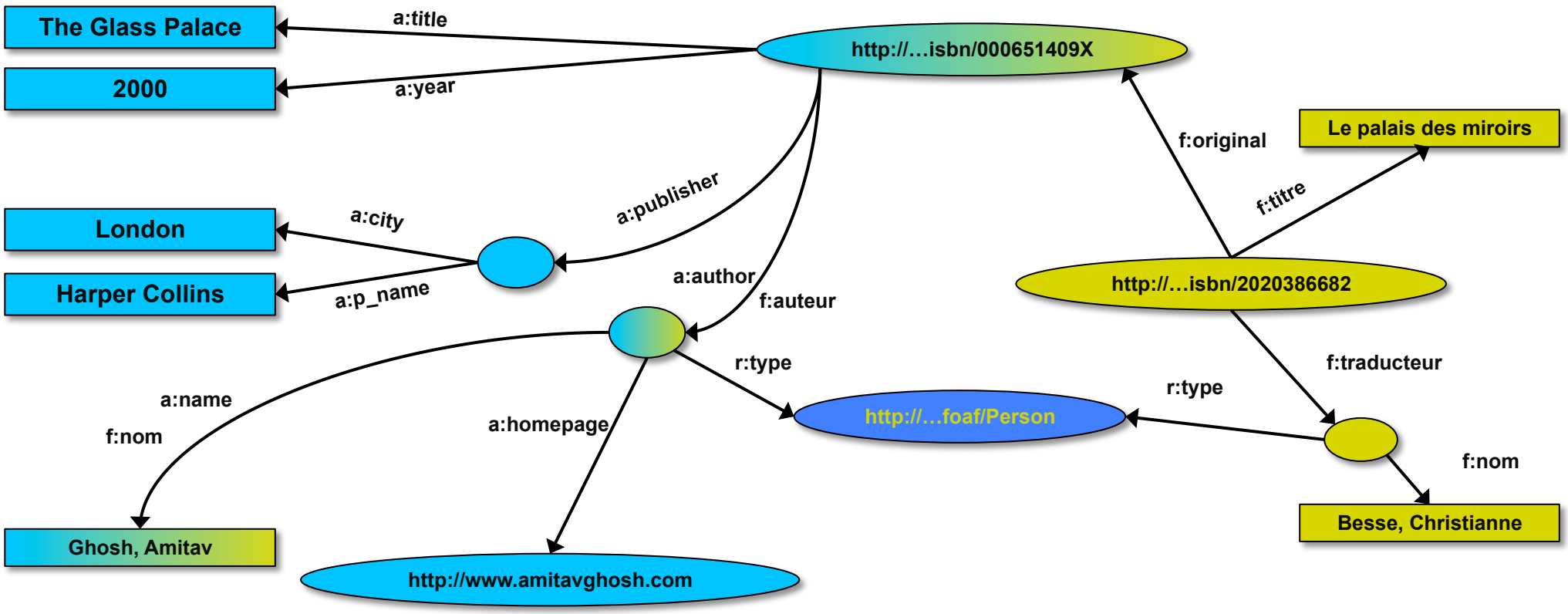
- User of data “F” can now ask queries like:
  - “give me the title of the original”
    - well, … « donnes-moi le titre de l’original »
- This information is not in the dataset “F”…
- …but can be retrieved by merging with dataset “A”!

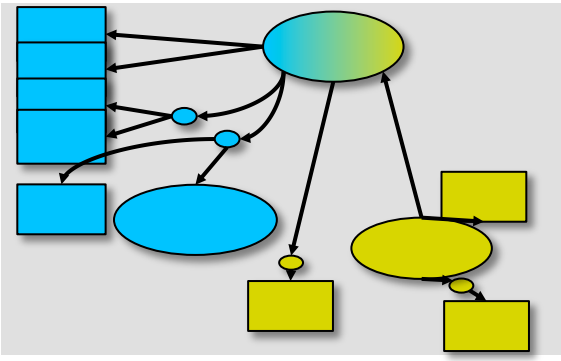


However, more can be achieved…

- We “feel” that a:author and f:auteur should be the same
- But an automatic merge doest 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

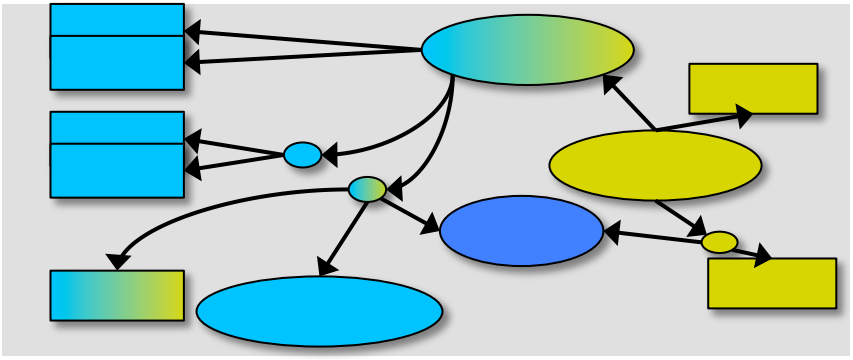
3rd revisited: use the extra knowledge





Start making richer queries!

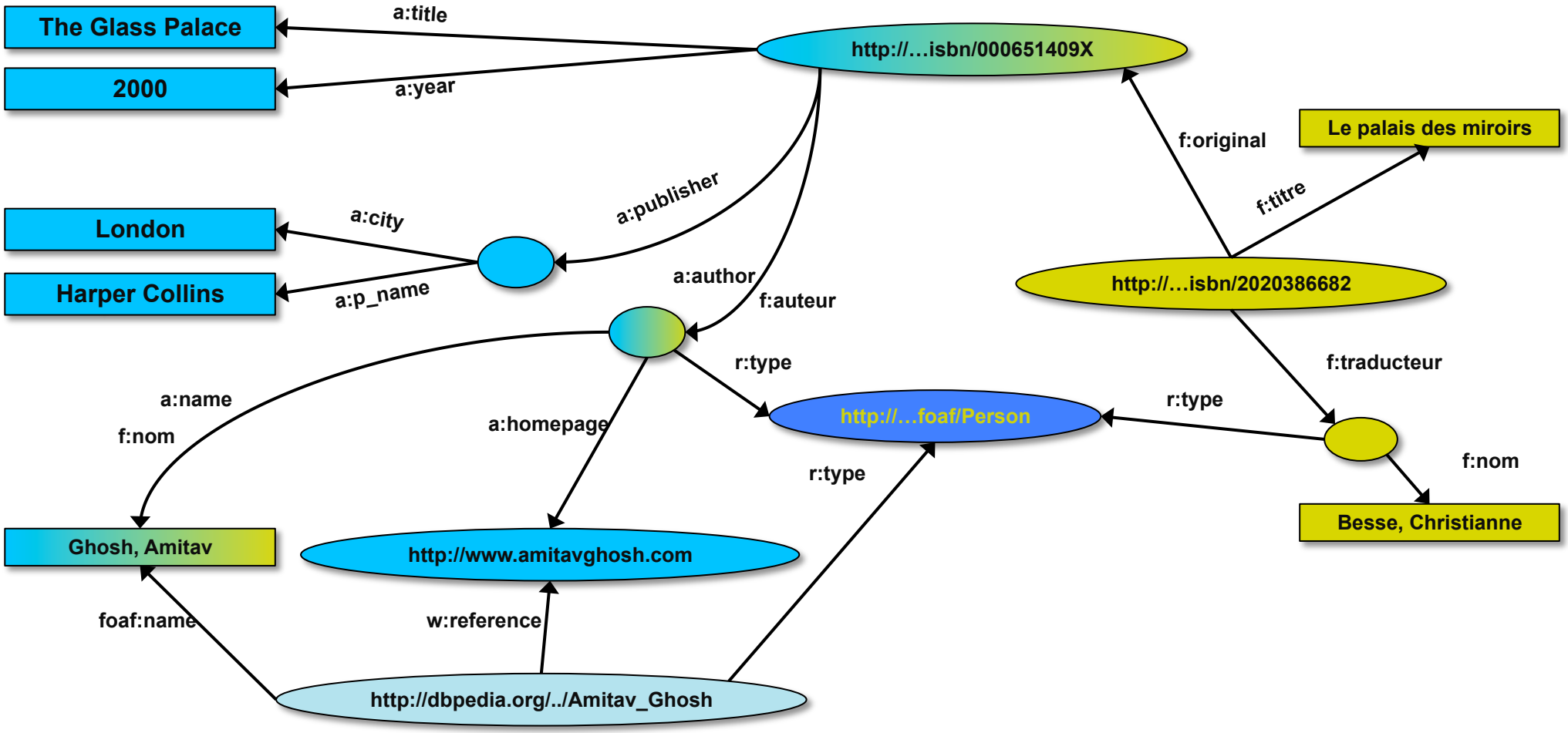
- User of dataset “F” can now query:
  - “donnes-moi la page d’accueil de l’auteur de l’original”
    - well… “give me the home page of the original’s ‘auteur’”
- The information is not in datasets “F” or “A”…
- …but was made available by:
  - merging datasets “A” and datasets “F”
  - adding three simple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



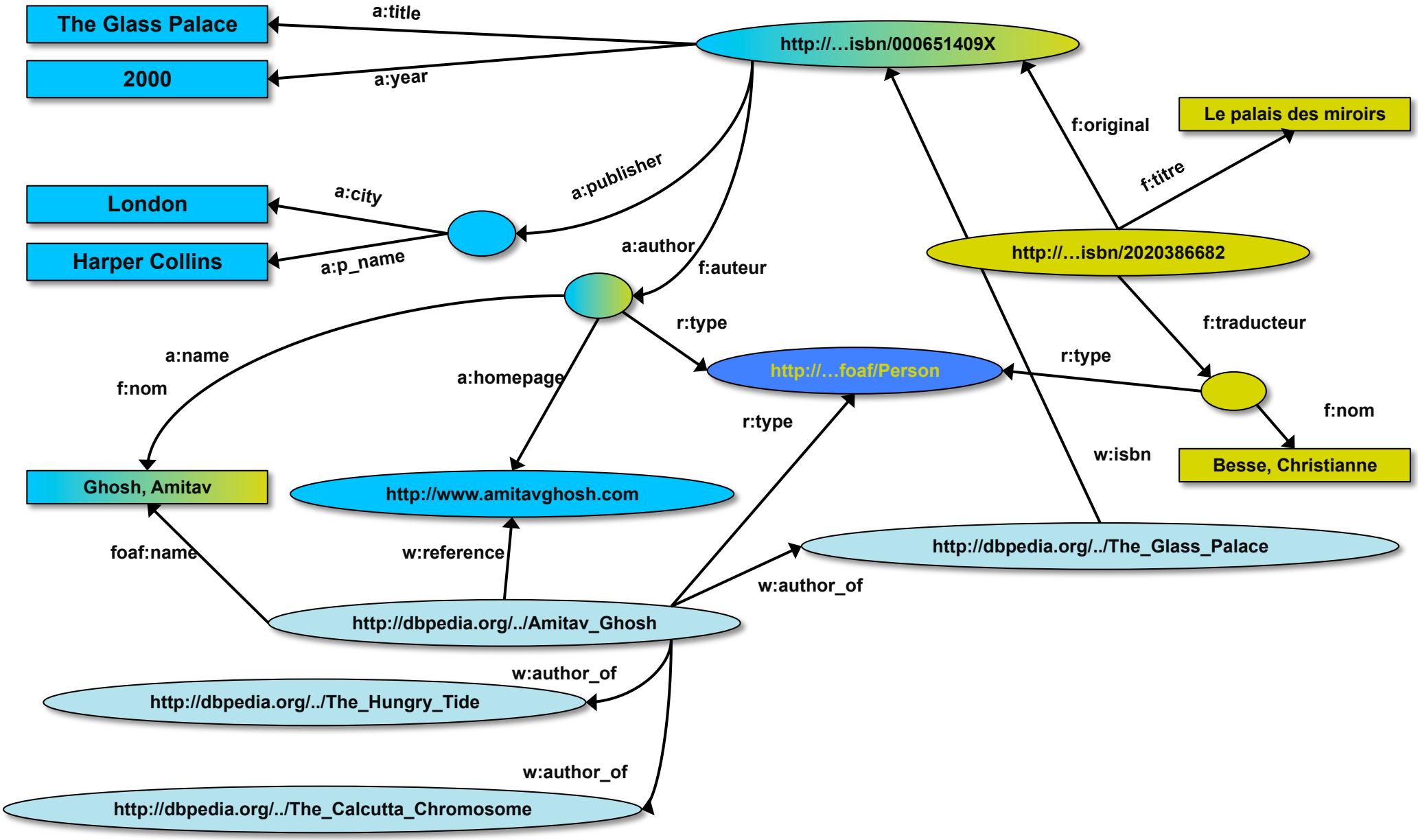
Combine with different datasets

- 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…

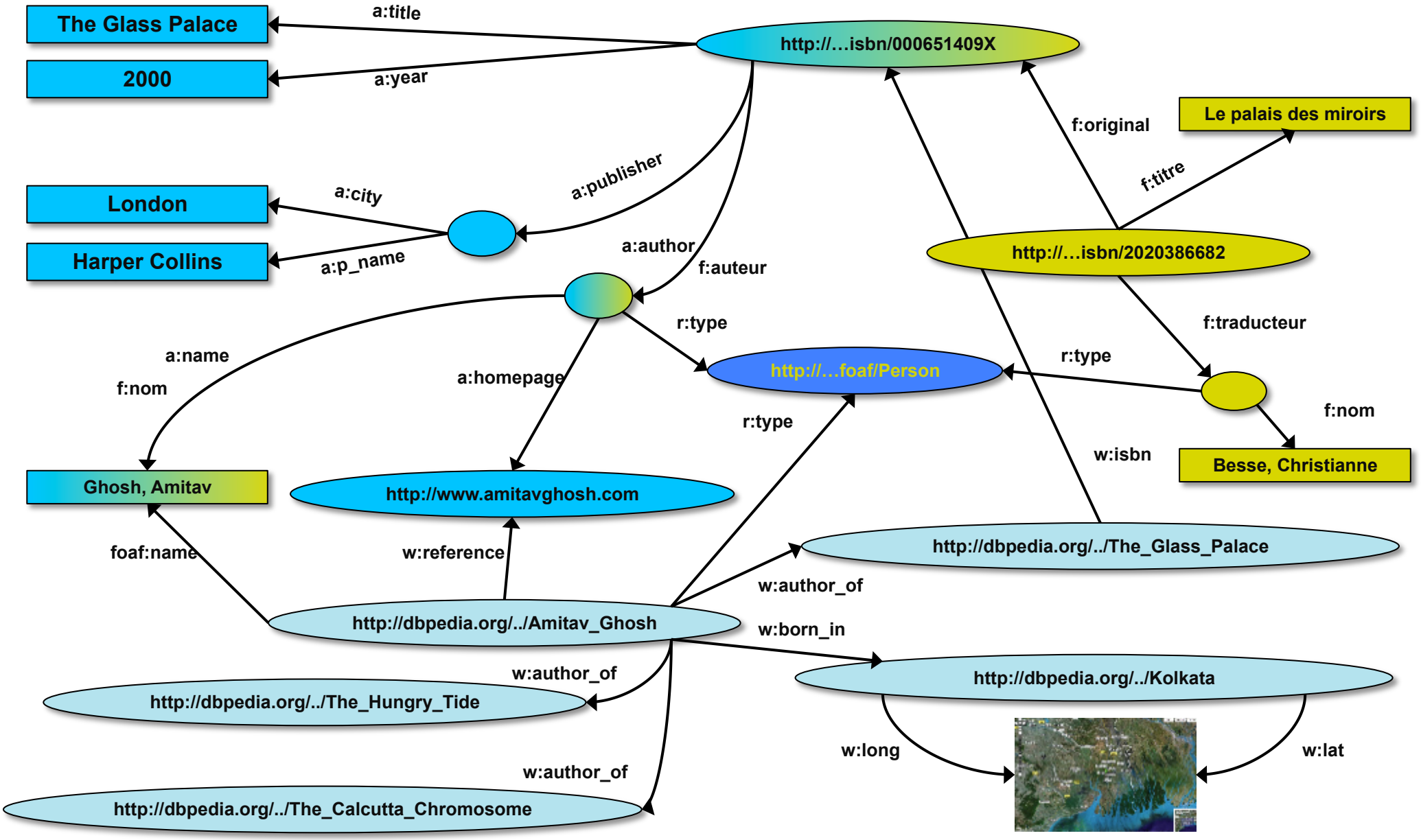
Merge with Wikipedia data



Merge with Wikipedia data



Merge with Wikipedia data



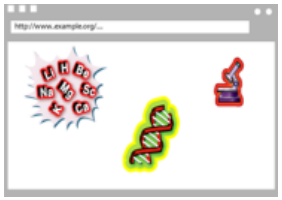
Is that surprising?

- 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

It could become even more powerful

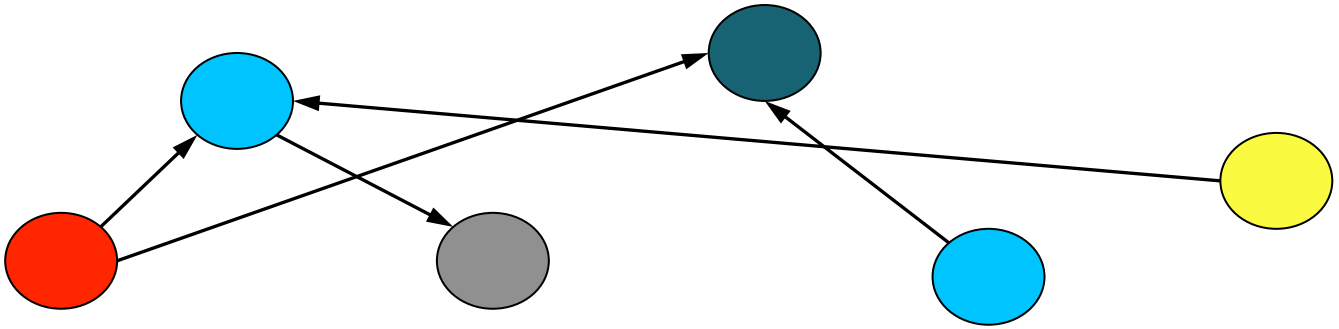
- We could add extra knowledge to the merged datasets
  - e.g., a full classification of various types of library data
  - geographical information
  - etc.
- This is where ontologies, extra rules, etc, come in
  - ontologies/rule sets can be relatively simple and small, or huge, or anything in between…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

What did we do?



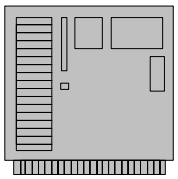
Applications

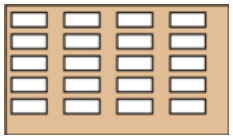
Manipulate
Query
…

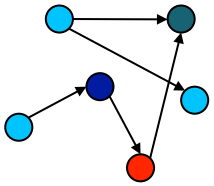


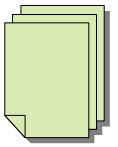
Data represented in abstract format

Map,
Expose,
…



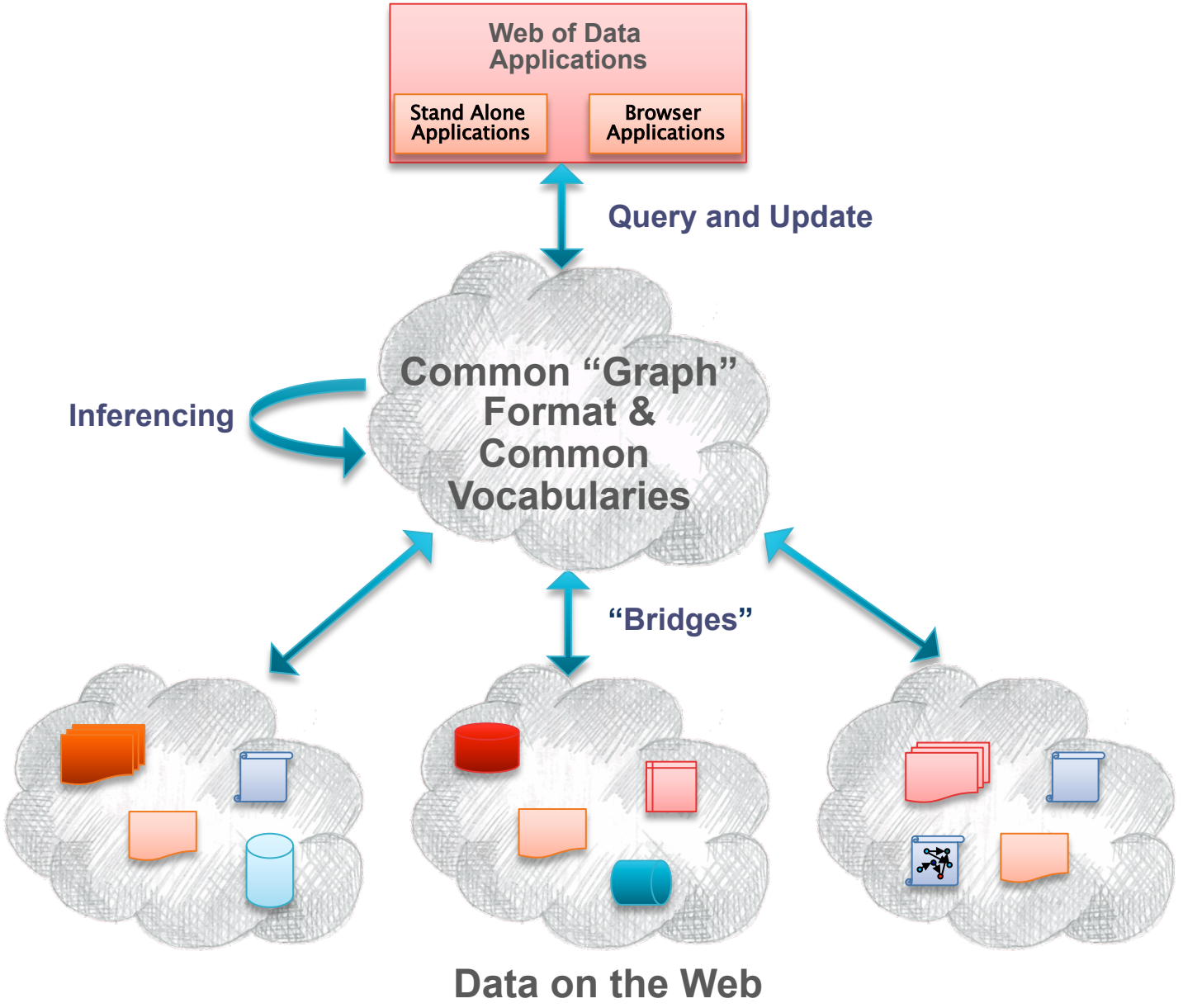






Data in various formats

What did we do? (alternate view)



The abstraction pays off because…

- … the graph representation is independent of the exact structures
- … a change in local database schema's, XHTML structures, etc, do not affect the whole
  - "schema independence"
- … new data, new connections can be added seamlessly

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…


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, RDFa, Json, …)

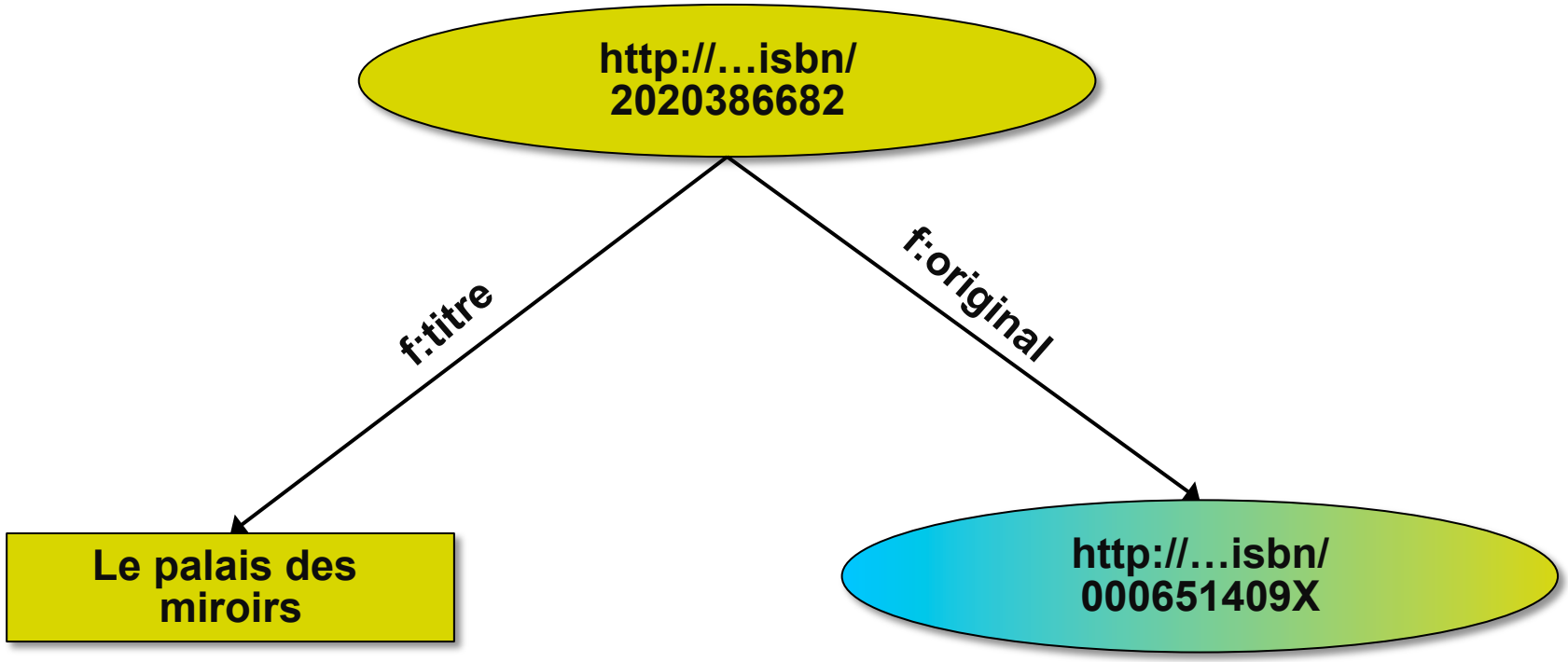
RDF triples (cont.)

- RDF triples are also referred to as “triplets”, or “statements”
- The “p” is sometimes referred to as “predicate”

RDF triples (cont.)

- Resources can use *any* URI
  - `http://www.example.org/file.html#home`
  - `http://www.example.org/f.xml#xpath(//q[@a=b])`
  - `http://www.example.org/form?a=b&c=d`
- RDF triples form a directed, labeled graph (the best way to think about them!)

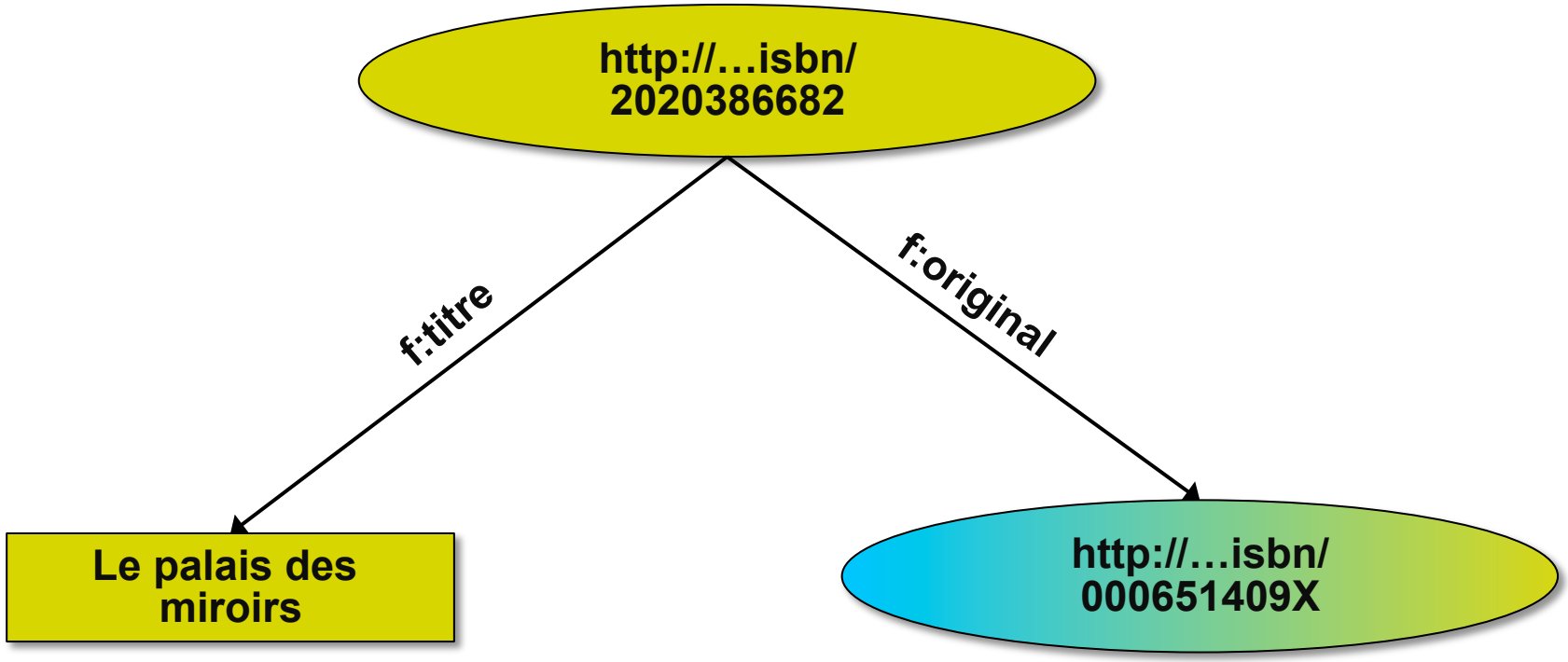
A simple RDF example (in RDF/XML)




```
<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>
```

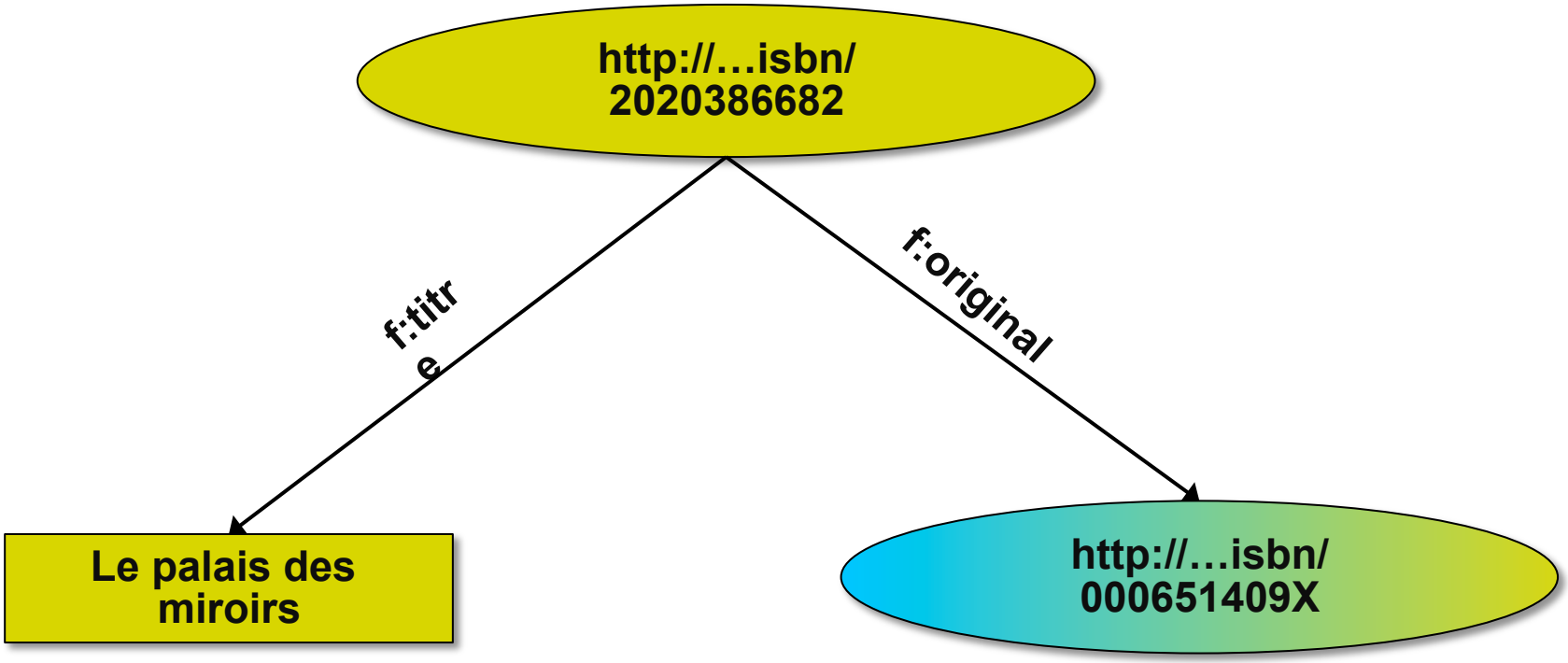
(Note: namespaces are used to simplify the URI-s)

A simple RDF example (in Turtle)




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

A simple RDF example (in RDFa)

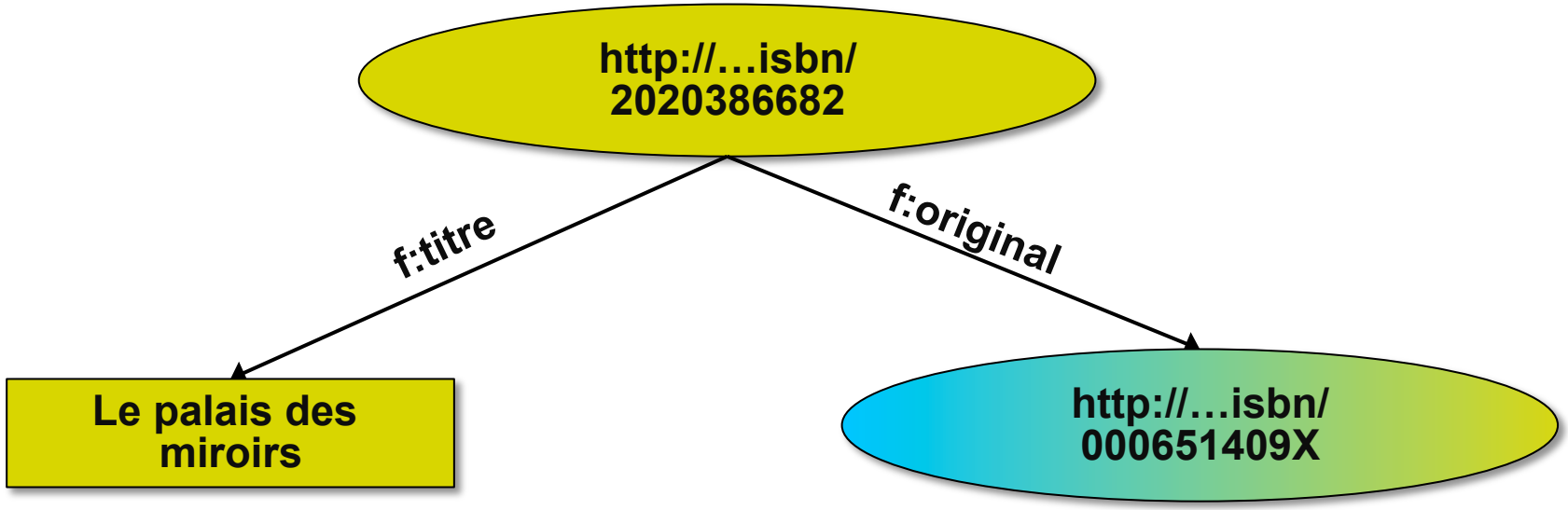



```
<p about="http://…/isbn/2020386682">The book entitled
"<span property="f:title" lang="fr">Le palais des mirroirs</span>"
is the French translation of the
"<span rel="f:original" resource="http://…/isbn/000651409X">Glass
Palace</span>"</p> .
```

URI-s play a fundamental role

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

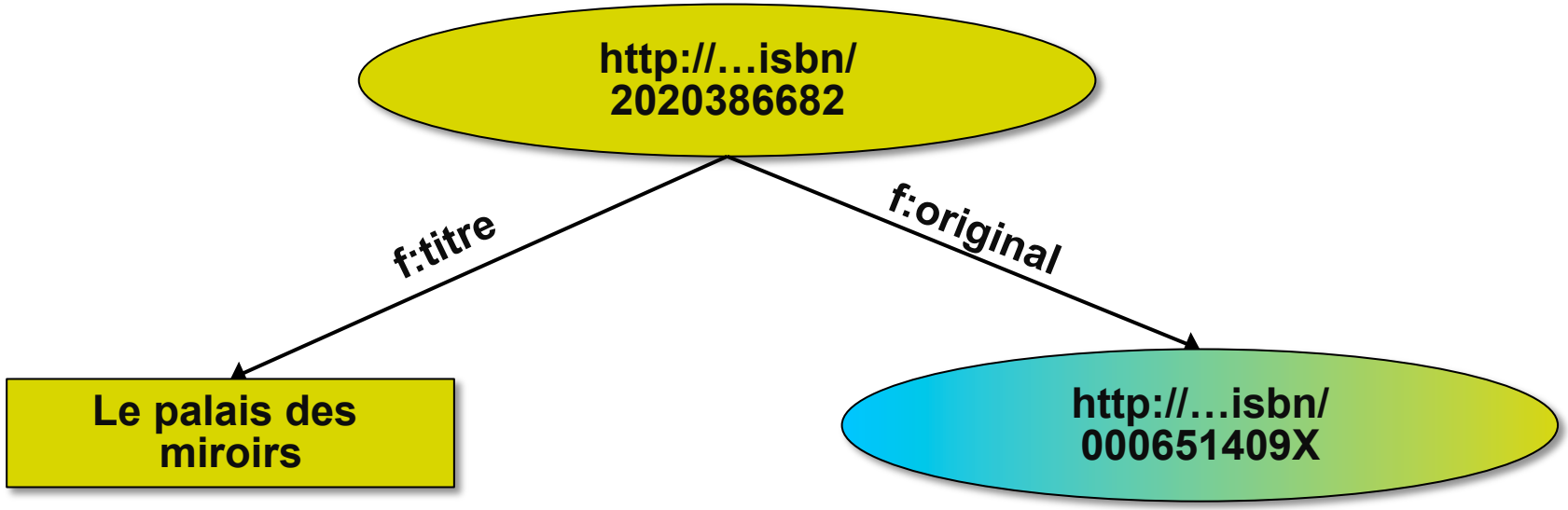
RDF/XML principles



- Encode nodes and edges as elements or literals:

```
«Element for http://…/isbn/2020386682»
  «Element for original»
      «Element for http://…/isbn/000651409X»
  «/Element for original»
«/Element for http://…/isbn/2020386682»
«Element for http://…/isbn/2020386682»
  «Element for titre»
      Le palais des mirroirs
  «/Element for titre»
«/Element for http://…/isbn/2020386682»
```

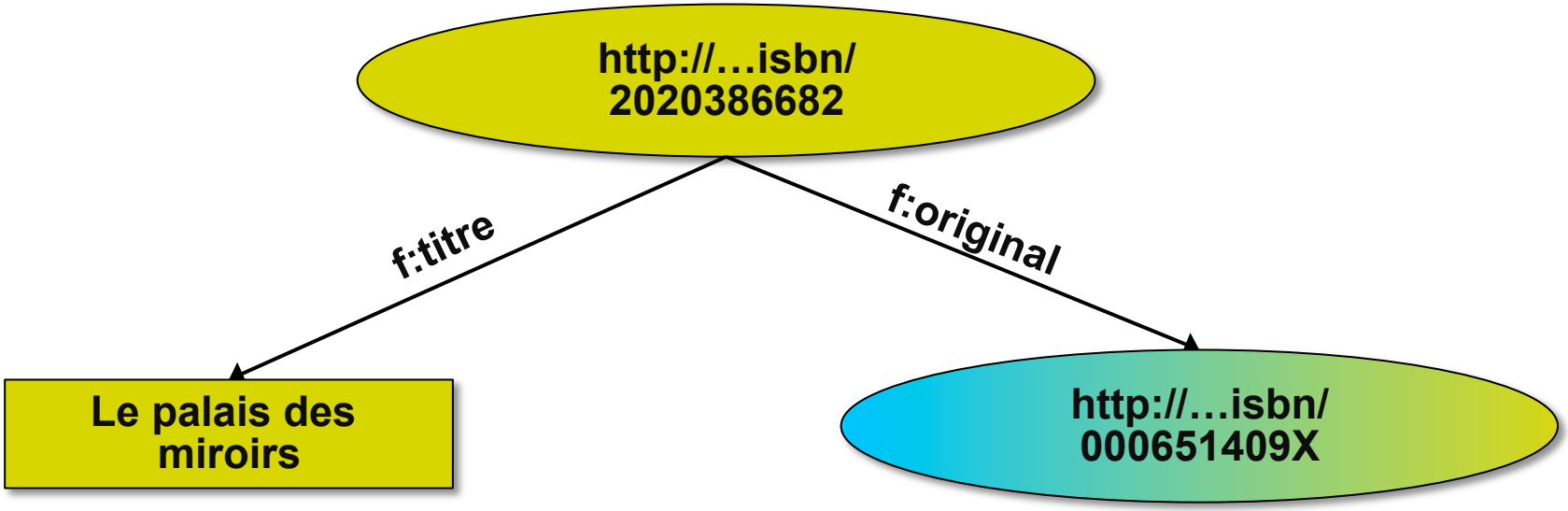
RDF/XML principles (cont.)



- Encode the resources (i.e., the nodes):

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about="http://…/isbn/2020386682">
        «Element for original»
            <rdf:Description rdf:about="http://…/isbn/000651409X"/>
        «/Element for f:original»
    </rdf:Description>
<rdf:RDF>
```

RDF/XML principles (cont.)



- Encode the properties (i.e., edges) in their own namespaces:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:f="http://www.editeur.fr"">
    <rdf:Description rdf:about="http://…/isbn/2020386682">
        <f:original>
            <rdf:Description rdf:about="http://…/isbn/000651409X"/>
        </f:original>
    </rdf:Description>
<rdf:RDF>
```

Examples of RDF/XML “simplifications”

- Object references can be put into attributes
- Several properties on the same resource

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

- There are other “simplification rules”, see the “RDF/XML Serialization” document for details

“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

- The resource will be “visible” on the Web
  - care should be taken to define unique URI-s

```
<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>
```

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".
```

- Internal = these resources are not visible outside

Blank nodes: the system can 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:publisher>
      <rdf:Description>
          <a:p_name>HarpersCollins</a:p_name>
          …
      </rdf:Description>
  </a:publisher>
</rdf:Description>
```

Same in Turtle

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

More on blank nodes

- 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 URIs “on-the-fly”
- From a logic point of view, blank nodes represent an “existential” statement
  - “there is a resource such that…”

RDF in programming practice

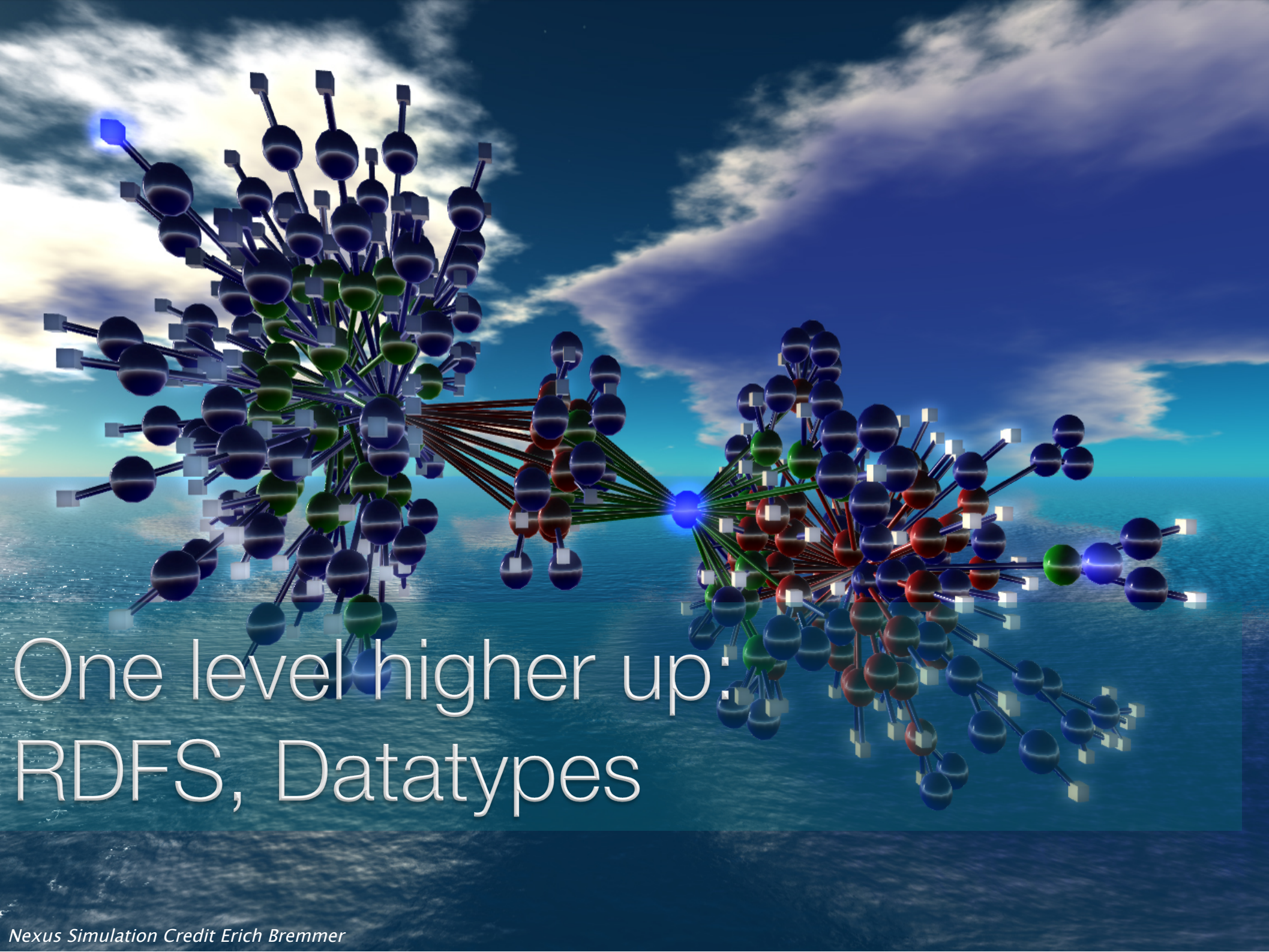
- For example, using Python+RDFLib:
  - a “Graph” object is created
  - the RDF file is parsed and results stored in the Graph
  - the Graph offers methods to retrieve (or add):
    - triples
    - (property,object) pairs for a specific subject
    - (subject,property) pairs for specific object
    - etc.
  - the rest is conventional programming…
- Similar tools exist in Java, PHP, etc.

Python example using RDFLib

```
# create a graph from a file
graph = rdflib.Graph()
graph.parse("filename.rdf", format="rdfxml")
# take subject with a known URI
subject = rdflib.URIRef("URI_of_Subject")
# process all properties and objects for this subject
for (s,p,o) in graph.triples((subject,None,None)) :
  do_something(p,o)
```

Merge in practice

- Environments merge graphs automatically
  - e.g., in Python+RDFLib, the Graph can load several files
  - the load merges the new statements automatically



Nexus Simulation Credit Erich Bremmer

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…

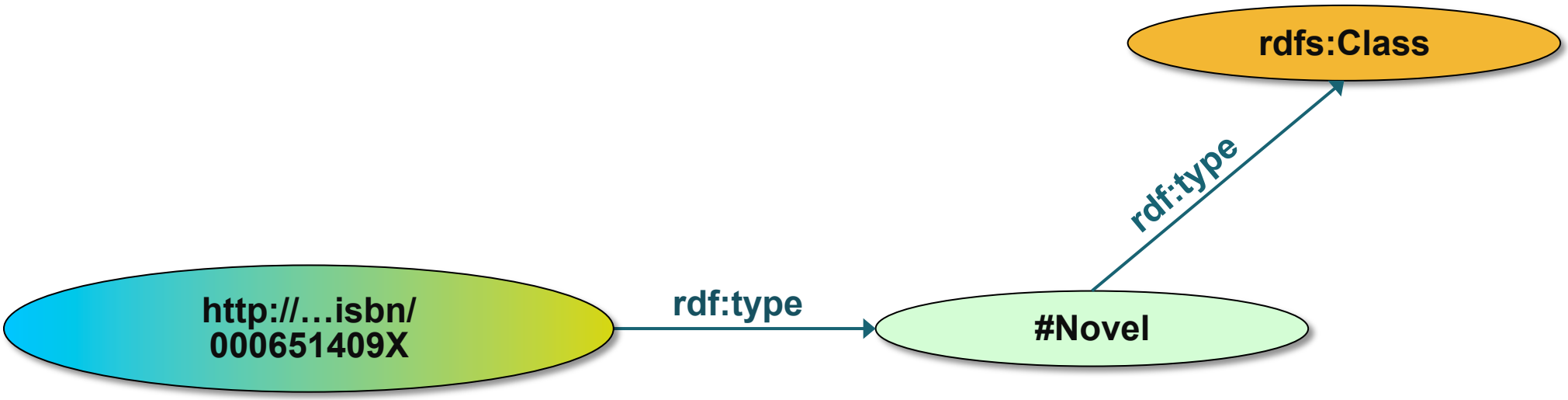
Classes, resources, …

- Think of well known traditional vocabularies:
  - 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 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/…/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>
```

An aside: typed nodes in RDF/XML

- A frequent simplification rule: instead of

```
<rdf:Description rdf:about="http://...">
    <rdf:type rdf:resource="http://..../something#ClassName">
    ...
</rdf:Description>
```

use:

```
<yourNameSpace:ClassName rdf:about="http://...">
    ...
</yourNameSpace:ClassName>
```

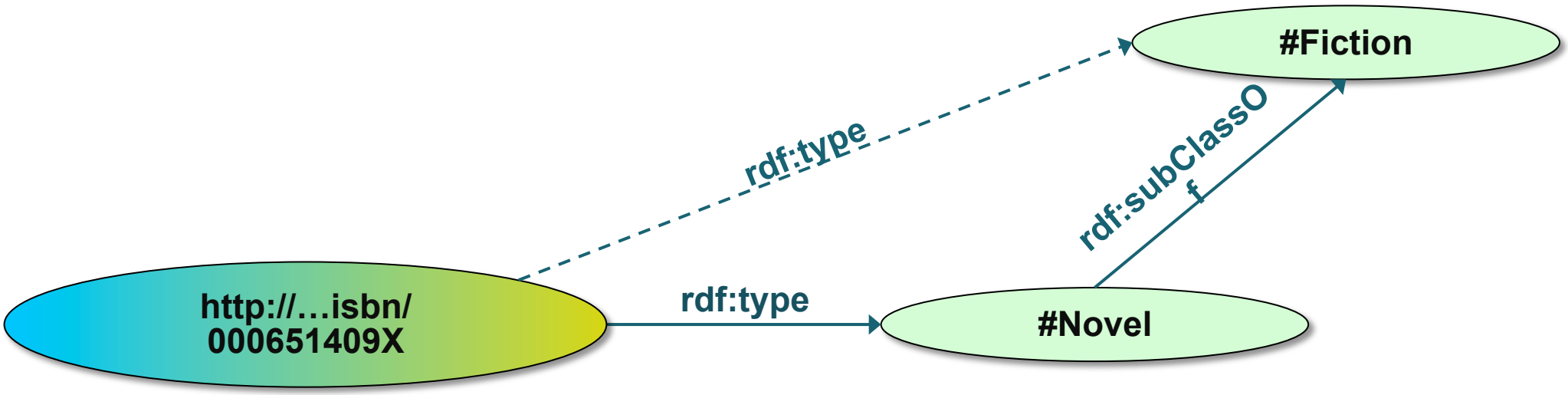
ie:

```
<a:Novel rdf:about="http://…/isbn/000651409X">
    ...
</a:Novel>
```

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 .
```

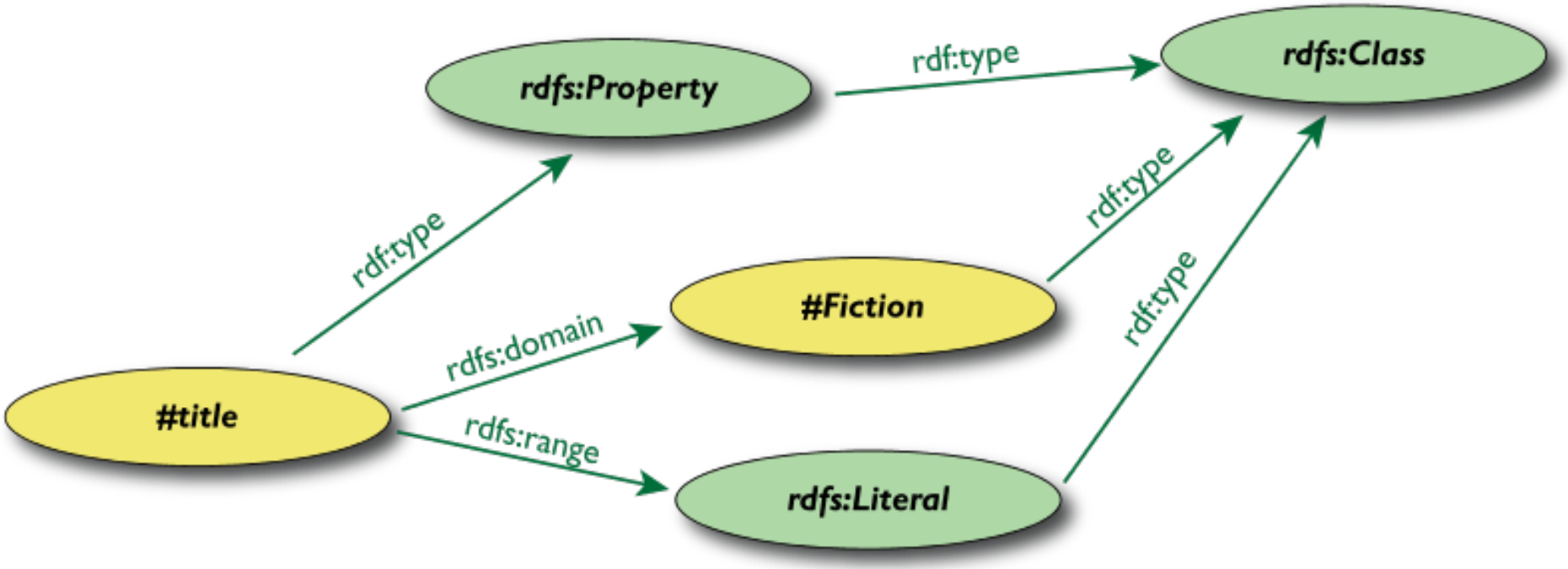
Properties

- 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

Properties (cont.)

- Properties are also resources (named via URI–s)…
- So properties of properties can be expressed as… RDF properties
  - this twists your mind a bit, but you can get used to it
- For example, (P rdfs:domain C) means:
  - P is a property
  - C is a class
  - when using P, I can infer that the “subject” is of type C

Property specification example



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

```
<rdf:Description rdf:about="http://…/isbn/000651409X">
   <page_number rdf:datatype="http://...#integer">543</page_number>
   <publ_date rdf:datatype="http://...#gYear">2000</publ_date>
   <price rdf:datatype="http://...#float">6.99</price>
</rdf:Description>
```

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

Examples for language tags

```
<rdf:Description rdf:about="http://…/isbn/000651409X">
   <title xml:lang="en">The Glass Palace</title>
   <fr:titre xml:lang="fr">Le palais des mirroirs</fr:titre>
</rdf:Description>
```

```
<http://…/isbn/000651409X>
      :title     "The Glass Palace"@en ;
      fr:titre   "Le palais des mirroirs"@fr .
```

XML literals in RDF/XML

- XML Literals
  - makes it possible to “include” XML vocabularies into RDF:

```
<rdf:Description rdf:about="#Path">
   <axsvg:algorithmUsed rdf:parseType="Literal">
      <math xmlns="...">
        <apply>
          <laplacian/>
          <ci>f</ci>
        </apply>
      </math>
   </axsvg:algorithmUsed>
</rdf:Description/>
```

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…)

Vocabularies

- RDFS makes it possible to define *vocabularies*:
  - collection of properties and classes
  - relationships among those and to *terms in other vocabularies*
- Some examples:
  - Dublin Core terms: creator, date, …
  - FOAF terms: characterization of persons
  - Good Relations: eCommerce terms
  - Creative Commons: copyright classes, license relations, …
  - schema.org terms: events, organizations, places, reviews, …
  - …


Some predefined structures...
(collections, containers)

Predefined classes and properties

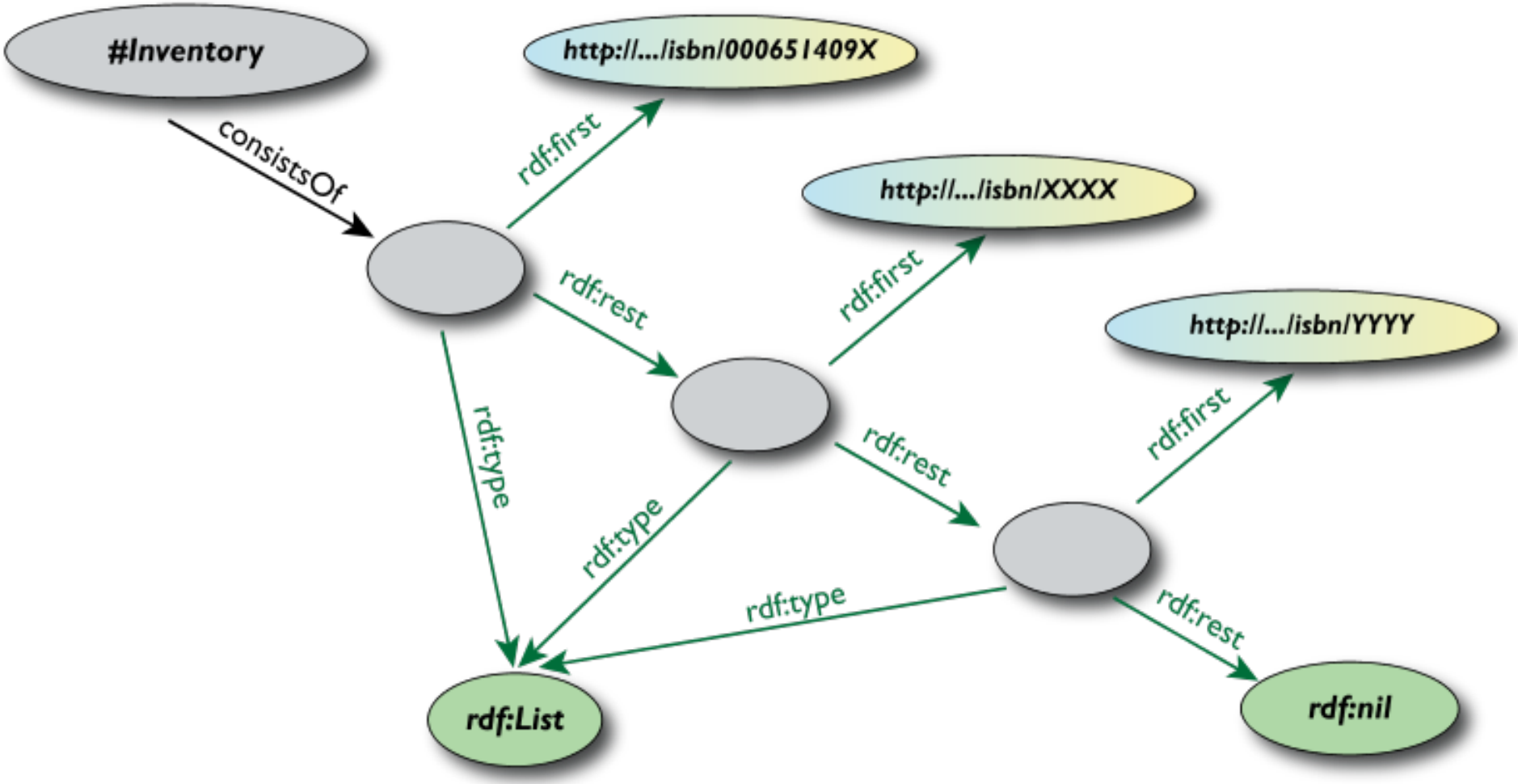
- RDF(S) has some predefined classes and properties
- These are not new “concepts” in the RDF Model, just resources with an agreed semantics
- Examples:
  - collections (a.k.a. lists)
  - containers: sequence, bag, alternatives
  - reification
  - rdfs:comment, rdfs:seeAlso, rdf:value

Collections (lists)

- We could have the following statement:
  - “The book inventory is a «thing» that consists of <http://…/isbn/000651409X>, <http://…/isbn/000XXXX>, …”
- But we also want to express the constituents in this order
- Using blank nodes is not enough

Collections (lists) (cont.)

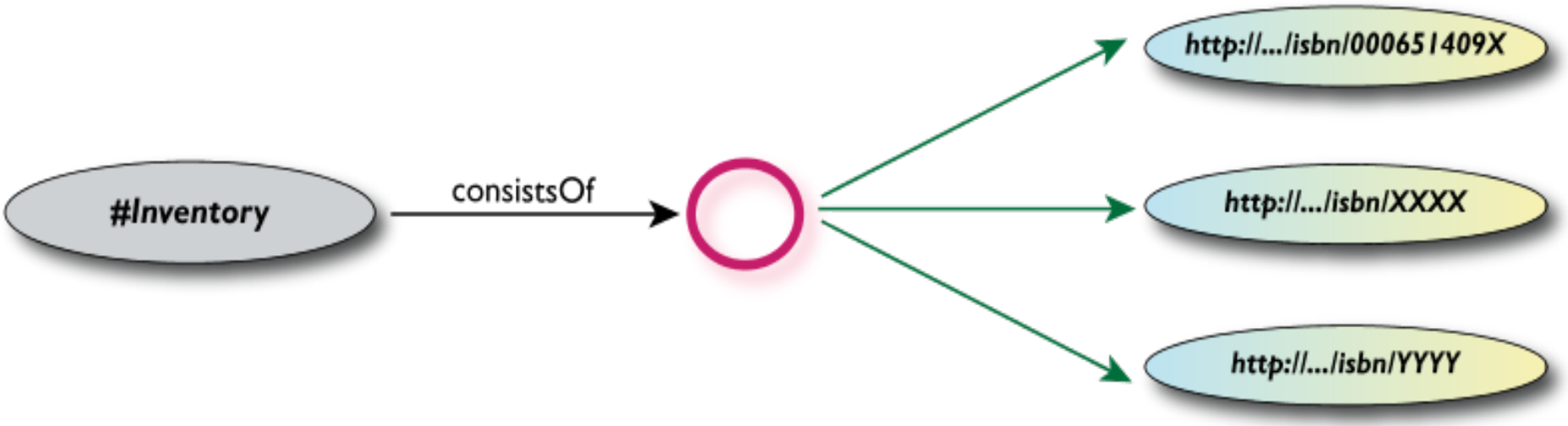
- Familiar structure for Lisp programmers…



The same in RDF/XML and Turtle

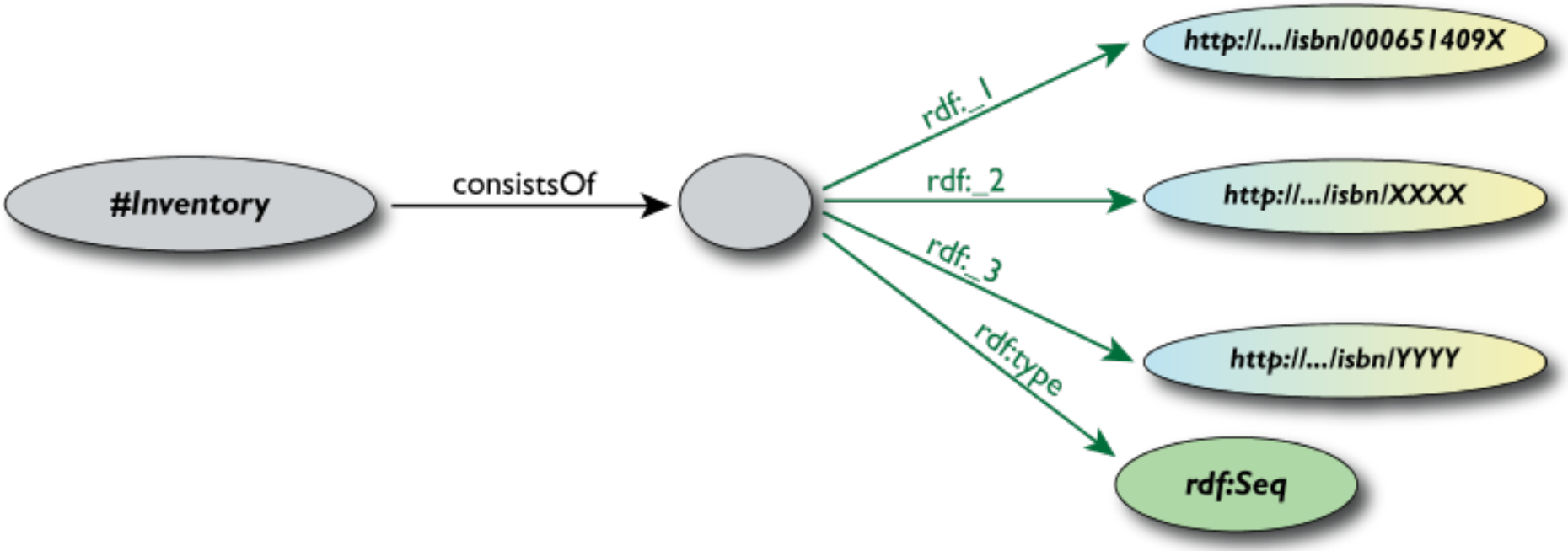
```
<rdf:Description rdf:about="#Inventory">
    <a:consistsOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://.../isbn/000651409X"/>
        <rdf:Description rdf:about="http://.../isbn/XXXX"/>
        <rdf:Description rdf:about="http://.../isbn/YYYY"/>
    </a:consistsOf>
</rdf:Description>
```

```
:Inventory a:consistsOf
     (<http://.../isbn/000651409X> <http://.../isbn/XXXX> ...) .
```



Sequences

- Use the predefined:
  - RDF Schema class Seq
  - RDF properties rdf:_1, rdf:_2, …
- The agreed semantics is of a sequential containment



Sequences serialized

- In RDF/XML:

```
<rdf:Description rdf:about="#Inventory">
  <a:consistsOf>
      <rdf:Description>
          <rdf:type rdf:resource="http:...rdf-syntax-ns#Seq">
          <rdf:_1 rdf:resource="http://.../isbn/000651409X">
          ...
      </rdf:Description>
  </a:consistsOf>
</rdf:Description/>
```

In Turtle:

```
:Inventory
  a:consistsOf [
    rdf:type <http:...rdf-syntax-ns#Seq>;
    rdf:_1   <http://.../isbn/000651409X>;
          ...
  ].
```

Sequences (simplified RDF/XML)

```
<rdf:Description rdf:about="#Inventory">
    <a:consistsOf>
        <rdf:Seq>
            <rdf:li rdf:resource="http://.../isbn/000651409X">
            ...
        </rdf:Seq>
    </a:consistsOf>
</rdf:Description/>
```

Other containers

- Also defined in RDFS
  - rdf:Bag
    - a general bag, no particular semantics attached
  - rdf:Alt
    - agreed semantics: only one of the constituents is “valid”
- Note: these containers are incompletely defined semantically; it is better not to use them…
  - use repeated predicates for bags
  - use lists for sequences

RDF Data and Web Pages?

(RDFa, microdata, microformats, GRDDL)

Photo credit "shetladd", Flickr

RDF with HTML

- Obviously, a huge source of information
- By adding some “meta” information, the same source can be reused for, e.g., data integration, better mashups, etc.
  - typical example: your personal information, like address, should be readable for humans and processable by machines
- Some solutions have emerged:
  - use microformats and convert the content into RDF
  - add extra statements in microdata, that can be converted to RDF
  - add RDF statements directly into XHTML via RDFa

Microformats

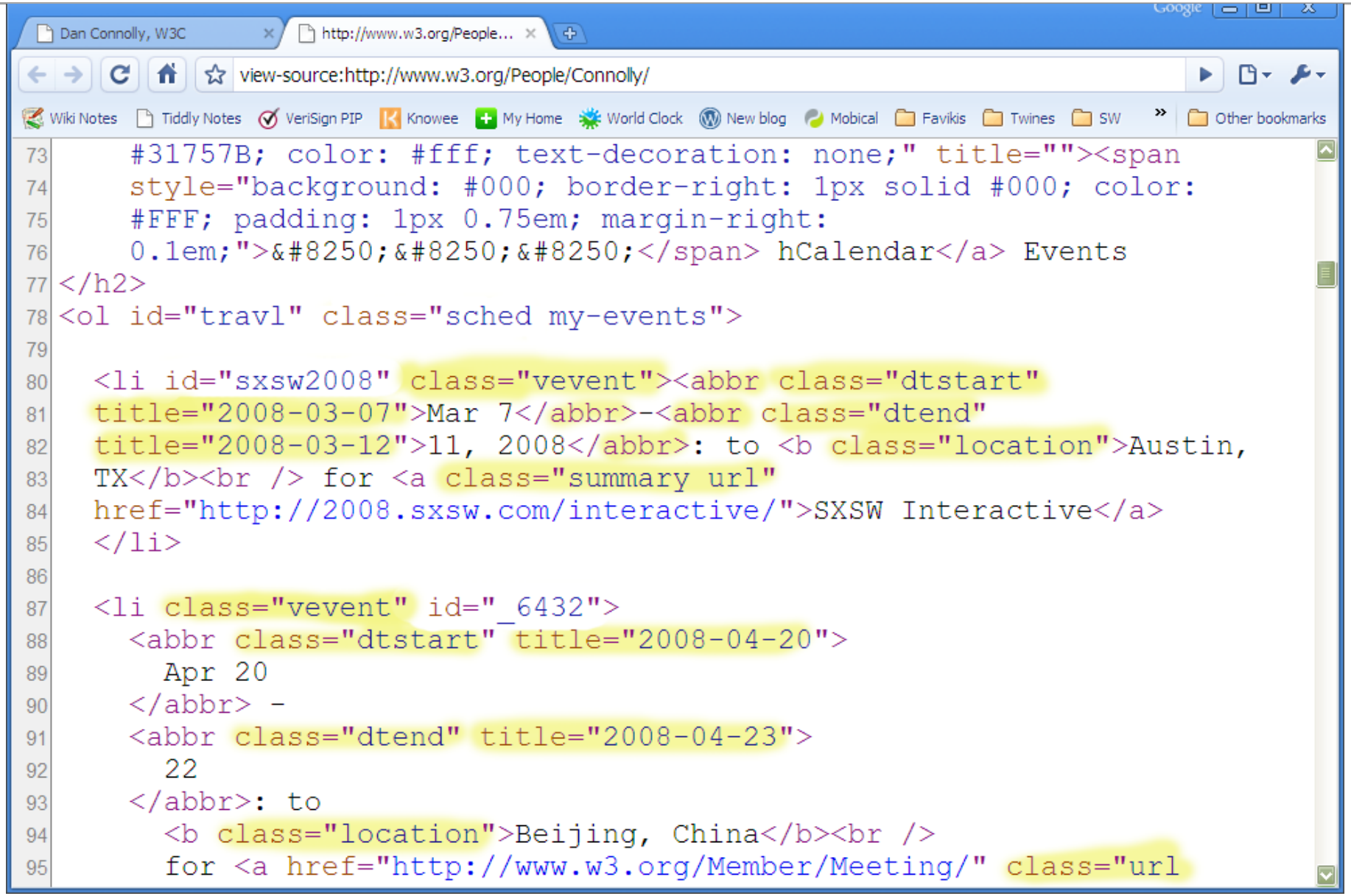
- Not a Semantic Web specification per se
  - there is a separate microformat community
- Approach: re-use (X)HTML attributes and elements to add “meta” information
  - typically @abbr, @class, @title, …
  - different agreements for different applications
- It is not a specification per se, rather a common methodology
  - each vocabulary has its own, independent specification

Microformat example: hCalendar

- Goal: “markup” calendaring information on your HTML page
  - use a community agreement using, eg, :
    - @class for event name
    - abbr element for dates
    - @title for date values
    - etc.
- Automatic procedures (ie, calendaring applications) may then get to the right data

Microformat example: hCalendar

Behind the scenes…



Microformat extraction

- To use it on the Semantic Web, microformat data should be converted to RDF
- A simple transformation (eg, in XSLT) can be defined, yielding:

```
<http://www.w3.org/People/Connolly/#sxsw2008>
    a hcal:Vevent;
    hcal:organizer <http://www.w3.org/People/Connolly/#me>;
    hcal:summary "SXSW Interactive";
    hcal:dtstart "2008-03-07"^^xsd:date;
    hcal:dtend "2008-03-12"^^xsd:date;
    hcal:url <http://2008.sxsw.com/interactive/>;
    hcal:location "Austin, TX" .
```

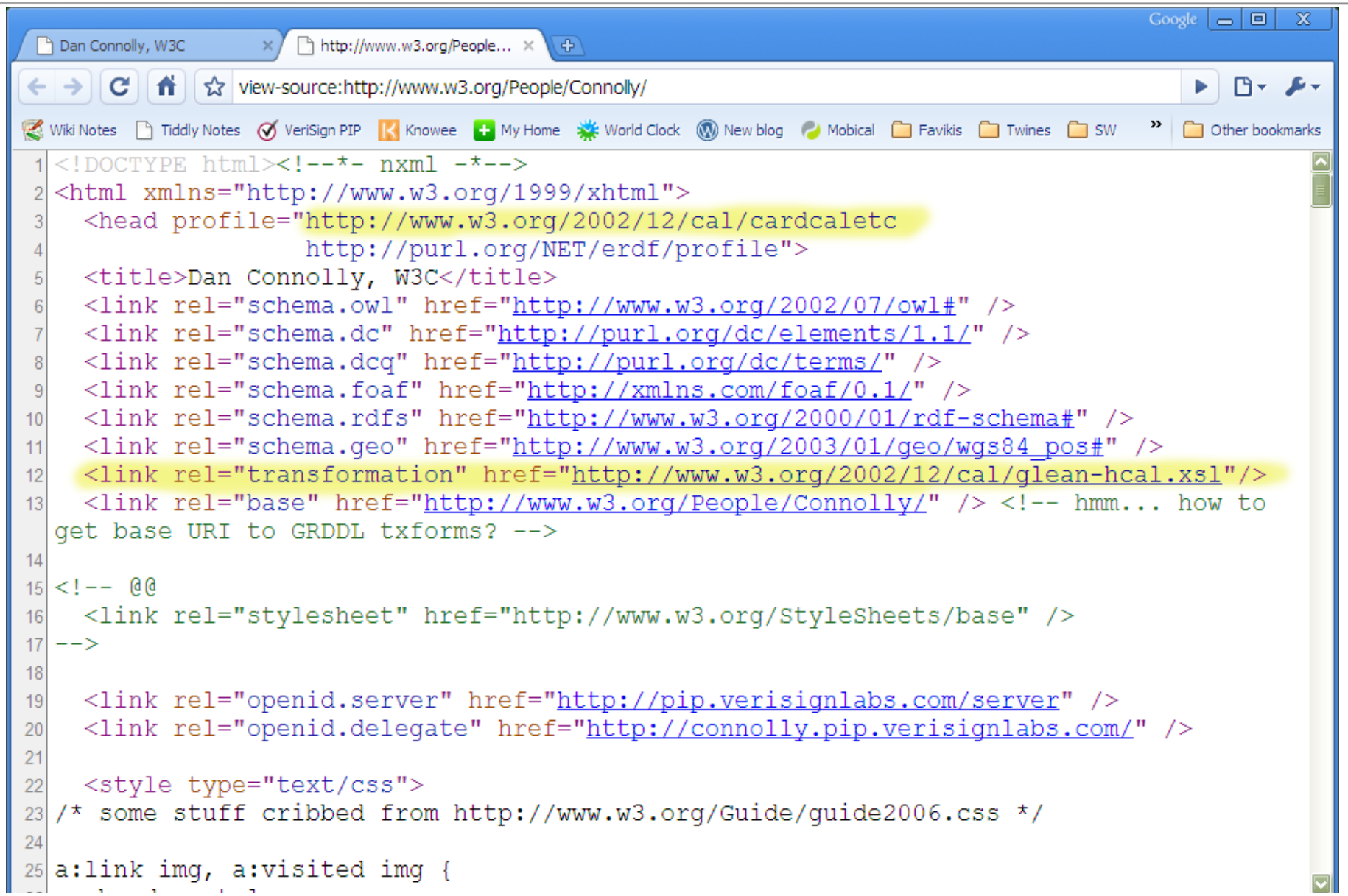
So far so good, but…

- The XSLT transformation is hCalendar specific
  - each microformat dialect needs its own
- How does a general processor find the right transformation?
- Enter GRDDL

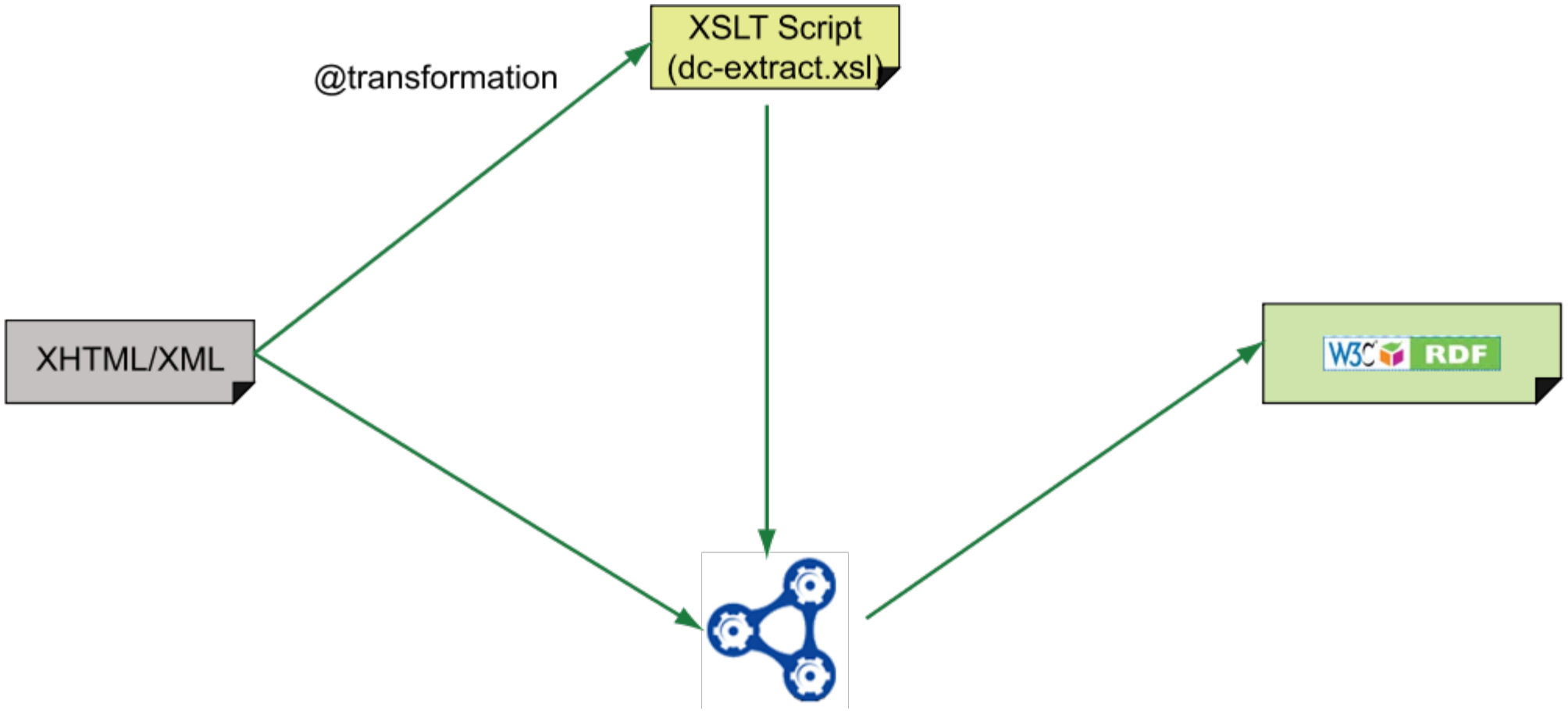
GRDDL: find the right transformation

- GRDDL defines
  - a few extra attribute values to locate the right transformation
  - a precise processing model on how the transformation should be applied to generate RDF
- Note: we describe GRDDL in terms of XHTML (and microformats) but GRDDL can be used for any XML data

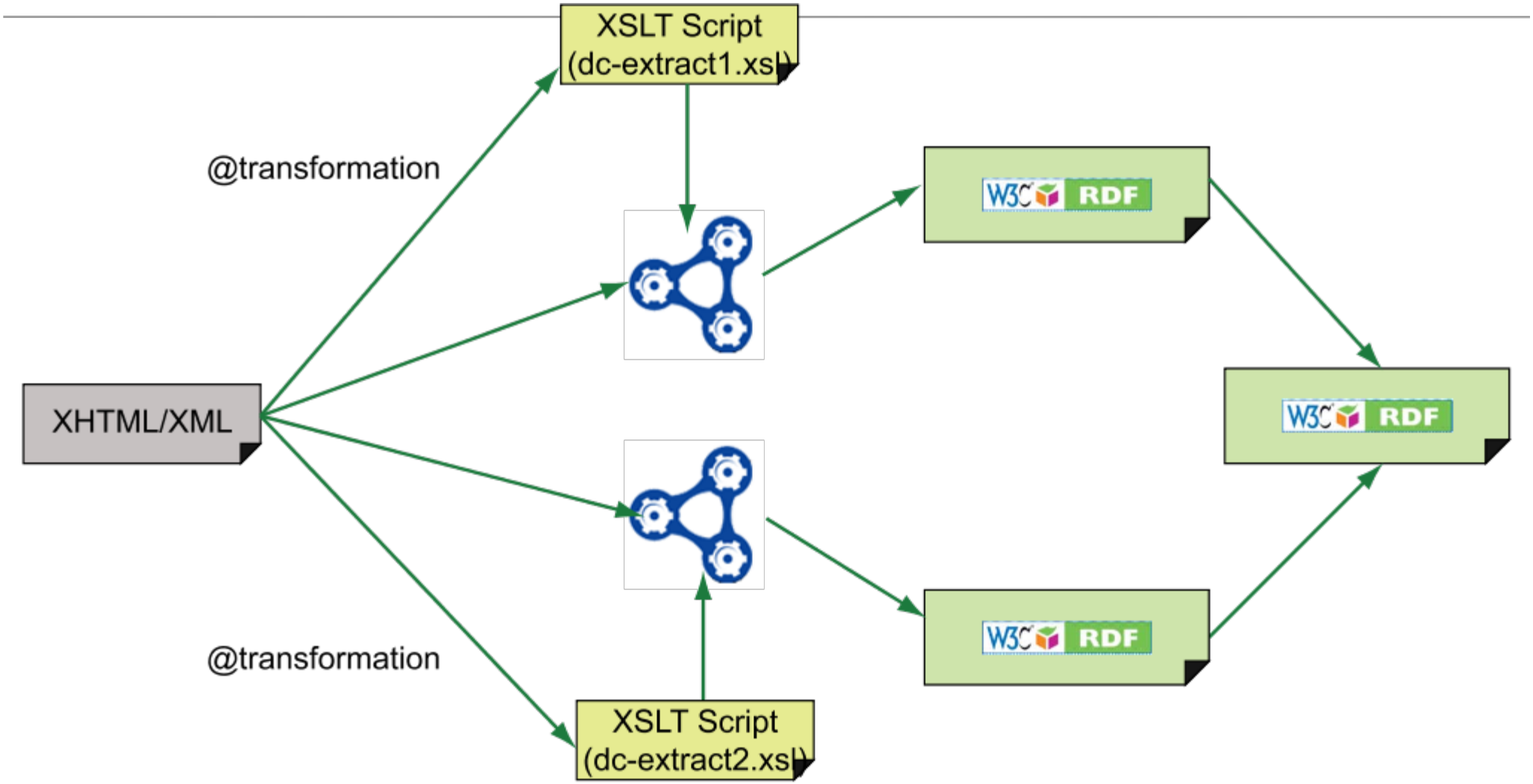
GRDDL: find the right transformation



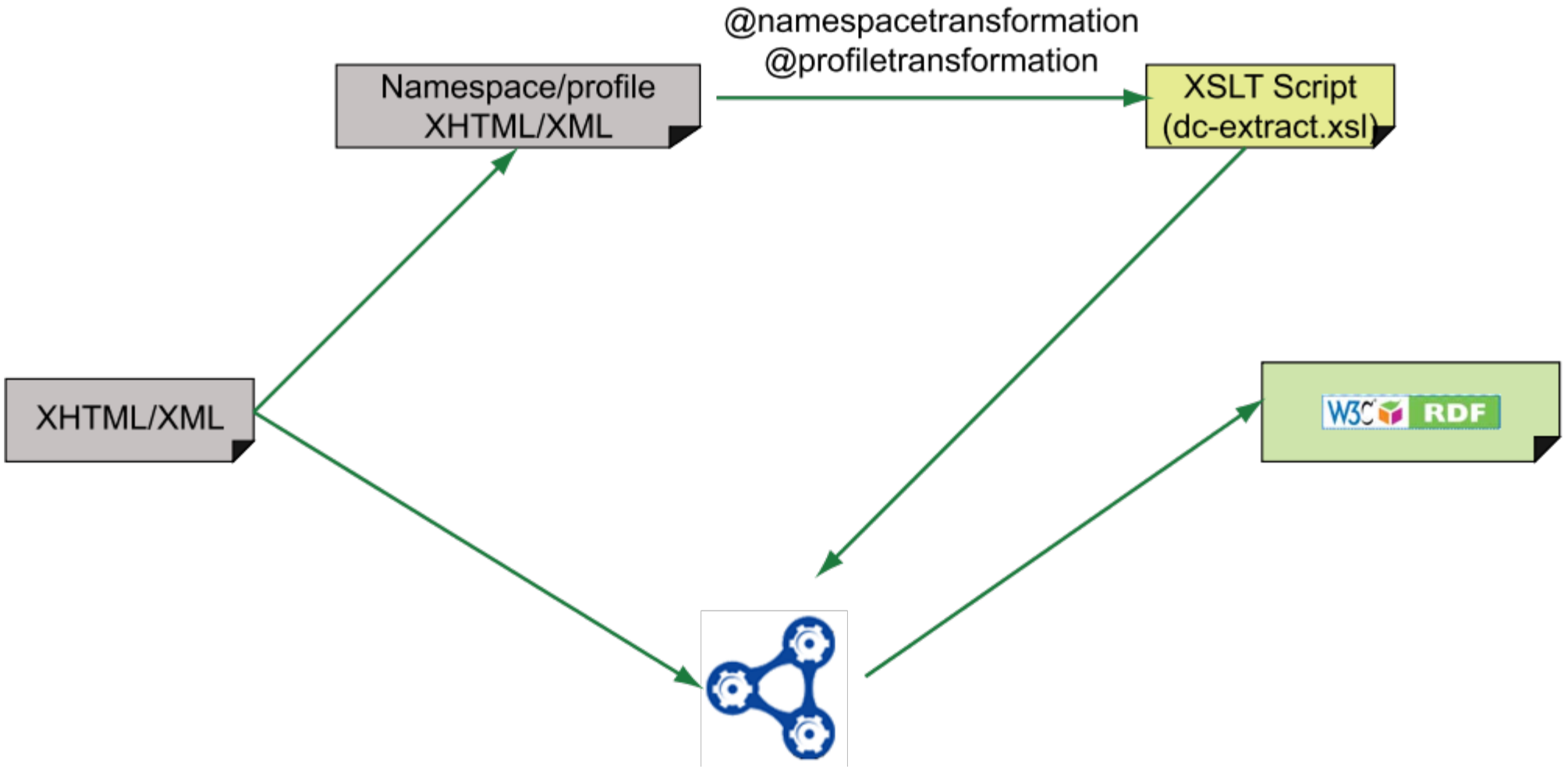
The GRDDL process: simple case



The GRDDL process: merging case



The GRDDL process: indirect case



Microformats & GRDDL: pros

- Pros:
  - simple to define/add new vocabularies
    - there is a strong microformat community for this
  - works with all current browsers, markup validators, etc
  - fairly user friendly, easy to understand and use

Microformats & GRDDL: cons

- Cons:
  - does not scale well for complex vocabularies
    - remember: needs a transformation per vocabulary
  - difficult to mix vocabularies within one page
    - what if the usage of an attribute clashes among different vocabularies?
  - the “target” RDF vocabulary is to be defined additionally to the specific microformat specification
  - some of the attributes are meant for other usage
    - eg, the abbr element, the @title attribute, …

A more general solution: microdata and RDFa

- General idea: define a set of extra attributes to add "structured data" to an HTML page
- This extra structured data can
  - be used directly by a 3rd party (e.g., to improve the user interface)
  - extracted, converted into RDF and integrate it with the rest of data out there
- Two "syntaxes" emerged at W3C:
  - RDFa (developed in general for XML languages and for HTML)
  - microdata (part of the HTML5 family of specifications)

HTML+microdata or HTML+RDFa are becoming a major source of data on the Web!

Ivan Herman

Who am I?

I graduated as mathematician at the Eötvös Loránd University of Budapest, Hungary, in 1979. After a brief scholarship at the Université Paris VI I joined the Hungarian research institute in computer science (SZTAKI) where I worked for 6 years (and turned into a computer scientist...). I left Hungary in 1986 and, after a few years in industry in Munich, Germany, I joined the Centre Mathematics and Computer Sciences (CWI) in Amsterdam where I have a tenure position since 1988. I received a PhD degree in Computer Science in 1990 at the University of Leiden, in the Netherlands. I joined the World Wide Web Consortium (W3C) Team as Head of W3C Offices in January 2001 while maintaining my position at CWI. I served as Head of Offices until June 2006, when I was asked to take the Semantic Web Activity Lead position, which is now my principal work at W3C.

Before joining W3C I worked in quite different areas (distributed and dataflow programming, language design, system programming), but I spend most of my research years in computer graphics and information visualization. I also participated in various graphics related ISO standardization activities and software developments. My "professional" home page contains a list of my publications (see also my Mendeley account), my public presentations, and details of the various projects I participated in the past. There is also a dblp entry for my publications generated automatically (although I am not sure it is complete...). (B.t.w., based on my publications, my Erdős number is ≤4...)

In my previous life (i.e., before joining W3C...) I was member of the Executive Committee of the Eurographics Association for 15 years, and I was vice-chair of the Association between 2000 and 2002. I was the co-chair of the 9th World Wide Web Conference, in Amsterdam, May 2000; since then, I have also been member of IW3C2 (International World Wide Web Conference Committee), responsible for the World Wide Web Conference series. Since autumn 2007 I am also member of SWSA (Semantic Web Science Association), the committee responsible for the International Semantic Web Conferences (better known as "ISWC") series.

Some personal data

- The Hungarian spelling of my full name is Herman Iván. Ie, my name is Ivan (well, spelled properly: Iván) and my surname is Herman (many in the Netherlands and in Germany mix it up, and use "Herman" as my name... this is aggravated by the fact that, uniquely in Europe, the Hungarian custom is to put surname first).
- Nationalities: French and Hungarian
- Gender: male
- Family: I am married and have a son, David.
- Date and city of birth: 24th February, 1955, Budapest, Hungary
- Email addresses: 'ivan' on my own ivan-herman.net domain, 'ivan' on the w3.org domain, or 'ivan.herman' on the cwi.nl domain
- (Mobile) Phone: +31-641044153
- Skype ID: ivan_herman
- I live in Amstelveen (see also geonames), the Netherlands (lat: 52.302063, long: 4.87397). This is a suburb of Amsterdam. The closest airport is Amsterdam Schiphol
- I am the administrator of the Semantic Web Activity Blog at W3C which can either be accessed directly or via its

some links

- personal homepage
- more data on me
- personal blog (RSS feed)
- homepage at W3C
- "professional" homepage
- "official" CV
- more about me
- my photos

"social" links

- facebook
- flickr
- picasa web
- linkedin
- IWIW
- tripit
- twitter
- Mendeley
- Google+
- freebase

Ivan Herman

Who am I?

I graduated as mathematician at the Eötvös Loránd University of Budapest, Hungary, in 1979. After a brief scholarship at the Université Paris VI I joined the Hungarian research institute in computer science (SZTAKI) where I worked for 6 years (and turned into a computer scientist...). I left Hungary in 1986 and, after a few years in industry in Munich, Germany, I joined the Centre Mathematics and Computer Sciences (CWI) in Amsterdam where I have a tenure position since 1988. I received a PhD degree in Computer Science in 1990 at the University of Leiden, in the Netherlands. I joined the World Wide Web Consortium (W3C) Team as Head of W3C Offices in January 2001 while maintaining my position at CWI. I served as Head of Offices until June 2006, when I was asked to take the Semantic Web Activity Lead position, which is now my principal work at W3C.

Before joining W3C I worked in quite different areas (distributed and dataflow programming, language design, system programming), but I spend most of my research years in computer graphics and information visualization. I also participated in various graphics related ISO standardization activities and software developments. My "professional" home page contains a list of my publications (see also my Mendeley account), my public presentations, and details of the various projects I participated in the past. There is also a dblp entry for my publications generated automatically (although I am not sure it is complete...). (B.t.w., based on my publications, my Erdős number is ≤4...)

In my previous life (i.e., before joining W3C...) I was member of the Executive Committee of the Eurographics Association for 15 years, and I was vice-chair of the Association between 2000 and 2002. I was the co-chair of the 9th World Wide Web Conference, in Amsterdam, May 2000; since then, I have also been member of IW3C2 (International World Wide Web Conference Committee), responsible for the World Wide Web Conference series. Since autumn 2007 I am also member of SWSA (Semantic Web Science Association), the committee responsible for the International Semantic Web Conferences (better known as "ISWC") series.

Some personal data

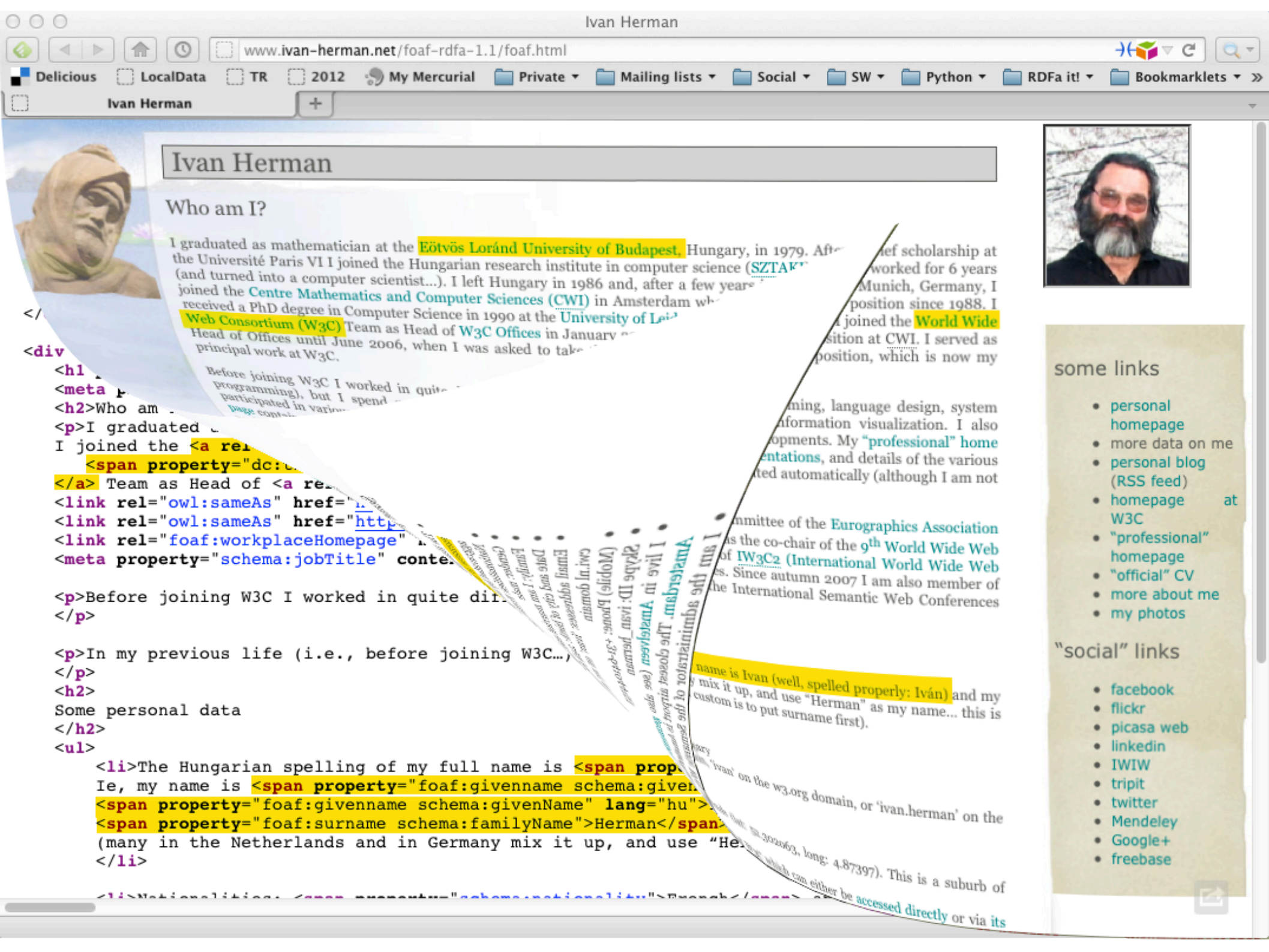
- The Hungarian spelling of my full name is Herman Iván. Ie, my name is Ivan (well, spelled properly: Iván) and my surname is Herman (many in the Netherlands and in Germany mix it up, and use "Herman" as my name... this is aggravated by the fact that, uniquely in Europe, the Hungarian custom is to put surname first).
- Nationalities: French and Hungarian
- Gender: male
- Family: I am married and have a son, David.
- Date and city of birth: 24th February, 1955, Budapest, Hungary
- Email addresses: 'ivan' on my own ivan-herman.net domain, 'ivan' on the w3.org domain, or 'ivan.herman' on the cwi.nl domain
- (Mobile) Phone: +31-641044153
- Skype ID: ivan_herman
- I live in Amstelveen (see also geonames), the Netherlands (lat: 52.302063, long: 4.87397). This is a suburb of Amsterdam. The closest airport is Amsterdam Schiphol
- I am the administrator of the Semantic Web Activity Blog at W3C which can either be accessed directly or via its

some links

- personal homepage
- more data on me
- personal blog (RSS feed)
- homepage at W3C
- "professional" homepage
- "official" CV
- more about me
- my photos

"social" links

- facebook
- flickr
- picasa web
- linkedin
- IWIW
- tripit
- twitter
- Mendeley
- Google+
- freebase


Ivan Herman
www.ivan-herman.net/foaf-rdfa-1.1/foaf.html
Delicious
LocalData
TR
2012
My Mercurial
Private
Mailing lists
Social
SW
Python
RDFa it!
Bookmarklets
Ivan Herman
Ivan Herman
Who am I?
I graduated as mathematician at the Eötvös Loránd University of Budapest, Hungary, in 1979.
the Université Paris VI I joined the Hungarian research institute in computer science (SZTA
(and turned into a computer scientist...). I left Hungary in 1986 and, after a few years
joined the Centre Mathematics and Computer Sciences (CWI) in Amsterdam
received a PhD degree in Computer Science in 1990 at the University of
Web Consortium (W3C) Team as Head of W3C Offices in January
Head of Offices until June 2006, when I was asked to take
principal work at W3C.
Before joining W3C I worked in quite
programming), but I spend
participated in
<h1
<meta
<h2>Who am
I graduated
I joined the <a rel
<span property="dc:
 Team as Head of <a re
<link rel="owl:sameAs" href="
<link rel="owl:sameAs" href="http
<link rel="foaf:workplaceHomepage"
<meta property="schema:jobTitle" conte
Before joining W3C I worked in quite dif

In my previous life (i.e., before joining W3C…)

<h2>
Some personal data
</h2>

The Hungarian spelling of my full name is <span prop
Ie, my name is <span property="foaf:givenname schema:given

Herman</span
(many in the Netherlands and in Germany mix it up, and use "He

ming, language design, system
nformation visualization. I also
opments. My "professional" home
entations, and details of the various
ted automatically (although I am not
mmittee of the Eurographics Association
as the co-chair of the 9th World Wide Web
of IW3C2 (International World Wide Web
es. Since autumn 2007 I am also member of
he International Semantic Web Conferences
name is Ivan (well, spelled properly: Iván) and my
mix it up, and use "Herman" as my name... this is
custom is to put surname first).
'ivan' on the w3.org domain, or 'ivan.herman' on the
some links
personal homepage
more data on me
personal blog (RSS feed)
homepage at W3C
"professional" homepage
"official" CV
more about me
my photos
"social" links
facebook
flickr
picasa web
linkedin
IWIW
tripit
twitter
Mendeley
Google+
freebase

```
                    <meta property="foaf:accountName" content="ivan-herman" />
                </a>
            </li>
            <li>
                <a href="https://plus.google.com/u/0/113268051484517627727" typeof="foaf:OnlineAccount">
                    <span property="foaf:accountServiceHomepage" href="http://www.mendeley.com/">Google+</span>
                    <meta property="foaf:accountName" content="113268051484517627727" />
                </a>
            </li>
            <li>
                <a about="http://www.ivan-herman.net/foaf#me" rel="owl:sameAs" resource="http://rdf.freebase.com/ns/en.
            </li>
        </ul>
      </div>
</div>

<div id="content" >
    <h1 property="schema:name foaf:name">Ivan Herman</h1>
    <meta property="foaf:title" content="Dr" />
    <h2>Who am I?</h2>
    <p>I graduated as mathematician at the <a rel="foaf:schoolHomepage schema:alumniOf" href="http://www.elte.hu/"><spa
    I joined the <a rel="schema:worksFor" href="http://www.w3.org" resource="http://www.w3.org/Data#W3C">
        <span property="dc:title">World Wide Web Consortium (W3C)</span>
    </a> Team as Head of <a rel="foaf:pastProject" href="http://www.w3.org/Consortium/Offices"><span property="dc:title"
    <link rel="owl:sameAs" href="http://www.ivan-herman.net/me" />
    <link rel="owl:sameAs" href="http://www.ivan-herman.net/Ivan_Herman" />
    <link rel="foaf:workplaceHomepage" href="http://www.w3.org"/>
    <meta property="schema:jobTitle" content="Semantic Web Activity Lead" />

    <p>Before joining W3C I worked in quite different areas (distributed and dataflow programming, language design, syst
    </p>

    <p>In my previous life (i.e., before joining W3C…) I was member of the Executive Committee of the <a rel="foaf:pastI
    </p>
    <h2>
    Some personal data
    </h2>
    <ul>
        <li>The Hungarian spelling of my full name is <span property="foaf:name" lang="hu">Herman Iván</span>.
        Ie, my name is <span property="foaf:givenname schema:givenName">Ivan</span> (well, spelled properly:
        <span property="foaf:givenname schema:givenName" lang="hu">Iván</span>) and my surname is
        <span property="foaf:surname schema:familyName">Herman</span>
        (many in the Netherlands and in Germany mix it up, and use “Herman” as my name… this is aggravated by the fact t
        </li>
```

In a slightly more readable format…

```
<html>
....
<body prefix="schema: http://schema.org/
              foaf:   http://xmlns.com/foaf/0.1/"

  <div about="http://www.ivan-herman.net/foaf#me" ... >
    ...
    <p>I graduated as mathematician at the
       <a rel="foaf:schoolHomepage schema:alumniOf"
          href="http://www.elte.hu/">
           <span property="dc:title">Eötvös Loránd University of
             Budapest</span>
       </a>, ...

    ...
```

In a slightly more readable format…

Yielding…

```
<http://www.ivan-herman.net/foaf#me>
    schema:alumniOf      <http://www.elte.hu> ;
    foaf:schoolHomePage <http://www.elte.hu> ;
    schema:worksFor      <http://www.w3.org/W3C#data> ;
    …
<http://www.elte.hu>
    dc:title "Eötvös Loránd University of Budapest" .
…
<http://www.w3.org/W3C#data>
    dc:title "World Wide Web Consortium (W3C)”
…
```

Oscars 2012: The Artist, review

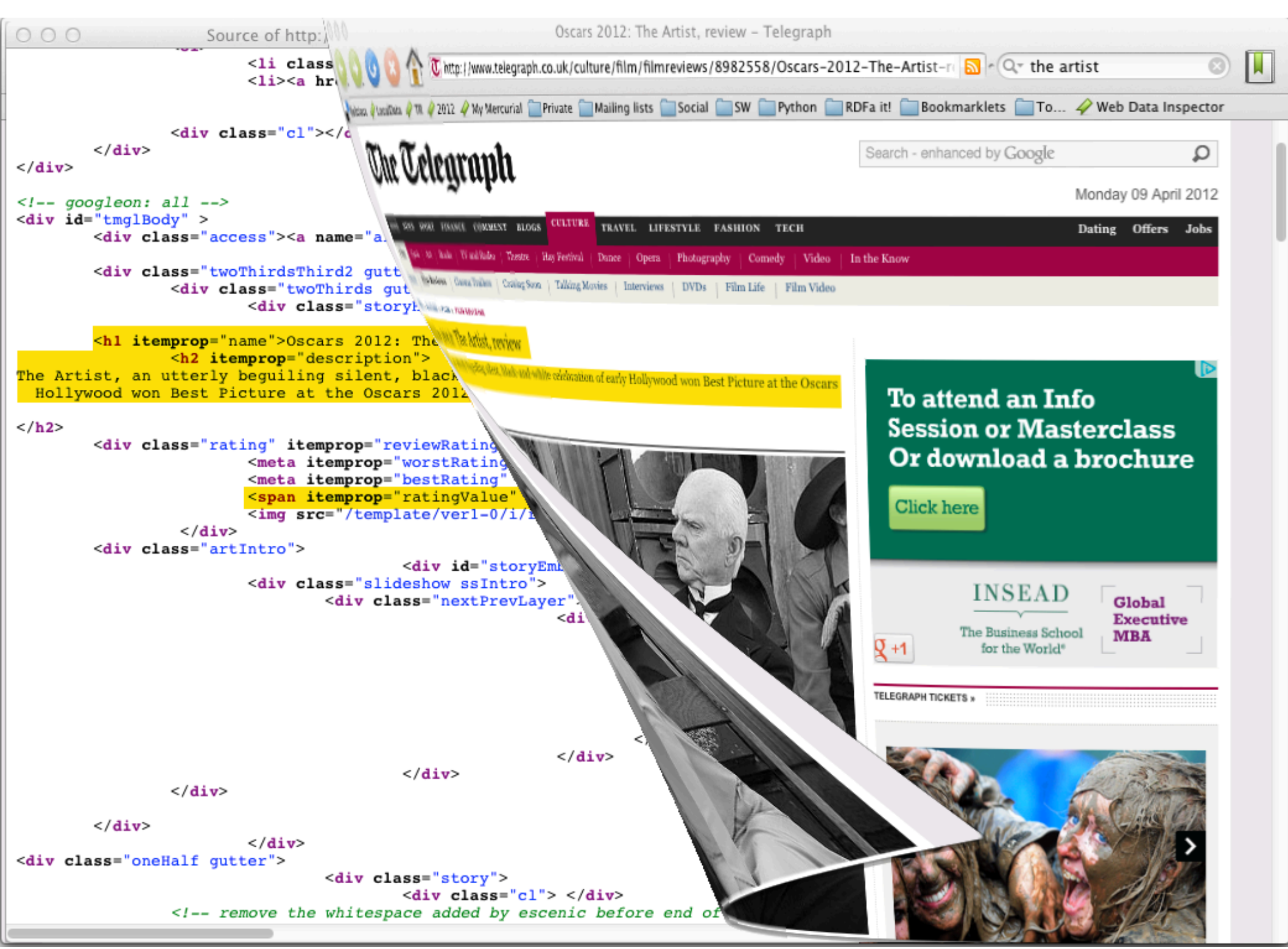
The Artist, an utterly beguiling silent, black-and-white celebration of early Hollywood won Best Picture at the Oscars 2012.

★★★★★

Oscars 2012: The Artist, review

The Artist, an utterly beguiling silent, black-and-white celebration of early Hollywood won Best Picture at the Oscars 2012.

★★★★★


Source of http:
Oscars 2012: The Artist, review - Telegraph
http://www.telegraph.co.uk/culture/film/filmreviews/8982558/Oscars-2012-The-Artist-r
the artist
Private
Mailing lists
Social
SW
Python
RDFa it!
Bookmarklets
Web Data Inspector
The Telegraph
Search - enhanced by Google
Monday 09 April 2012
CULTURE
TRAVEL
LIFESTYLE
FASHION
TECH
Dating
Offers
Jobs
Photography
Comedy
Video
In the Know
Film Life
Film Video
<h1 itemprop="name">Oscars 2012: The
<h2 itemprop="description">
The Artist, an utterly beguiling silent, black
Hollywood won Best Picture at the Oscars 2012
</h2>
<div class="rating" itemprop="reviewRating
<meta itemprop="worstRating
<meta itemprop="bestRating"
<span itemprop="ratingValue"
<img src="/template/verl-0/i/

<div id="storyEm

<div class="nextPrevLayer"
<di

<!-- remove the whitespace added by escenic before end of
To attend an Info Session or Masterclass Or download a brochure
Click here
INSEAD
The Business School for the World
Global Executive MBA
+1
TELEGRAPH TICKETS »

Source of http://www.telegraph.co.uk/culture/film/filmreviews/8982558/Oscars-2012-The-Artist-review.html

```
                    <li class="first"><a href="/">Home</a><span>&raquo;</span></li>
                    <li><a href="http://www.telegraph.co.uk/culture/">Culture</a><span>&raquo;</span></li>
                                <li><a href="http://www.telegraph.co.uk/culture/film/">Film</a><span>&raquo;</span></li>
                                <li class="styleSix"><a href="http://www.telegraph.co.uk/culture/film/filmreviews/">Film R
            <div class="cl"></div>
        </div>
</div>

<!-- googleon: all -->
<div id="tmglBody" >
        <div class="access"><a name="article"></a></div>

        <div class="twoThirdsThird2 gutterUnder">
            <div class="twoThirds gutter"  itemscope itemtype="http://schema.org/Review">
                <div class="storyHead">

        <h1 itemprop="name">Oscars 2012: The Artist, review</h1>
            <h2 itemprop="description">
The Artist, an utterly beguiling silent, black-and-white celebration of early
  Hollywood won Best Picture at the Oscars 2012.

</h2>
        <div class="rating" itemprop="reviewRating" itemscope itemtype="http://schema.org/Rating">
                    <meta itemprop="worstRating" content = "0.5">
                    <meta itemprop="bestRating" content = "5">
                    <span itemprop="ratingValue" class="hidden">5</span>
                    <img src="/template/ver1-0/i/ratings/star_5_styleSix.png" alt="5 out of 5 stars" width="73" height="14" />
             </div>
        <div class="artIntro">
                                <div id="storyEmbSlide">
                    <div class="slideshow ssIntro">
                        <div class="nextPrevLayer">
                                        <div class="ssImg">
                                                <img src="http://i.telegraph.co.uk/multimedia/archive/0209
                                                <div class="artImageExtras" >

                                                    <div class="ingCaptionCredit">
                                                        <span class="caption">Bérénice Bejo as ris
                                                </div>
                                    </div>
                                </div>
                            </div>
            </div>

        </div>
                    </div>
<div class="oneHalf gutter">
                            <div class="story">
                                <div class="cl"> </div>
            <!-- remove the whitespace added by escenic before end of </a> tag -->
```

In a slightly more readable format…

```
<div itemscope itemtype="http://schema.org/Review">
    ...
  <h1 itemprop="name">Oscars 2012: The Artist, review</h1>
  <h2 itemprop="description">The Artist, an utterly beguiling…</h2>
  ...
  <span itemprop="ratingValue" class="hidden">5</span>
  ...
```

Yielding…

```
[ rdf:type schema:Review ;
  schema:name "Oscars 2012: The Artist, review" ;
  schema:description "The Artist, an utterly beguiling…" ;
  schema:ratingValue "5" ;
  …
]
```

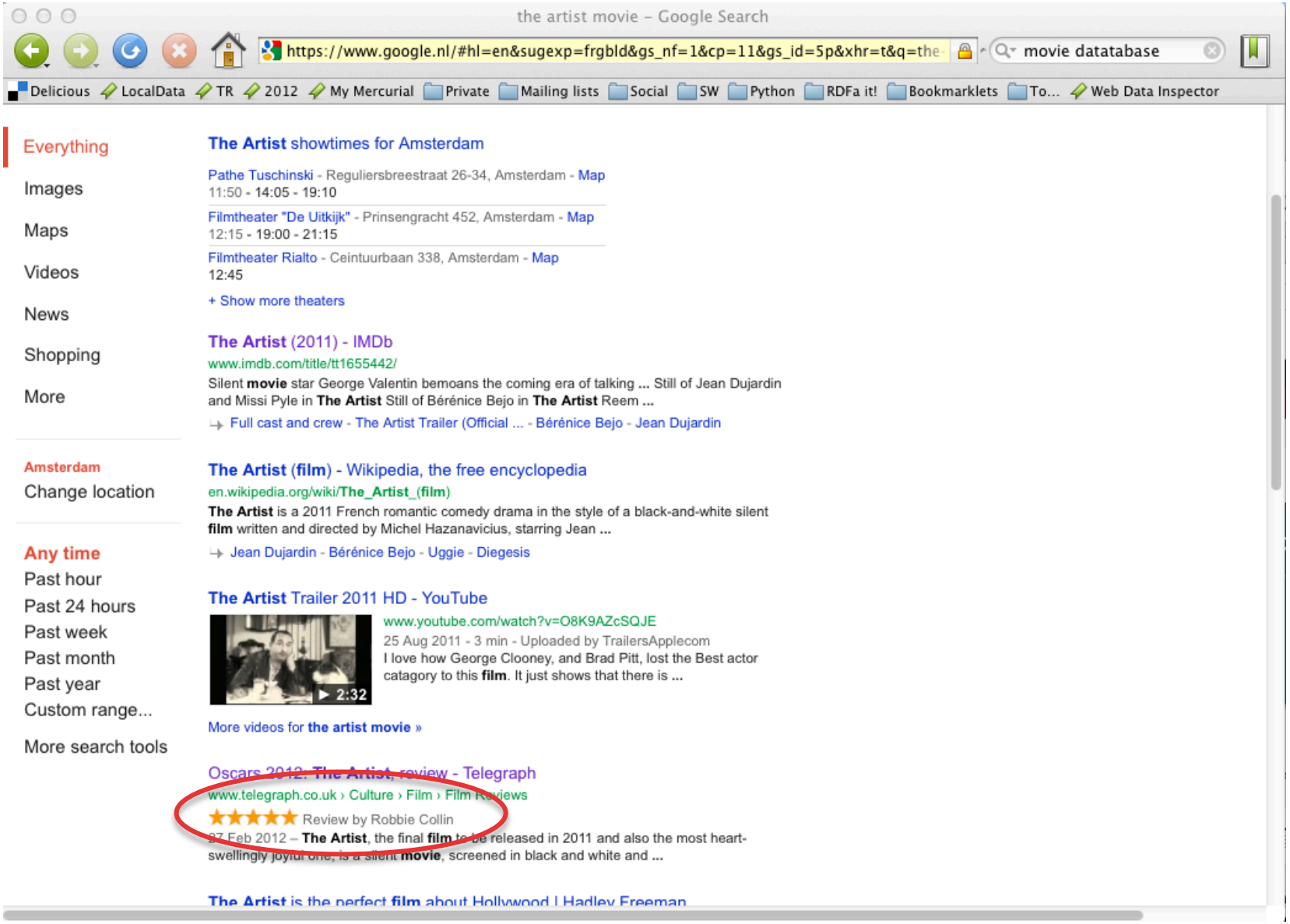
RDFa and microdata: similarities

- Both have similar philosophies:
  - the structured data is expressed *via attributes only* (no specialized elements)
  - both define some special attributes
    - e.g., **`itemscope`** for microdata, **`resource`** or **`about`** for RDFa
  - both reuse *some* HTML core attributes (e.g., **`href`**)
  - both reuse the textual content of the HTML source, if needed
- RDF data can be extracted from both

RDFa and microdata: differences

- Microdata has been *optimized* for simpler use cases:
  - one vocabulary at a time
  - tree shaped data
  - no datatypes
- RDFa provides a full serialization of RDF in XML or HTML
  - the price is an extra complexity compared to microdata
- RDFa 1.1 Lite is a simplified authoring profile of RDFa, very similar to microdata

Typical usage of structured data



Flickr: Your Photostream

http://www.flickr.com/photos/ivan_herman/

Le Palais des Miroirs

Delicious | LocalData | TR | 2012 | My Mercurial | Private | Mailing lists | Social | SW | Python | RDFa it! | Bookmarklets | To... | Web Data Inspector

flickr from YAHOO!

Signed in as Ivan Herman | Help | Sign Out

Home | You | Organize & Create | Contacts | Groups | Explore | Upload | Search

Your photostream

pro 727 items / 69,139 views

Slideshow | Share

Collections | Sets | Galleries | Tags | People | Map | Archives | Favorites | Popular | Profile

The timid spring in the Netherlands...

Click here to add a description

Anyone can see this photo (edit)

Uploaded on Mar 11, 2012 | Map | Delete

203 views / 0 comments

The timid spring in the Netherlands...

Click here to add a description

Anyone can see this photo (edit)

Uploaded on Mar 11, 2012 | Map | Delete

21 views / 0 comments

Incredible winter in Amsterdam... (Prinsengracht)

Incredible winter in Amsterdam... (Prinsengracht)

Anyone can see this photo (edit)

Uploaded on Feb 12, 2012 | Map | Delete

565 views / 0 comments

Incredible winter in Amsterdam... (Prinsengracht)

Incredible winter in Amsterdam... (Prinsengracht)

Anyone can see this photo (edit)

Uploaded on Feb 12, 2012 | Map | Delete

49 views / 0 comments

Incredible winter in Amsterdam... (Prinsengracht)

Incredible winter in Amsterdam... (Prinsengracht)

Anyone can see this photo (edit)

Uploaded on Feb 12, 2012 | Map | Delete

46 views / 0 comments

Incredible winter in Amsterdam... (Prinsengracht)

Incredible winter in Amsterdam... (Prinsengracht)

Anyone can see this photo (edit)

Uploaded on Feb 12, 2012 | Map | Delete

41 views / 1 comment

How to "assign" RDF data to resources?

How to “assign” RDF data to resources?

- This is important when the RDF data is used as “metadata”
- Some examples:
  - copyright information for your photographs
  - is a Web page usable on a mobile phone and how?
  - bibliographical data for a publication
  - annotation of the data resulting from a scientific experiment
  - etc

If I know the URI of the resource (photograph, publication, etc), how do I find the relevant RDF data?

The data might be embedded

- Some data formats allow the direct inclusion of (RDF) metadata:
  - SVG (Scalable Vector Graphics)
    - direct inclusion of RDF/XML
    - via RDFa attributes
  - HTML with RDFa
  - JPG files using the comment area and, eg, Adobe's XMP technology

Simple linkage

- Some formats have special link statements. Eg, in (X)HTML:

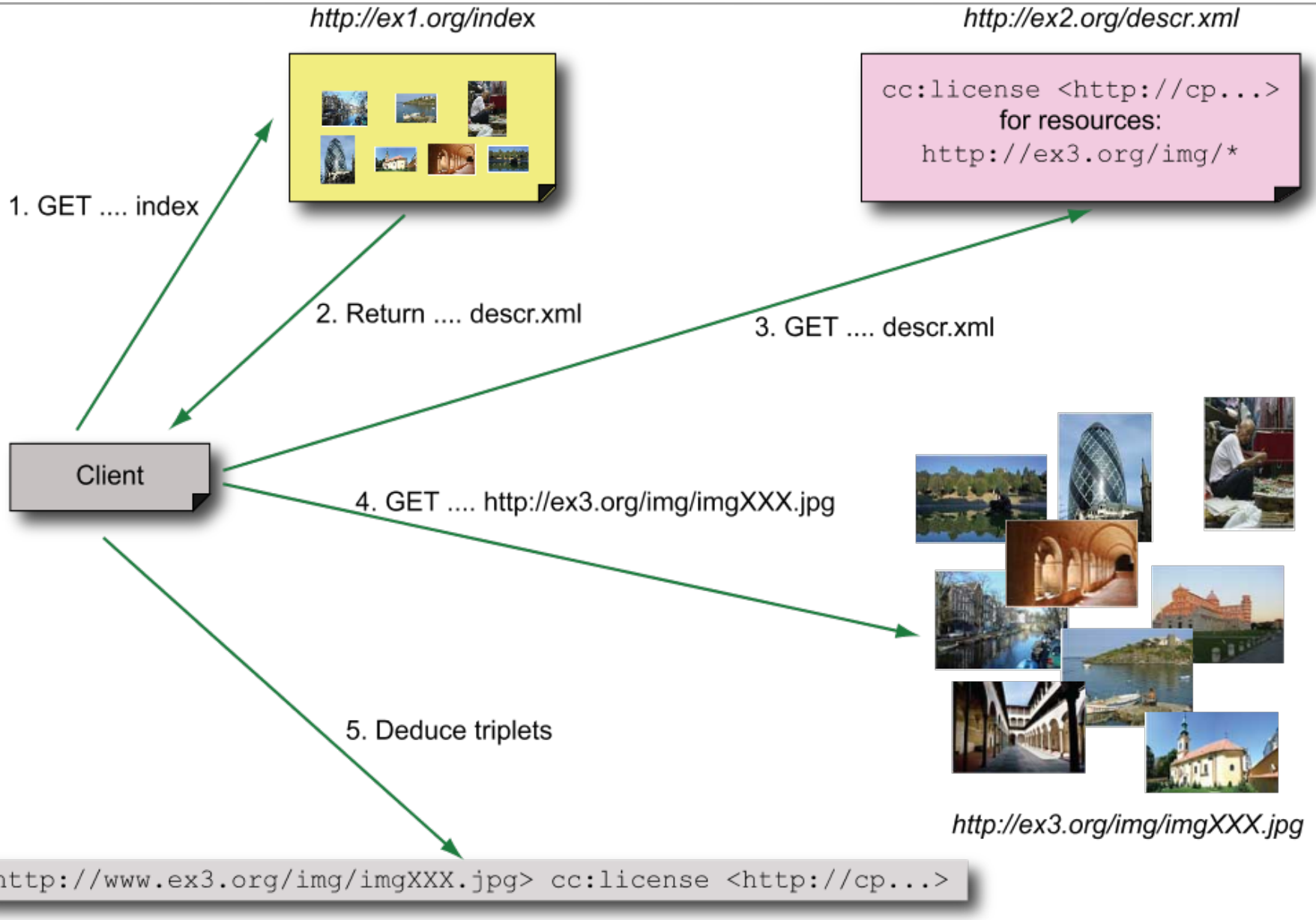
```
<html>
  <head>
    <link rel="meta" href="meta.rdf"/>
 ...
```

- Similar facilities might exist in other formats (eg, SMIL)

POWDER

- POWDER provides for a more elaborate scenarios:
  - define a set of resources by constraints on the URIs; eg
    - URIs must begin with http://www.example.com/bla/
    - the port number in the URI-s should be XYZW
  - define “description resources” that bind those resources to additional information
  - get such description resources, eg, via a link statements, via HTTP, via SPARQL, …
- Use cases: licensing information, mobileOK (and other) trustmarks, finding accessible Web sites, content labeling (eg, child protection), …

A POWDER scenario: copyright for photos



The gory details…

- The “description resource” is an XML file:

```
<powder xmlns="http://www.w3.org/2007/05/powder#"
        xmlns:cc="http://creativecommons.org/ns#">
  <attribution>
    <issuedby src="http://www.ivan-herman.net/me"/>
  </attribution>
  <dr>
    <iriset>
      <includehosts>www.ex2.org</includehost>
      <includepathstartswith>/img/</includepathstartswith>
    </iriset>
    <descriptorset>
      <cc:license rdf:resource="http://..."/>
    </descriptorset>
  </dr>
```

The gory details…

- Powder processors may then return
  - direct RDF triples, eg:

```
<http://www.ex2.org/img/imgXXX.jpg> cc:license <http://...>.
```

- or can transform this XML file into an RDF (OWL) for more generic processors
  - a canonical processing of the XML file is defined by the POWDER specification

POWDER Service

- Online POWDER service can be set up:
  - a Web service with
    - submit a URI and a resource description file
    - return the RDF statements for that URI
  - such service should be set up, eg, at W3C
- A GRDDL transform is also defined

But there is a hidden “hiccup”

- RDF makes a strong separation between
  - URI as an ID for a resource
  - URI as a datatype (xsd:anyURI)
  - there is no “bridge” between the two
- POWDER includes a small extension to the formal semantics of RDF for two properties:
  - wdrs:matchregex and wdrs:notmatchregex such that
    - (R wdrs:matchregex Regex) holds iff the URI of R matches Regex

If you want the OWL version…

```
<> wdrs:issuedBy <http://www.ivan-herman.net/me> .

_:iriset_1 a owl:Class; owl:intersectionOf (
  [ a owl:Restriction;
    owl:onProperty wsdr:matchregex ;
    owl:hasValue "..ugly regex for ex2.org"^^xsd:string ]
  [ a owl:Restriction;
    owl:onProperty wsdr:matchregex ;
    owl:hasValue "..ugly regex for /img"^^xsd:string ]
  ).

_:desc_1 a owl:Restriction;
  owl:onProperty cc:license;
  owl:hasValue <http://...>.

_:iriset_1 rdfs:subClassOf _:desc_1 .
```

Consequences of the “hiccup”

- In practice this means that only “POWDER aware” agents can fully handle the description files
  - note that the extension is fairly easy to add, so it is not a big implementation issue…
- Existence of the services to provide the triplets automatically relieve the pain…

Other POWDER features

- There are a number of additional features:
  - built in authentication: description resources must be attributed and this is open for authentication
  - assignments may carry validity dates
  - packaging several resource descriptions in one, possibly control their processing order
  - using tags to identify resources instead of URI patterns

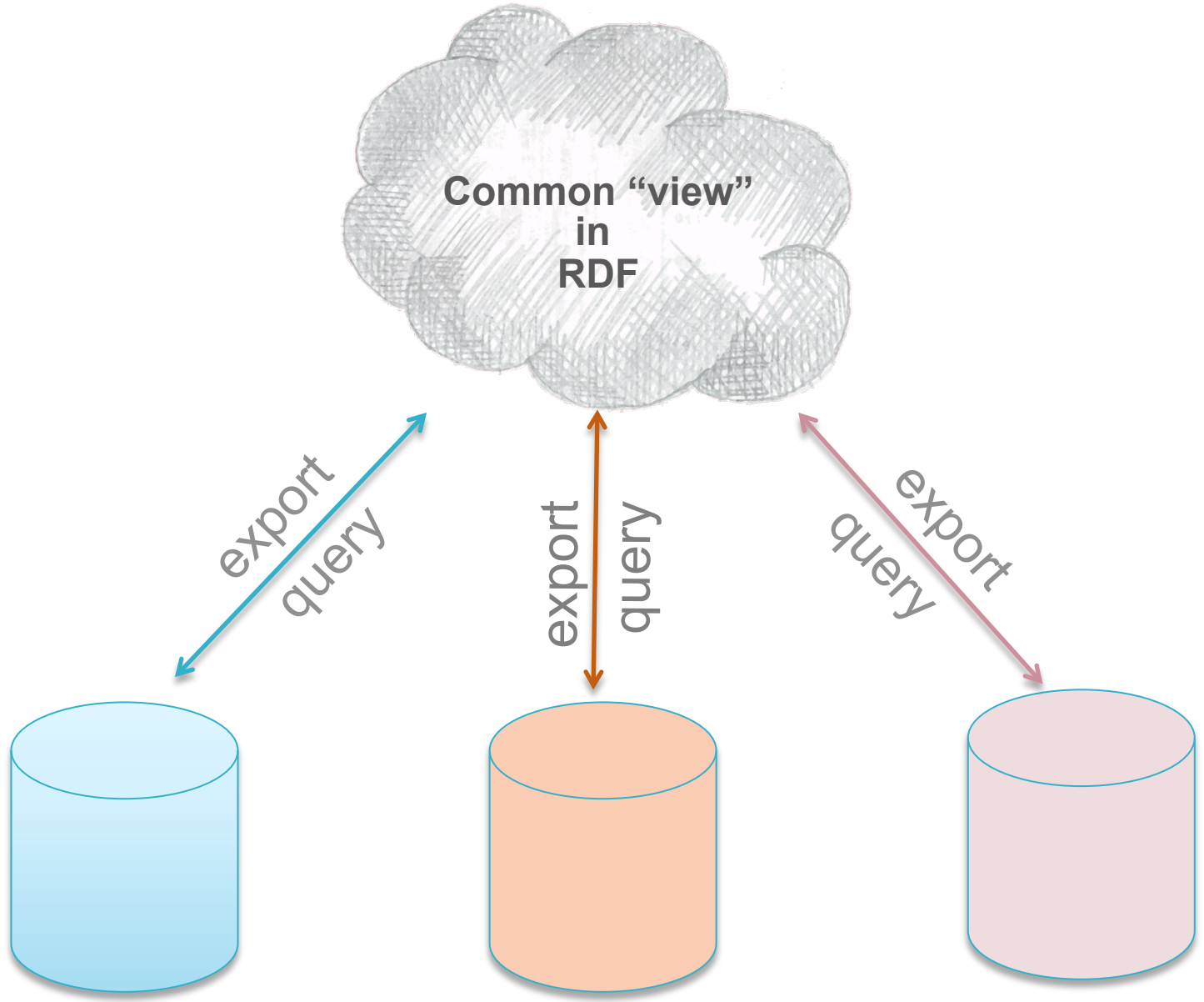
Access to Relational Databases (Direct Mapping, R2RML)

Photo credit "mayhem", Flickr

Relational Databases and RDF

- Most of the data on the Web is, in fact, in RDB-s
- Proven technology, huge systems, many vendors…
- Data integration on the Web must provide access to RDB-s

RDF provides a common “view”



What is “export”?

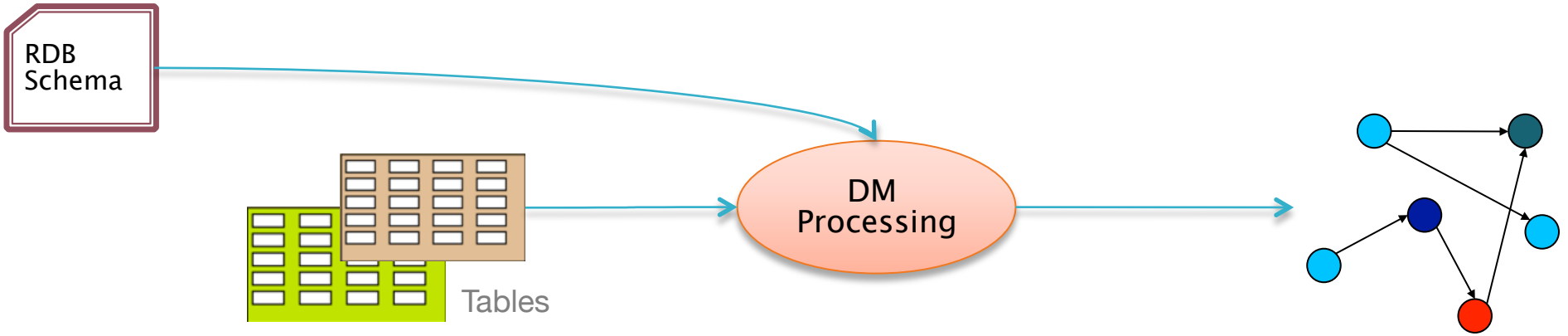
- “Export” does not *necessarily* mean physical conversion
  - for very large databases a “duplication” would not be an option
  - systems may provide SQL “bridges” to make queries on the fly
- Result of export is a “logical” view of the RDB content

Simple export: Direct Mapping

- A standard RDF “view” of RDB tables
- Valid for all RDB-s, independently of the RDB schema
- Fundamental approach:
  - each row is turned into a series of triples with a common subject
  - column names provide the predicate names
  - cell contents are the objects as literals
  - linked tables are expressed with URI subjects

What DM processor does

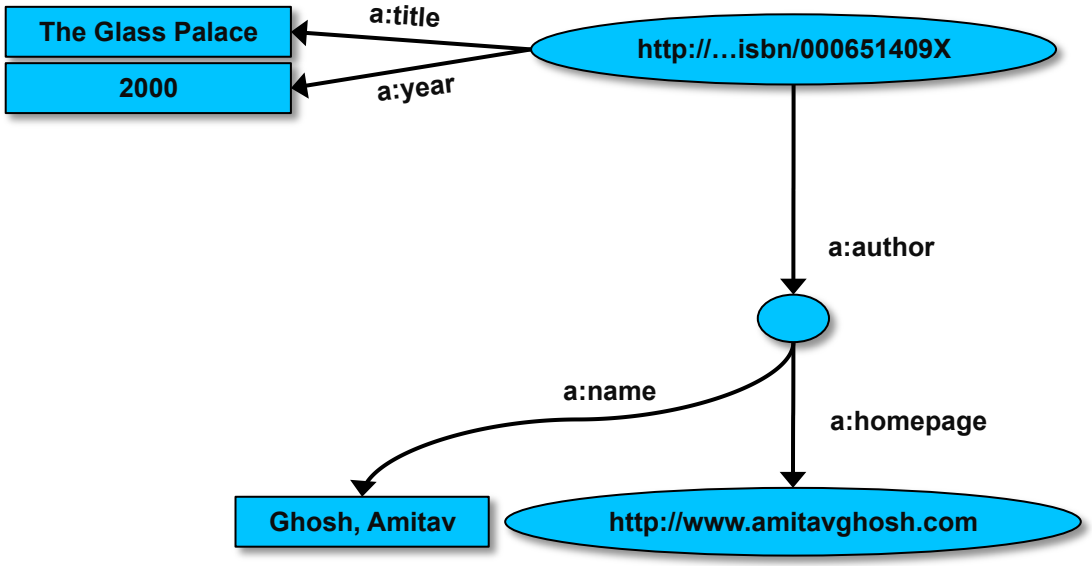
- An DM processor has access to:
  - an RDB schema
  - a database governed by the schema
- … and produces an RDF graph



Back to the bookshop example

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

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



Direct mapping of the bookshop tables

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

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

Direct mapping of the bookshop tables

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

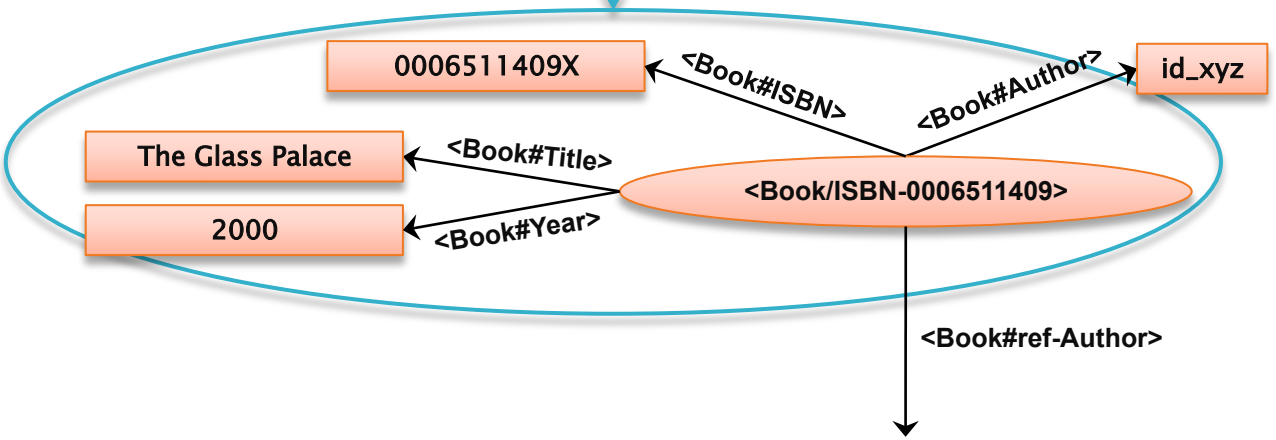
ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com

- “ISBN” and “ID” are keys
- Book table refers to the author table

Direct mapping of the bookshop tables

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

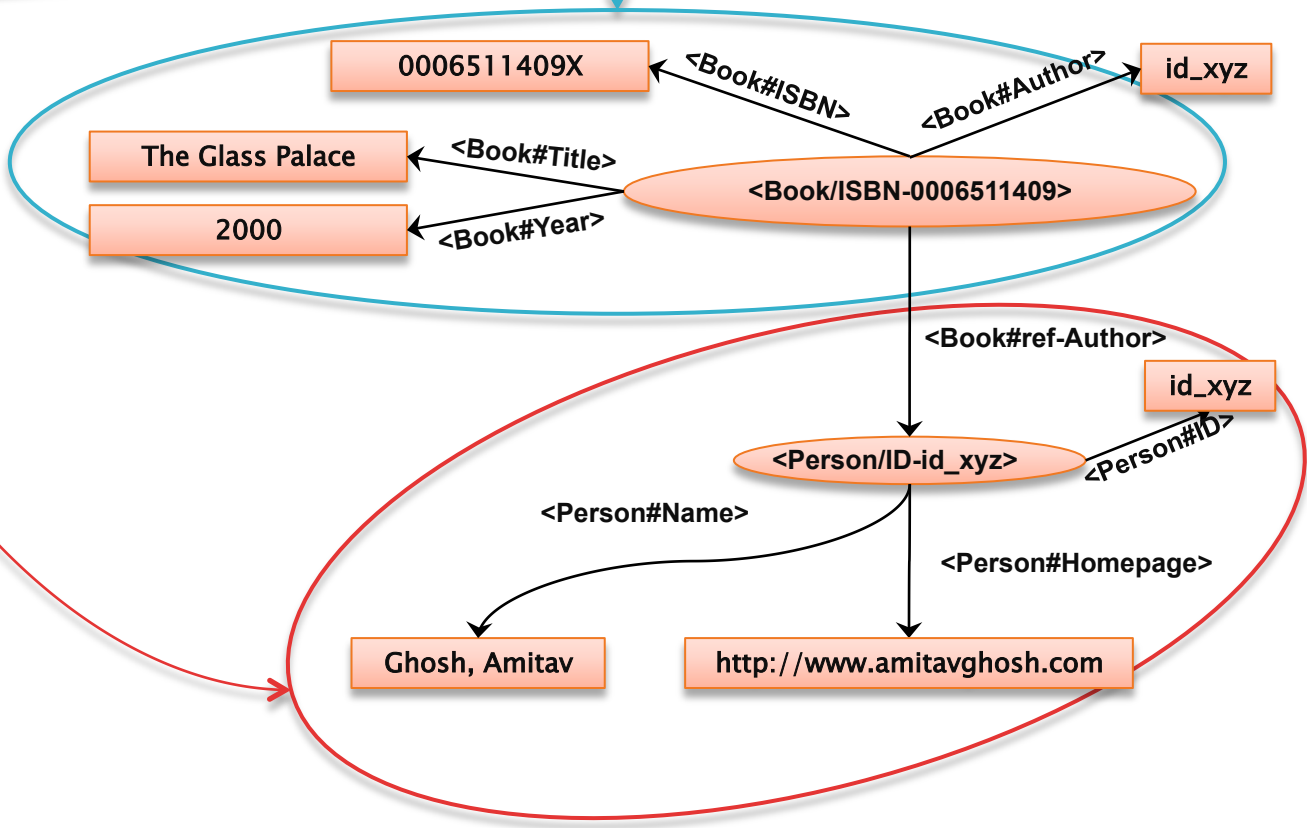
ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com



Direct mapping of the bookshop tables

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

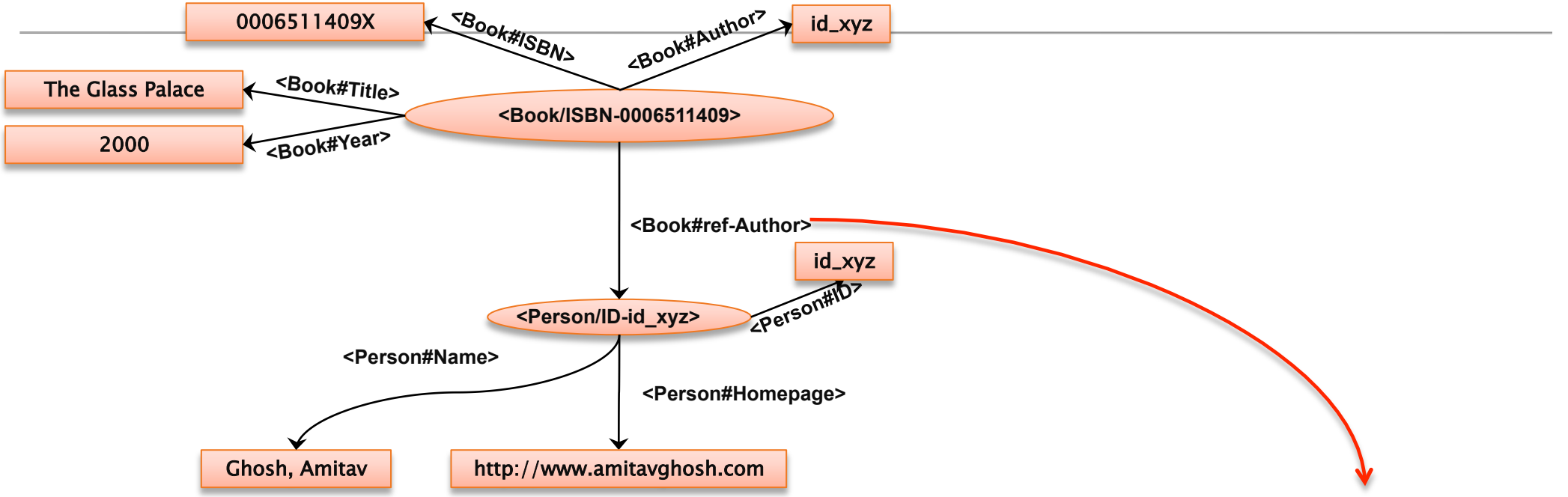
ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com



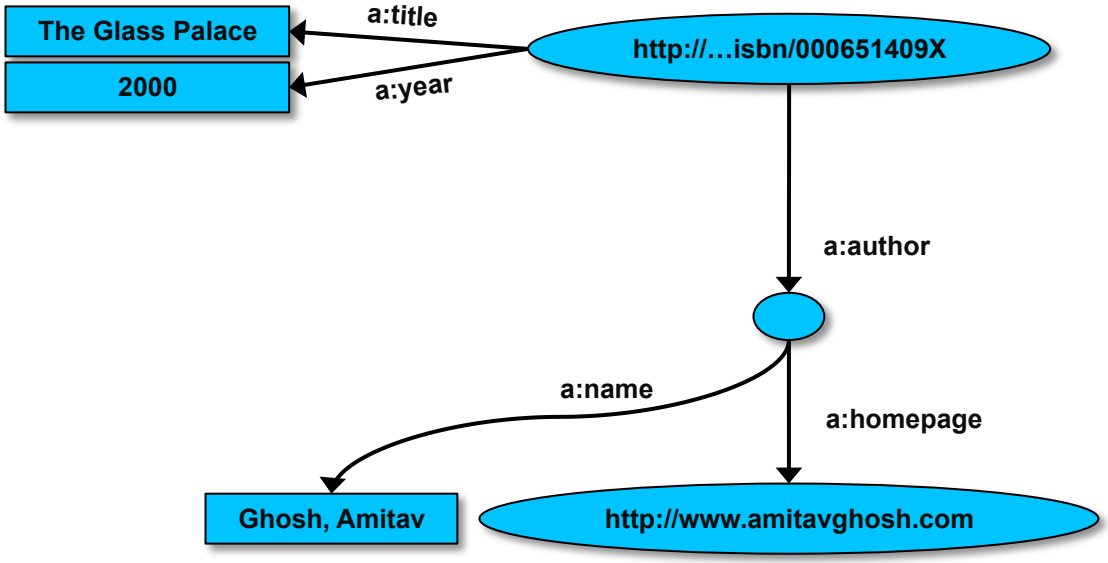
Result of the Direct Mapping

- What do we have?
  - we have an RDF “view” of the two tables
- What do we miss?
  - an RDF view that is close to our application; a more “natural” view of the book data

Direct graph must be transformed



- Property names should be mapped
- URI-s should be minted
- Literals should be replaced by URI-s



Pros and cons of Direct Mapping

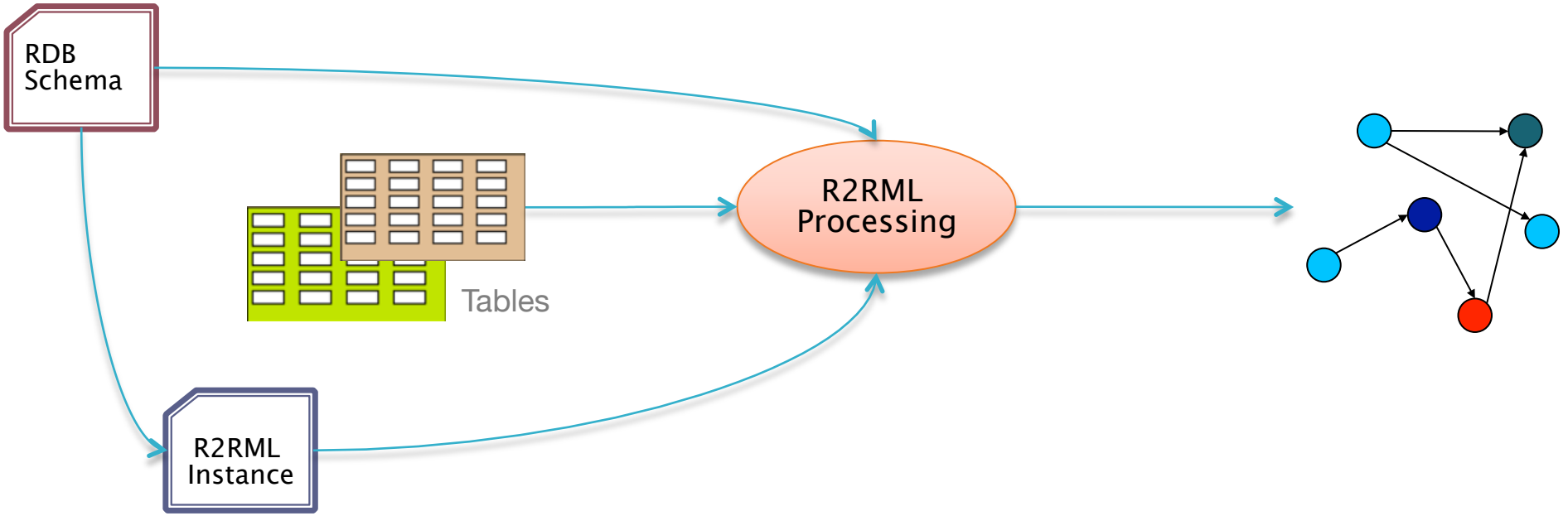
- Pros:
  - Direct Mapping is simple, does not require any other concepts
  - know the Schema ⇒ know the RDF graph structure
  - know the RDF graph structure ⇒ good idea of the Schema(!)
- Cons:
  - relies on other tools to get the “final” RDF graph

Enters R2RML

- Separate vocabulary to control the mapping
- Gets to the final RDF graph with one processing step
- Fundamentals are similar:
  - each row is turned into a series of triples with a common subject
- But:
  - cell contents *may* become a resource (not a literal)
  - there is a control over the generated URI-s
- Direct mapping is a “default” R2RML mapping

What R2RML processor does

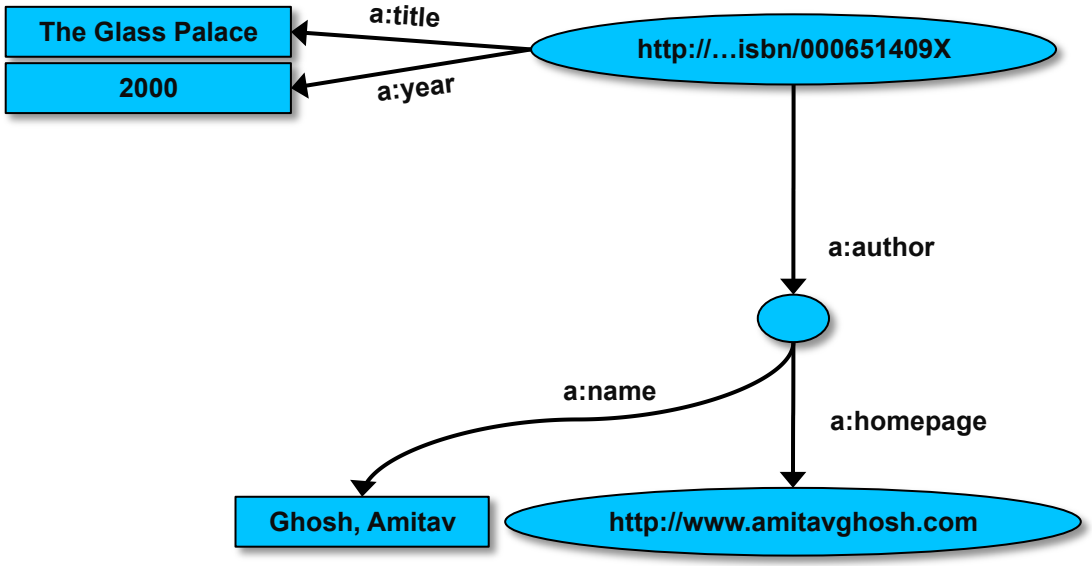
- An R2RML processor has access to:
  - an RDB schema
  - an R2RML instance
  - a database governed by the schema
- … and produces an RDF graph



Back to the bookshop example

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

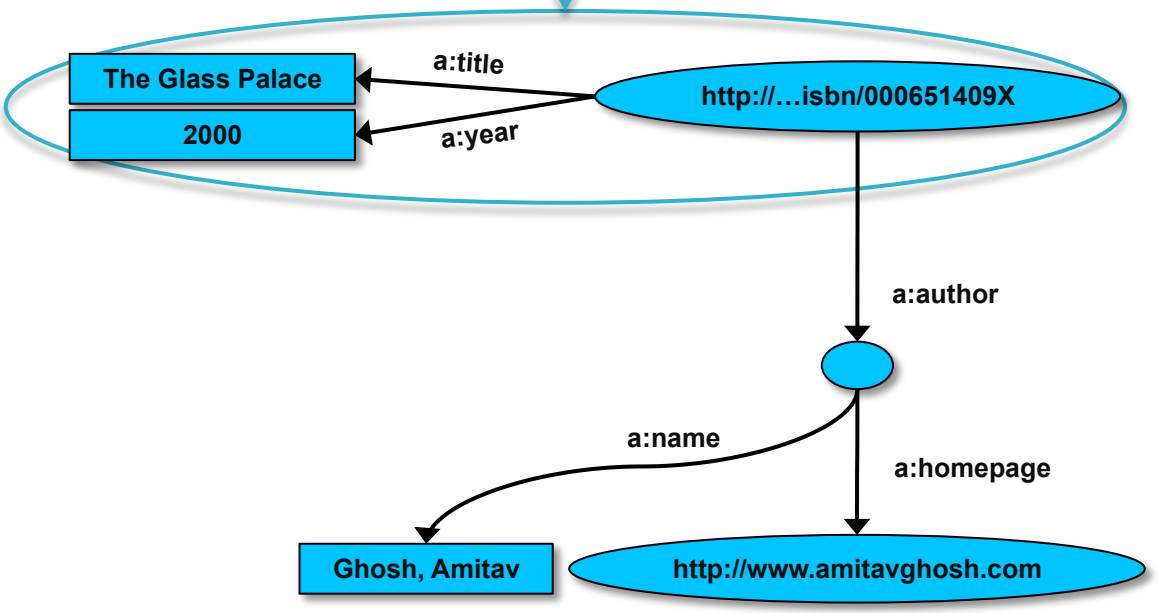
ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com



Back to the bookshop example

ISBN	Author	Title	Publisher	Year
000651409X	id_xyz	The Glass Palace	id_qpr	2000

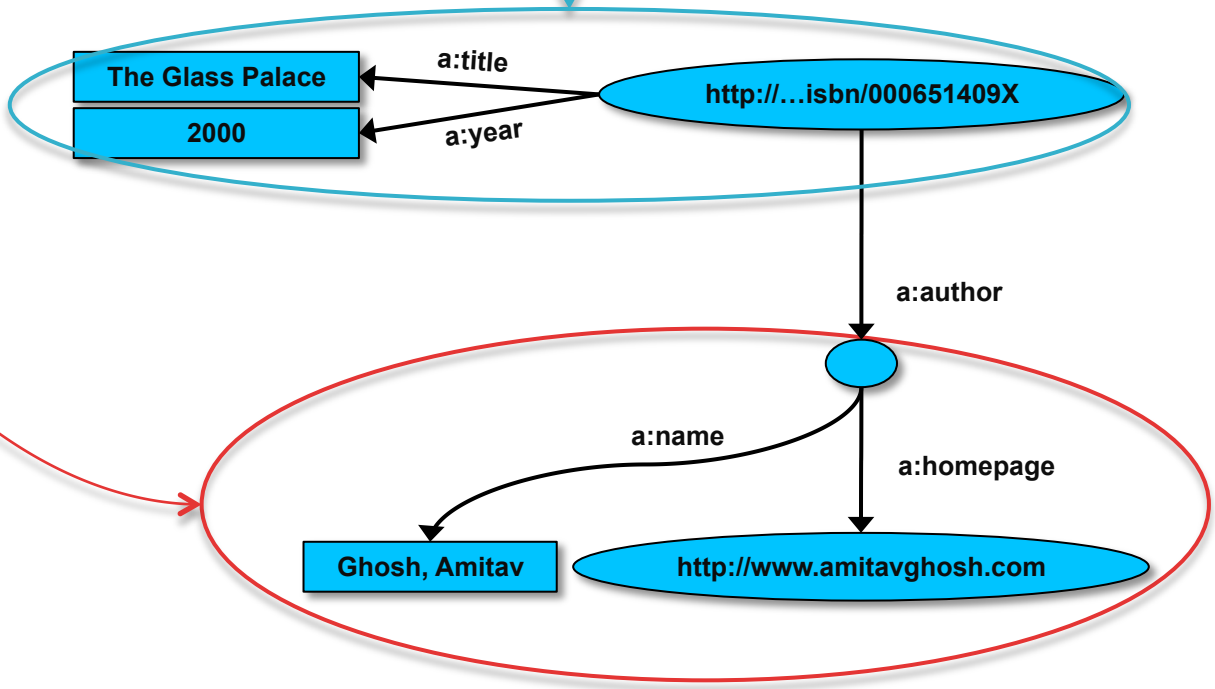
ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com



Back to the bookshop example

ISBN	Author	Title	Publisher	Year
000651409X	id_xyz	The Glass Palace	id_qpr	2000

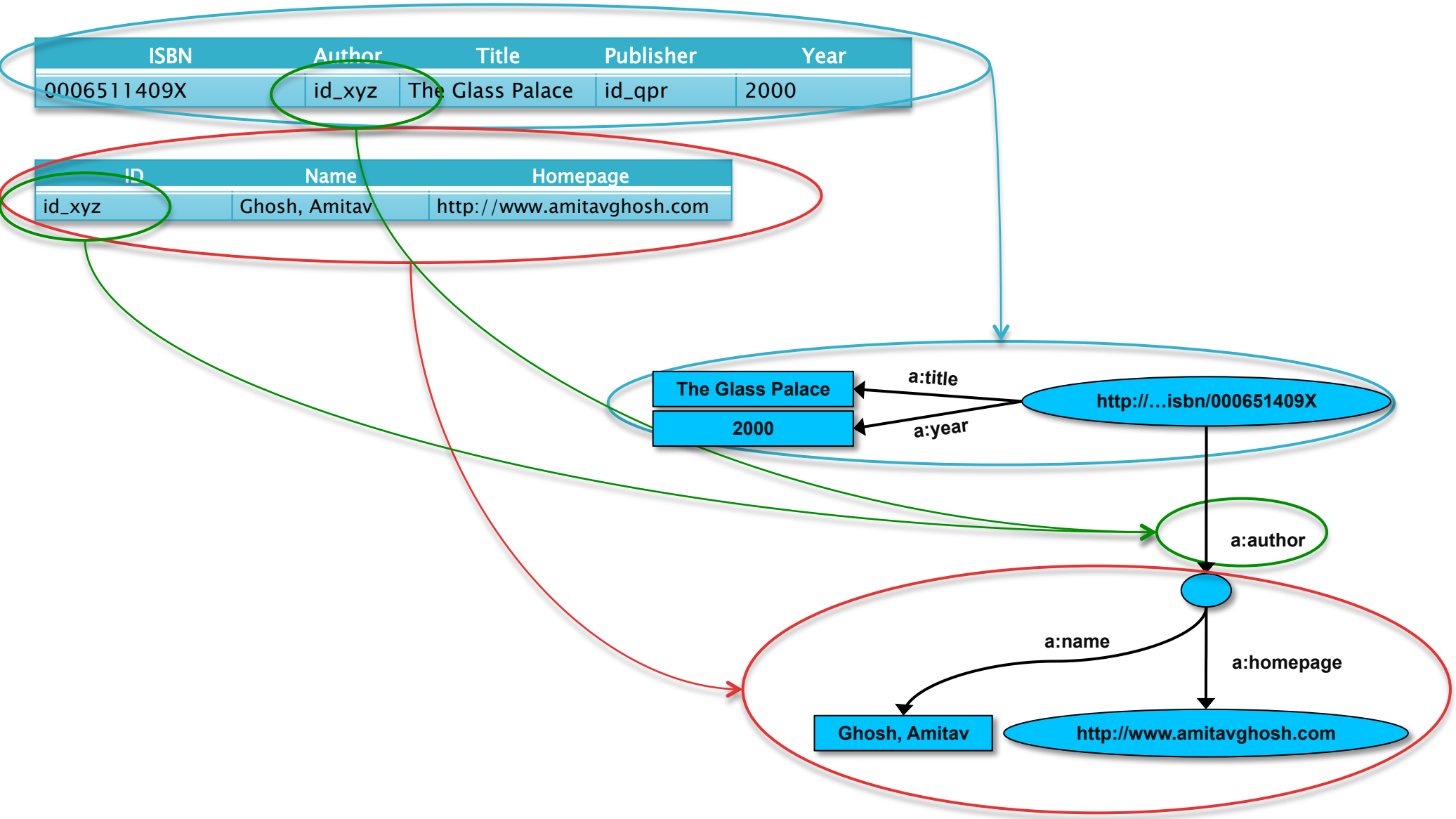
ID	Name	Homepage
id_xyz	Ghosh, Amitav	http://www.amitavghosh.com



Back to the bookshop example

ISBN	Author	Title	Publisher	Year
000651409X	id_xyz	The Glass Palace	id_qpr	2000

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

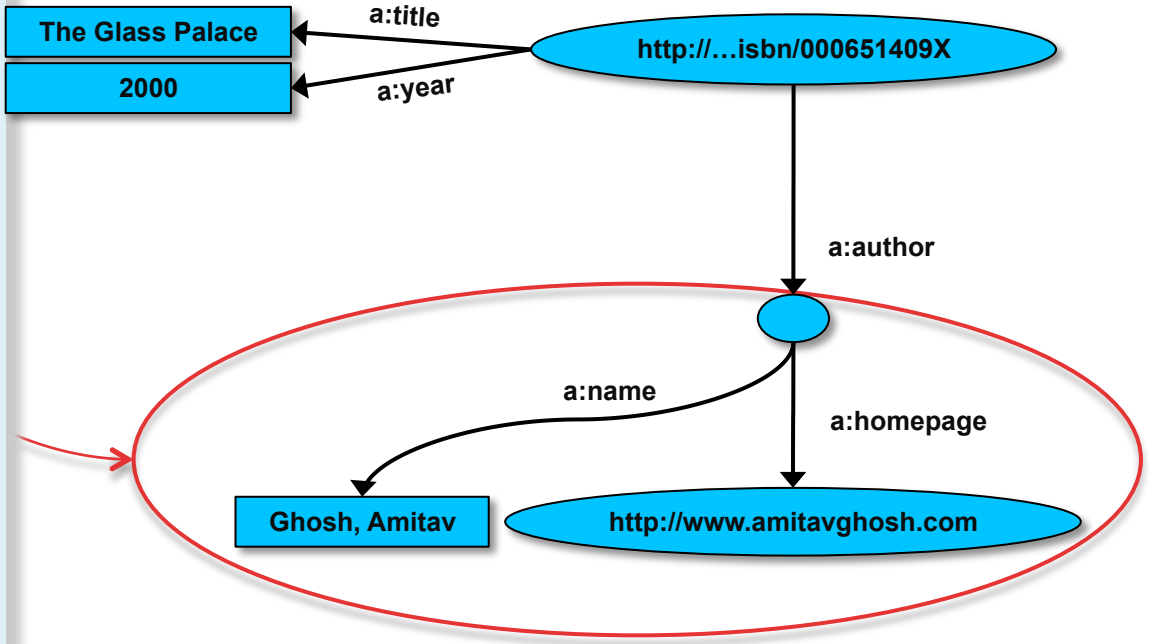


Step 1: transform “Person Table”

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

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

```
<Person>
  rr:tableName "Person_Table" ;
  rr:subjectMap [
    rr:termtype rr:BlankNode ;
  ] ;
  rr:predicateObjectMap [
    rr:predicate a:name
    rr:objectMap [
      rr:column "Name"
    ]
  ] ;
  rr:predicateObjectMap [
    rr:predicate a:homepage ;
    rr:objectMap [
      rr:column "Homepage" ;
      rr:termtype "IRI"
    ]
  ] ;
```

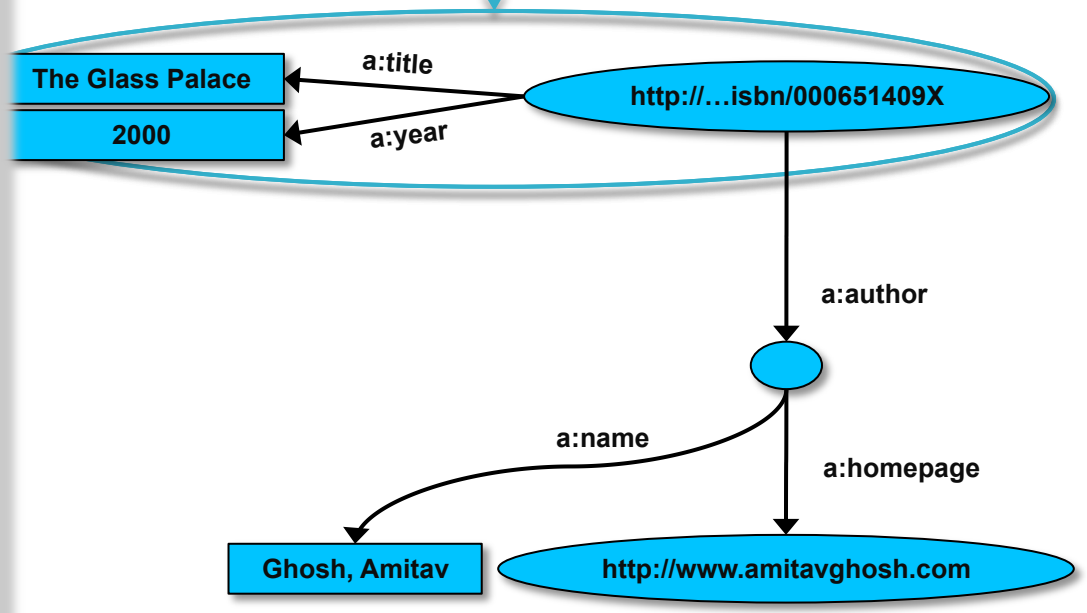


Step 2: transform “Book Table”

ISBN	Author	Title	Publisher	Year
000651409X	id_xyz	The Glass Palace	id_qpr	2000

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

```
<Book>
  rr:tableName "Book_Table" ;
  rr:subjectMap [
    rr:template "http://...isbn/{ISBN}" ;
  ];
  rr:predicateObjectMap [
    rr:predicate a:title ;
    rr:objectMap [
      rr:column "Title"
    ]
  ] ;
  rr:predicateObjectMap [
    rr:predicate a:year ;
    rr:objectMap [
      rr:column "Year" ;
    ]
  ] ;
```

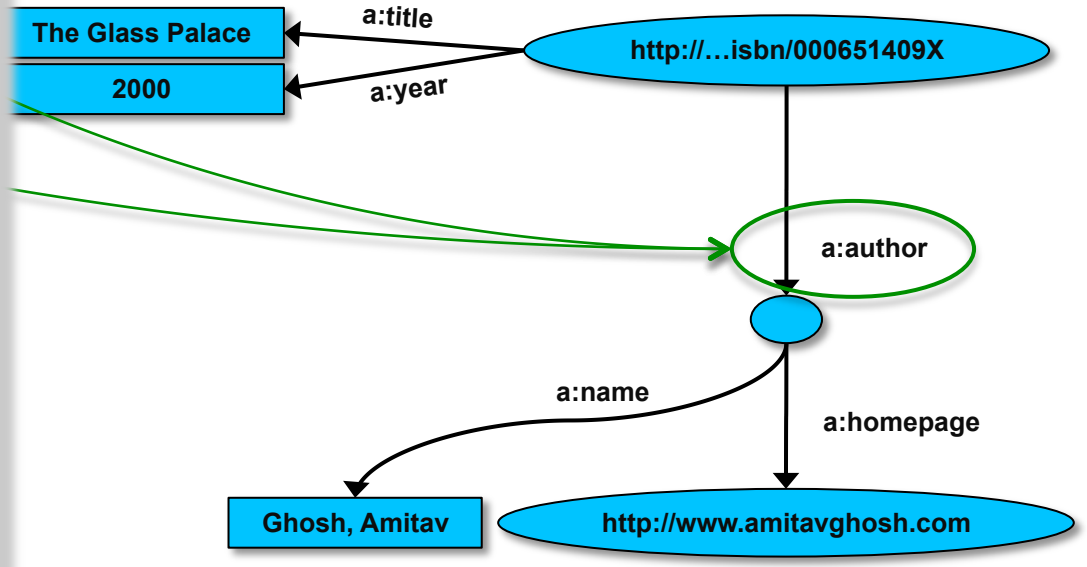


Step 3: “bind” the two tables

ISBN	Author	Title	Publisher	Year
0006511409X	id_xyz	The Glass Palace	id_qpr	2000

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

```
<Book>
  ...
  rr:refPredicateObjectMap [
    rr:predicate a:author ;
    rr:objectMap [
      rr:parentTriplesMap <Person> ;
      rr:joinCondition
        rr:child "Author" ;
        rr:parent "ID”
      ]
    ]
  ].
```



Further R2RML features

- There are some additional features
  - assign a datatype to a literal object
  - more complicated object assignments (e.g., for a specific column the object is a cell of *another* column)

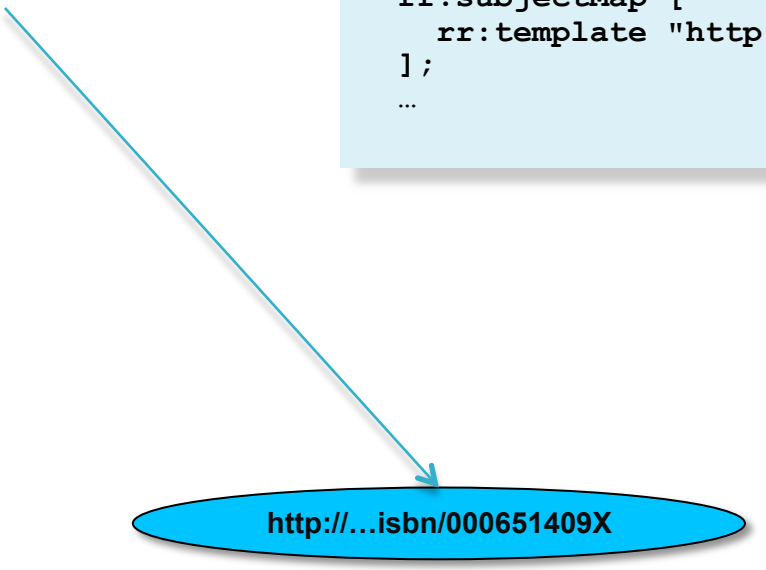
```
<Book>
  ...
  rr:predicateObjectMap [
    rr:objectMap [
      rr:column "Year" ;
      rr:datatype xsd:year
    ]
    ...
  ] ;
```

Further R2RML features: logical table

- Back to our example:

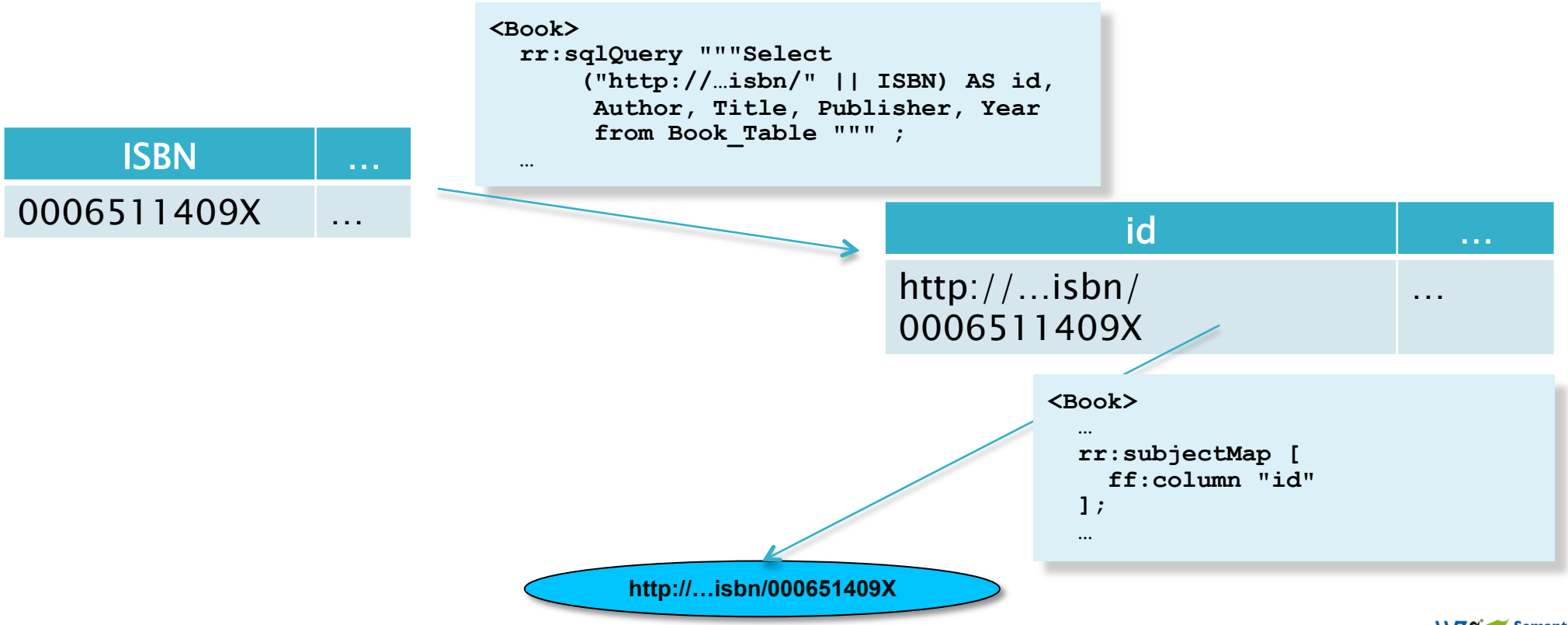
ISBN	…
0006511409X	…

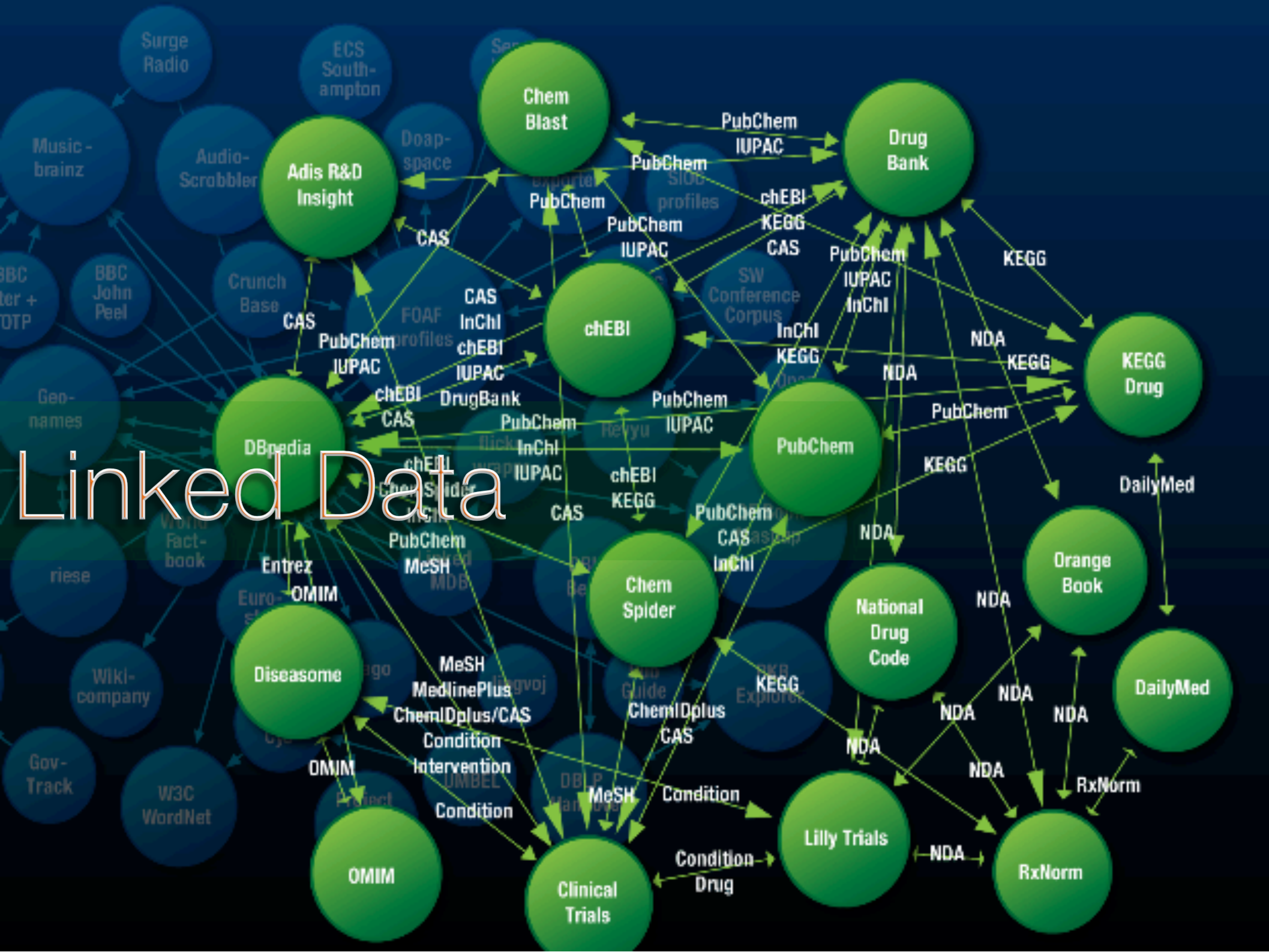
```
<Book>
  rr:tableName "Book_Table"
  rr:subjectMap [
    rr:template "http://...isbn/{ISBN}";
  ];
  …
```



Further R2RML features: logical table

- An alternative could have been to use SQL
  - generate a “logical table”
  - all other definitions are on that logical table
- Would be an overkill for our example, but can be very powerful for complicated cases!




Linked Data
Chem Blast
Drug Bank
Adis R&D Insight
chEBI
KEGG Drug
DBpedia
PubChem
Chem Spider
Diseasome
National Drug Code
Orange Book
DailyMed
OMIM
Clinical Trials
Lilly Trials
RxNorm
PubChem
IUPAC
CAS
InChI
KEGG
NDA
Entrez
MeSH
MedlinePlus
ChemIDplus/CAS
Condition
Intervention
Drug
Surge Radio
ECS South-ampton
Music-brainz
Audio-Scrobbler
Doap-space
BBC John Peel
Crunch Base
FOAF profiles
SW Conference Corpus
Geo-names
Fact-book
Wiki-company
Gov-Track
W3C WordNet
UMBEL

Linked Data “Project”

- Goal: “expose” datasets on the Web
  - remember the importance of data!
- Set links among the data items from different datasets
  - we want to avoid the silo effects

Is your data 5 Star?

★ Available on the web (whatever format), but with an open license

★★ Available as machine-readable structured data (e.g., excel instead of an image scan)

★★★ As before, but using a non-proprietary format (e.g., CSV instead of excel)

★★★★ All the above, plus use open standards (RDF & Co.) to identify things, so that people could point at your stuff

★★★★★ All the above, plus link your data to other people's data to provide context

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 structured data from Wikipedia

```
@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:website <http://www.amsterdam.nl> ;
   dbterm:populationUrban "1364422" ;
   dbterm:areaTotalKm "219" ;
   ...
dbpedia:ABN_AMRO
   dbterm:location dbpedia:Amsterdam ;
   ...
```

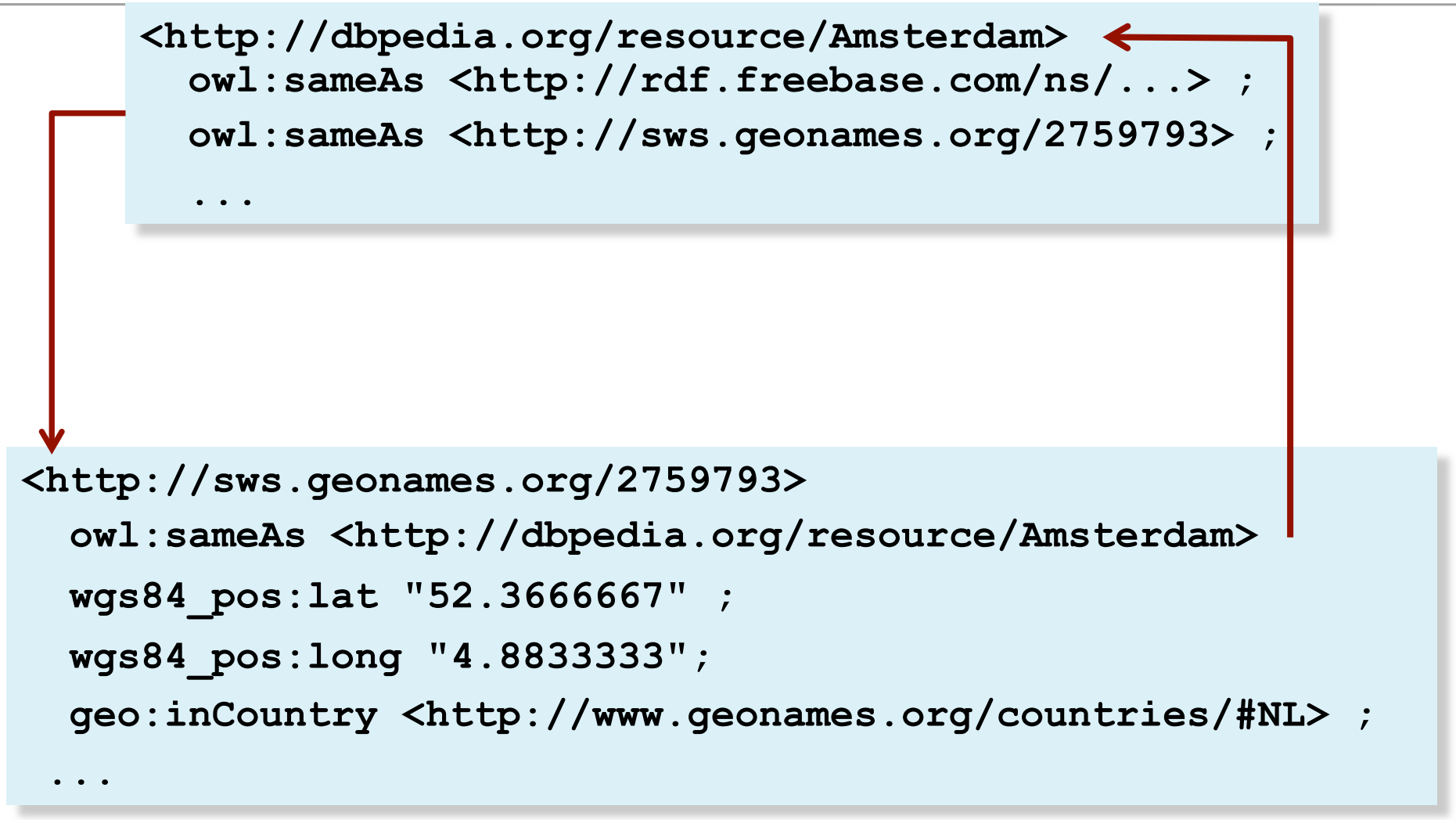
Amsterdam

— Municipality / City —

Coordinates: 52°22′23″N 4°53′32″E

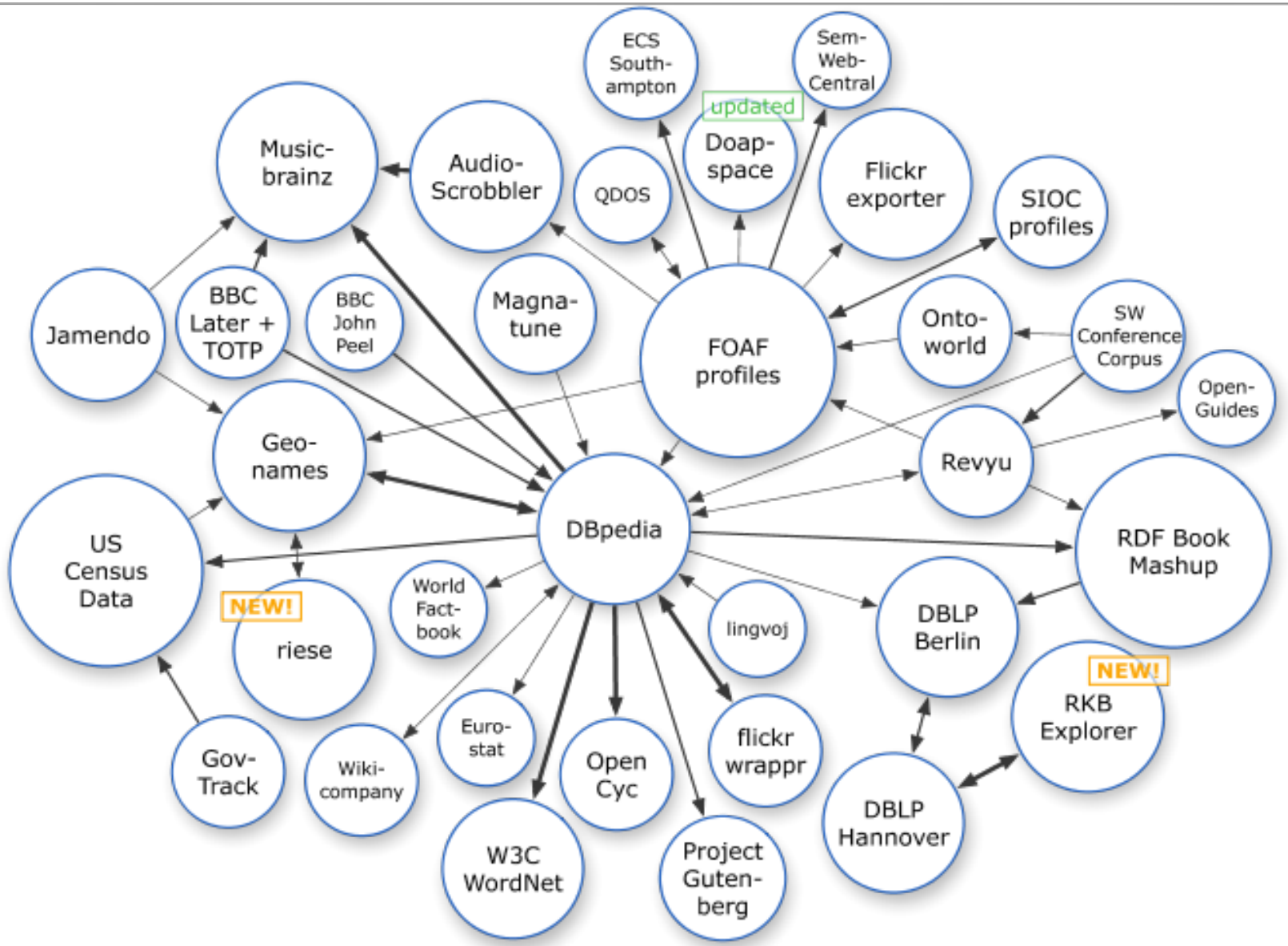
Country	Netherlands
Province	North Holland
COROP	Amsterdam
Boroughs	Boroughs
Government	
- Mayor	Eberhard van der Laan (PvdA)
- Aldermen	Carolien Gjlkiels Hans Gerson Maarten van Poelgeest Freek Ossel Marijke Vos
- Secretary	Henk de Jong
Area[1][2]	
- Municipality / City	219 km² (84.6 sq mi)
- Land	166 km² (64.1 sq mi)
- Water	53 km² (20.5 sq mi)
- Urban	1,003 km² (387.3 sq mi)
- Metro	1,815 km² (700.8 sq mi)
Elevation[3]	2 m (7 ft)
Population (June 2009)[4][5]	
- Municipality / City	762,057
- Density	4,459/km² (11,548.8/sq mi)
- Urban	1,364,422
- Metro	2,158,372
- Demonym	Amsterdammer
Time zone	CET (UTC+01)
- Summer (DST)	CEST (UTC+02) (UTC)
Postal codes	1011–1109
Area code(s)	020
Website	www.amsterdam.nl

Automatic links among open datasets



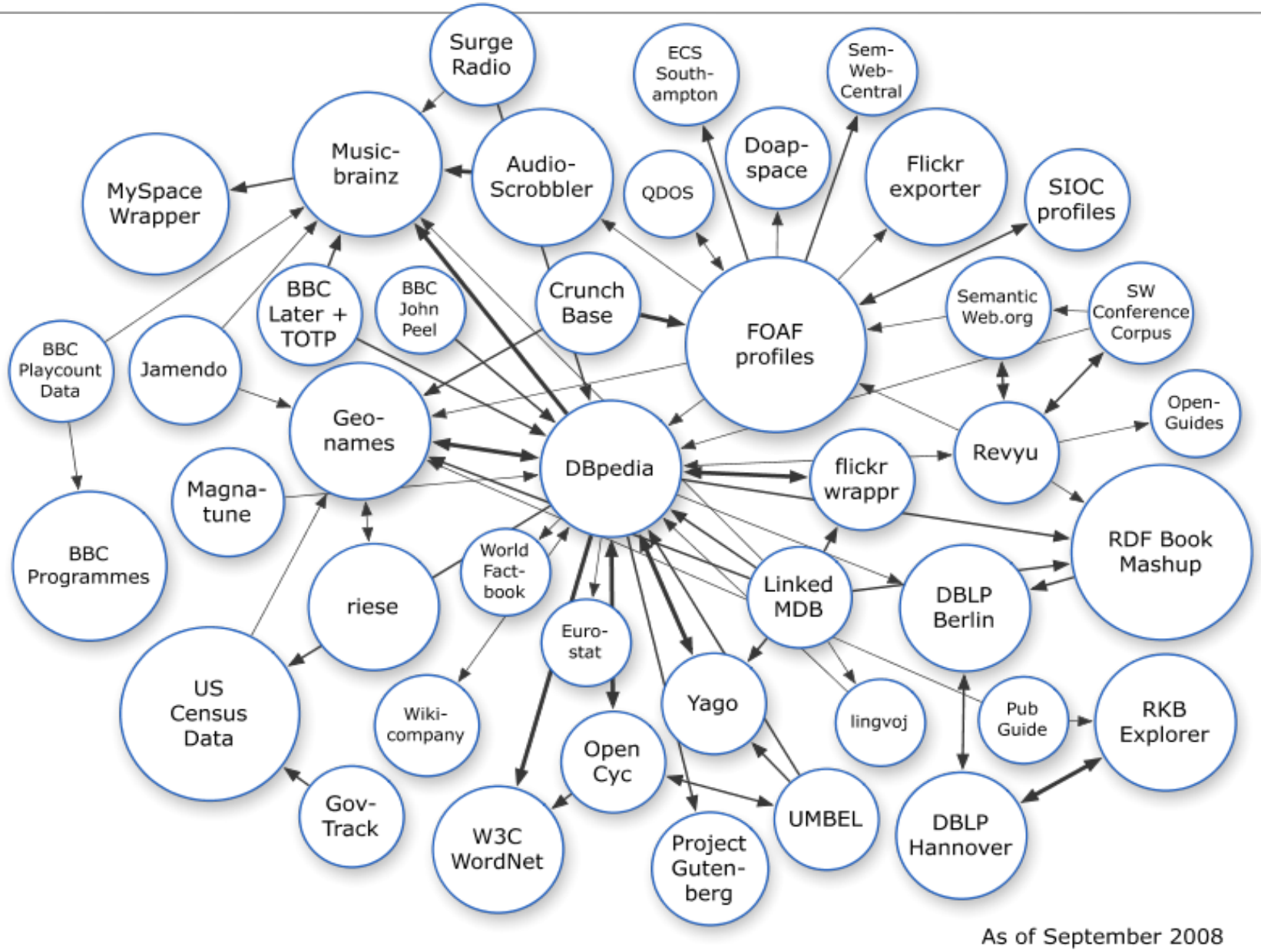
Processors can switch automatically from one to the other…

The LOD “cloud”, March 2008



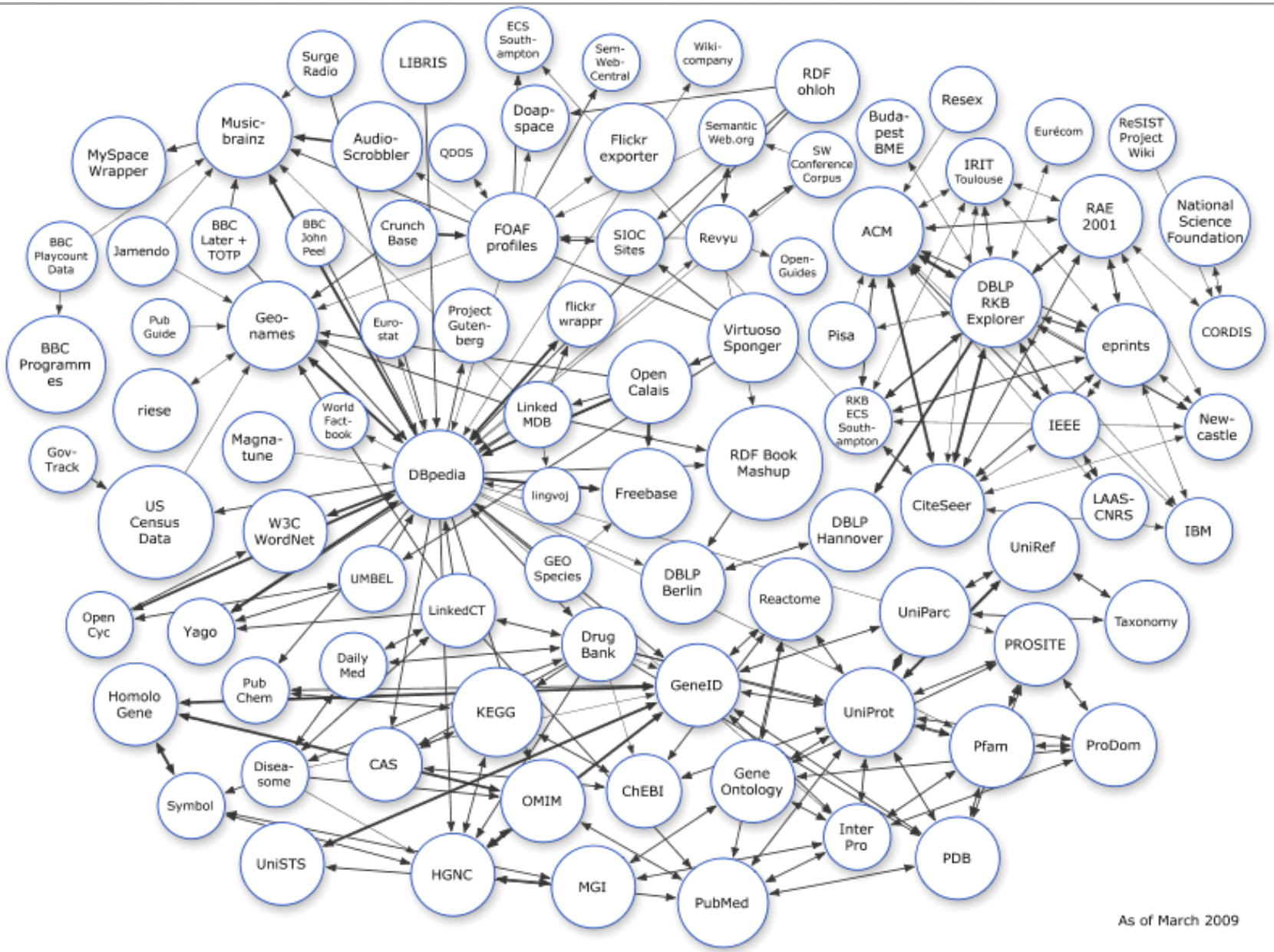
Courtesy of Richard Cyganiak and Anja Jentzsch

The LOD “cloud”, September 2008



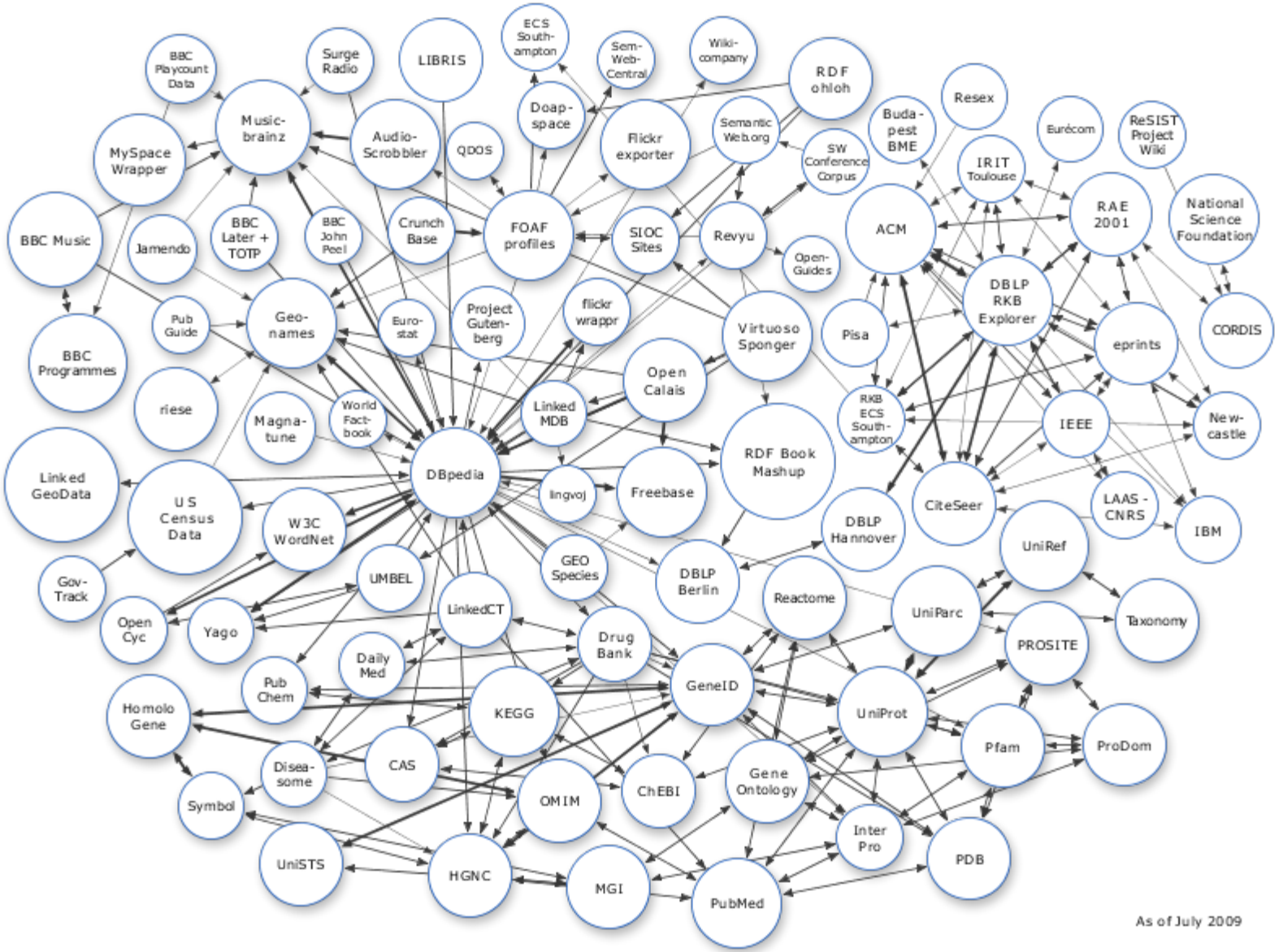
Courtesy of Richard Cyganiak and Anja Jentzsch

The LOD “cloud”, March 2009



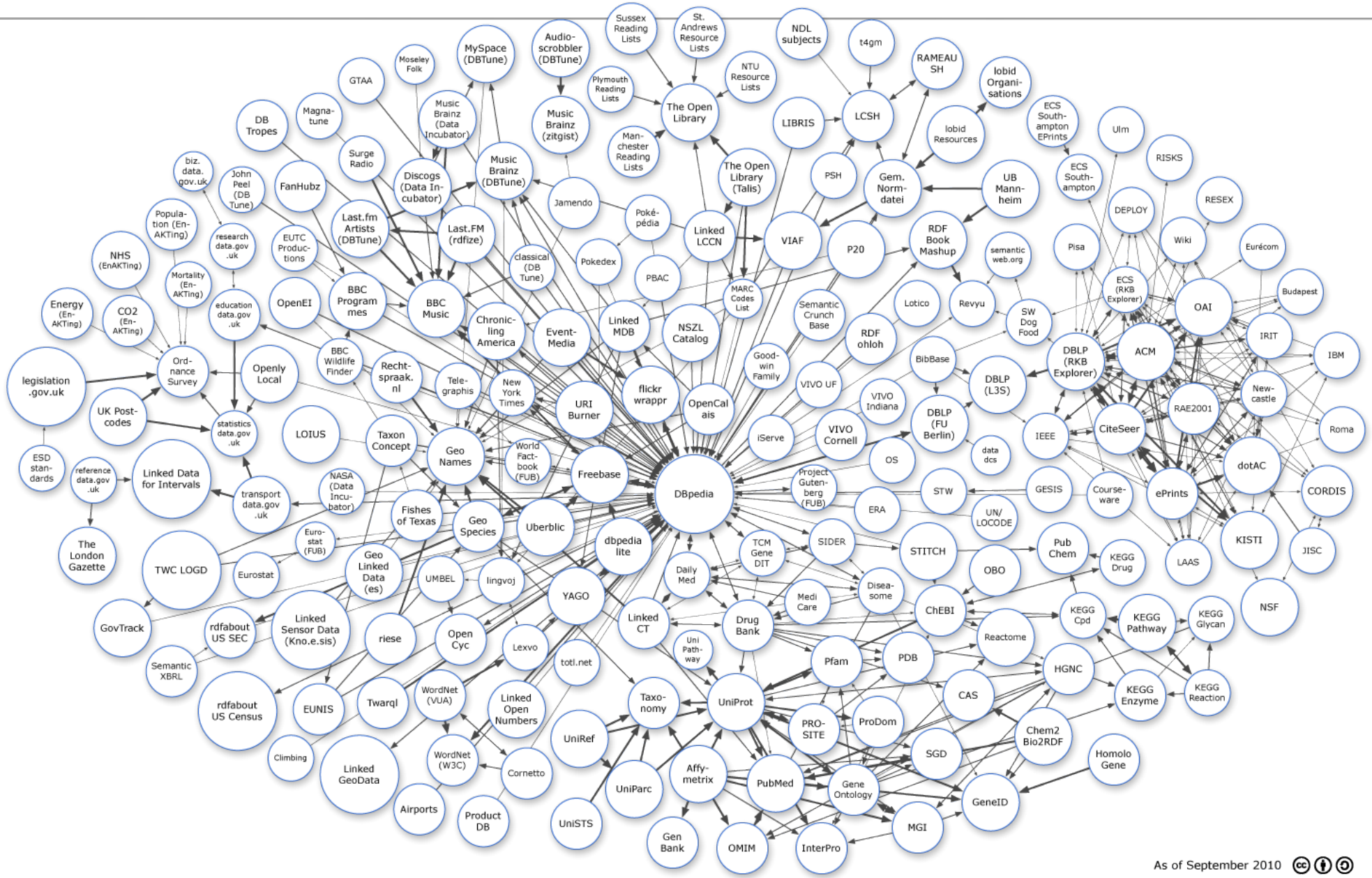
Courtesy of Richard Cyganiak and Anja Jentzsch

The LOD “cloud”, June 2009



Courtesy of Richard Cyganiak and Anja Jentzsch

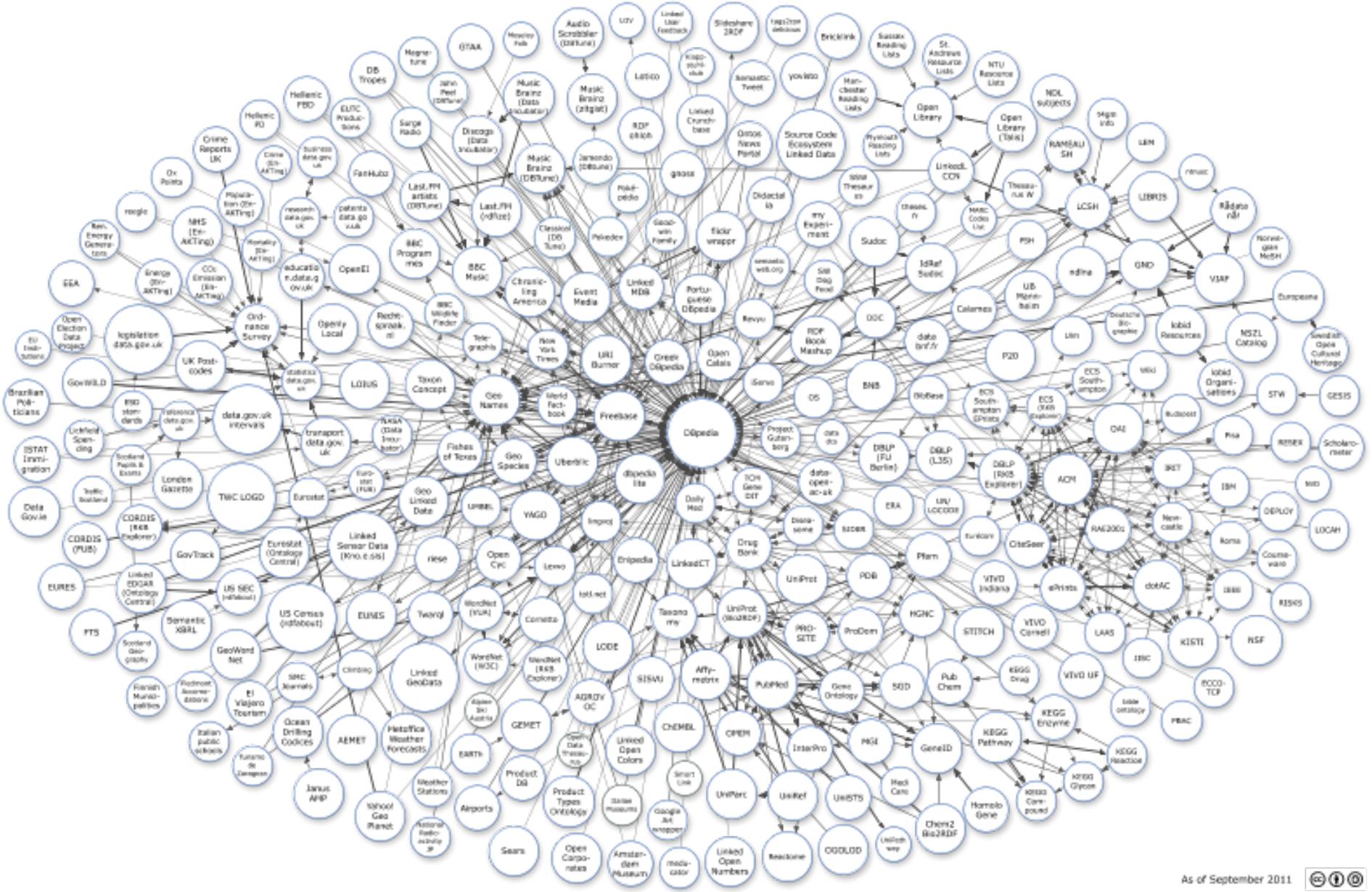
The LOD “cloud”, September 2010



As of September 2010

Courtesy of Richard Cyganiak and Anja Jentzsch

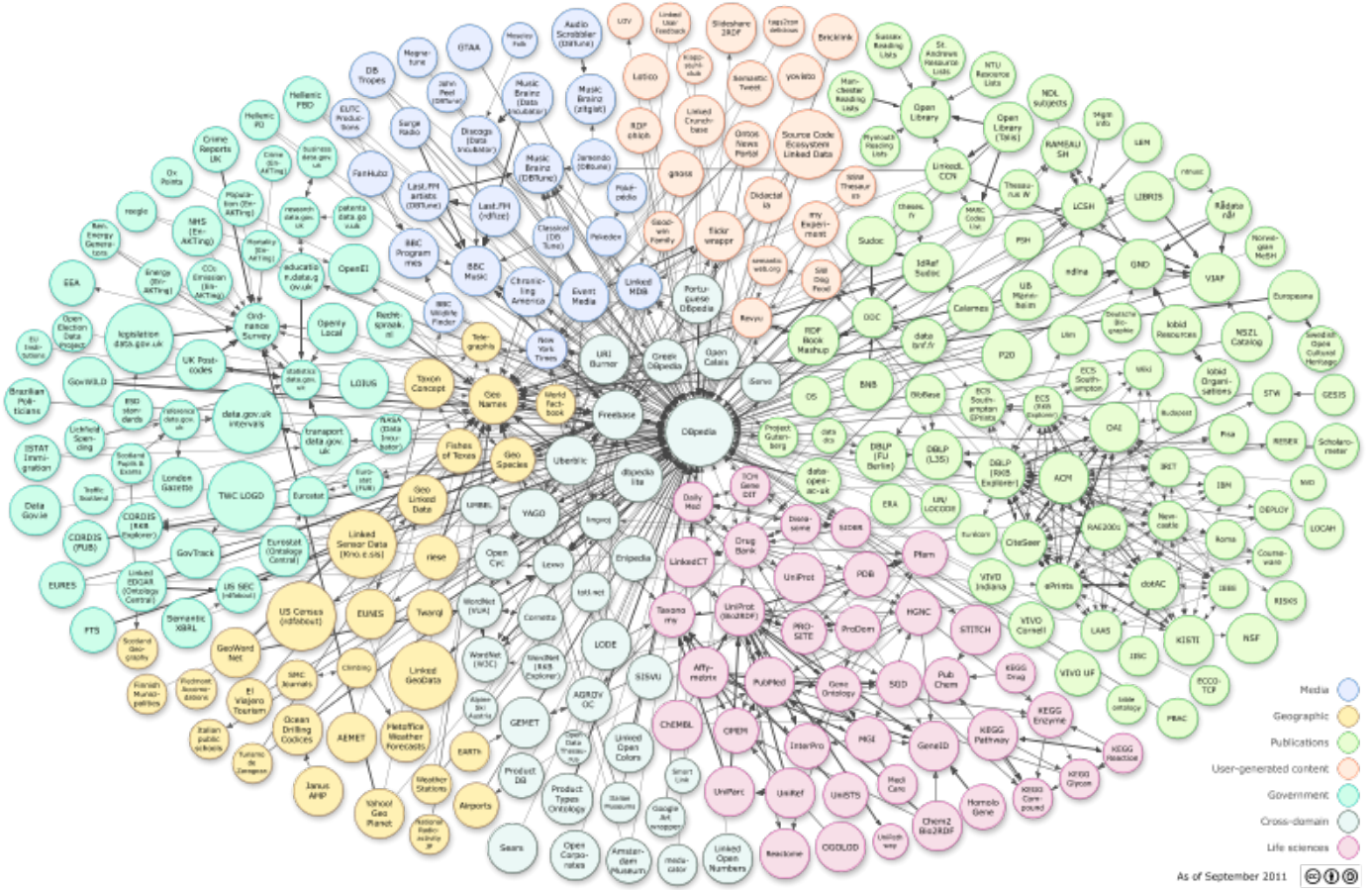
The LOD “cloud”, September 2011



Courtesy of Richard Cyganiak and Anja Jentzsch

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



Courtesy of Richard Cyganiak and Anja Jentzsch

The importance of Linked Data

- It provides a core set of data that Semantic Web applications can build on
  - stable references for “things”,
    - e.g., http://dbpedia.org/resource/Amsterdam
  - many many relationships that applications may reuse
    - e.g., the BBC application!
  - a “nucleus” for a larger, semantically enabled Web!
- For many, publishing data may be the first step into the world of Semantic Web

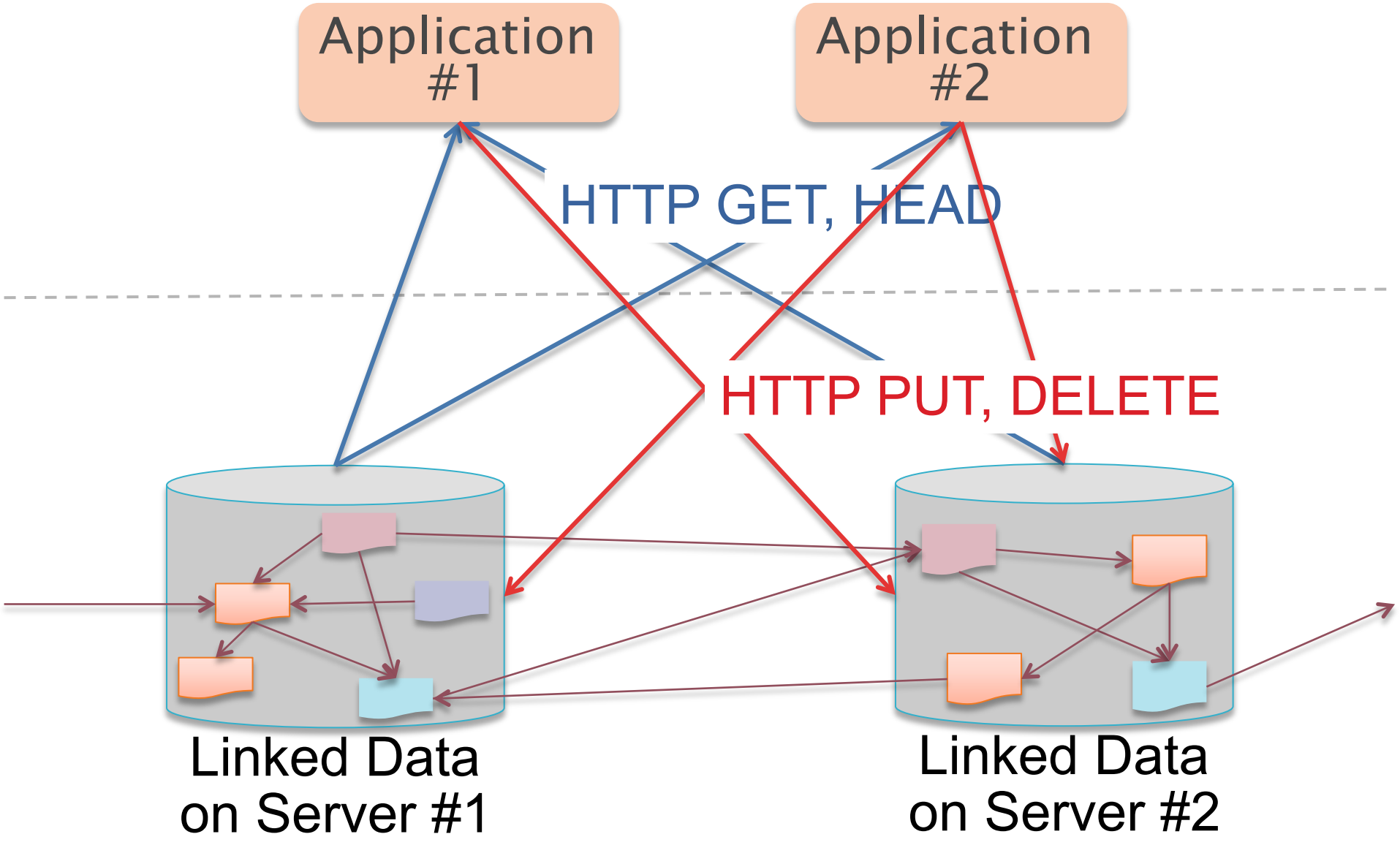
Some things to remember if you publish data

- Publish your data first, care about sexy user interfaces later!
  - the “raw data” can become useful on its own right and others may use it
  - you can add your added value later by providing nice user access
- If possible, publish your data in RDF but if you cannot, others may help you in conversions
  - trust the community…
- *Add links to other data*. “Just” publishing isn’t enough…

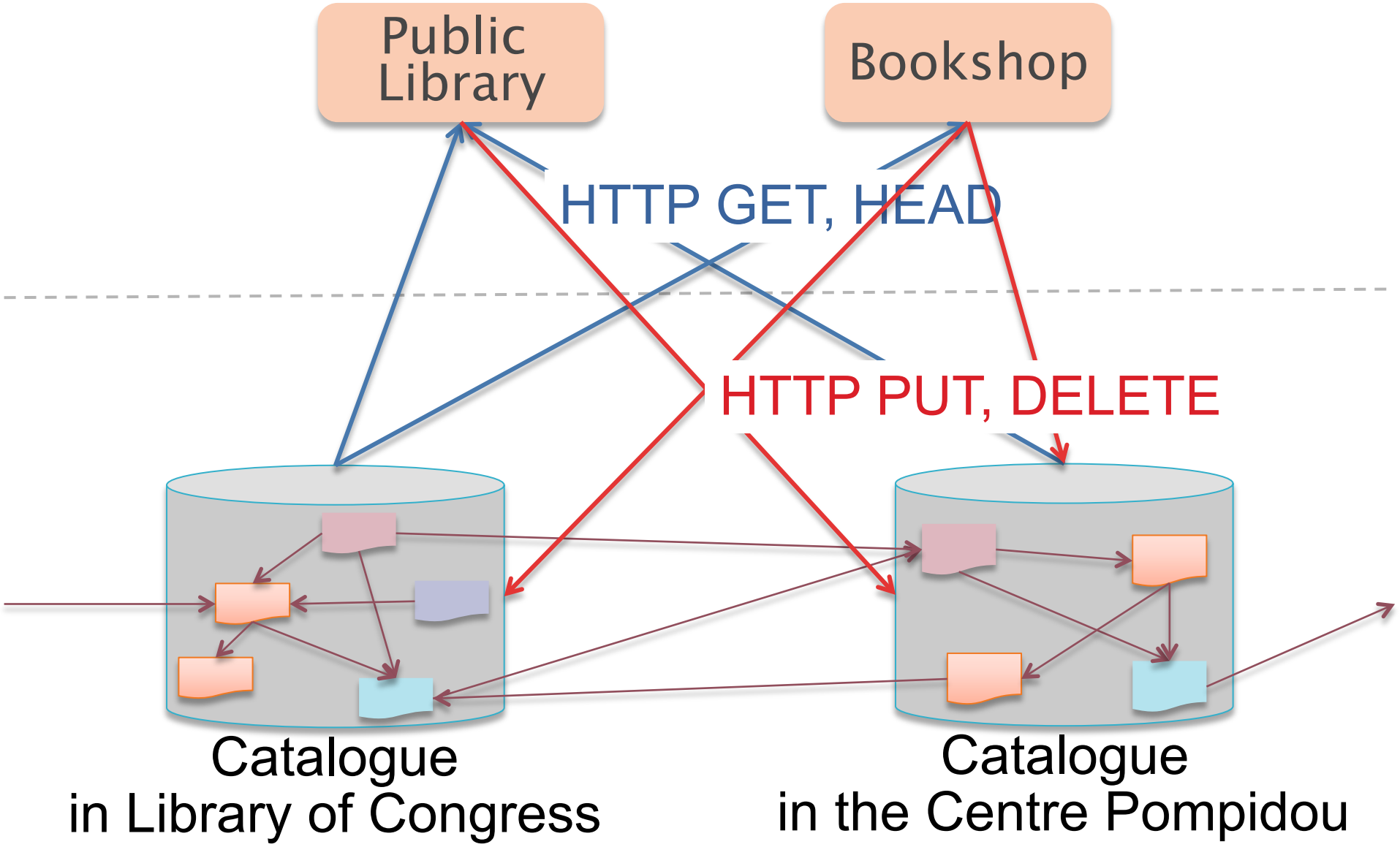
Read only vs. Read/Write

- Currently, Linked Data is dominated by *publishing* data for read-only usage
  - creating/updating the data is done “out of band”
- The future to read *and* write Linked Data

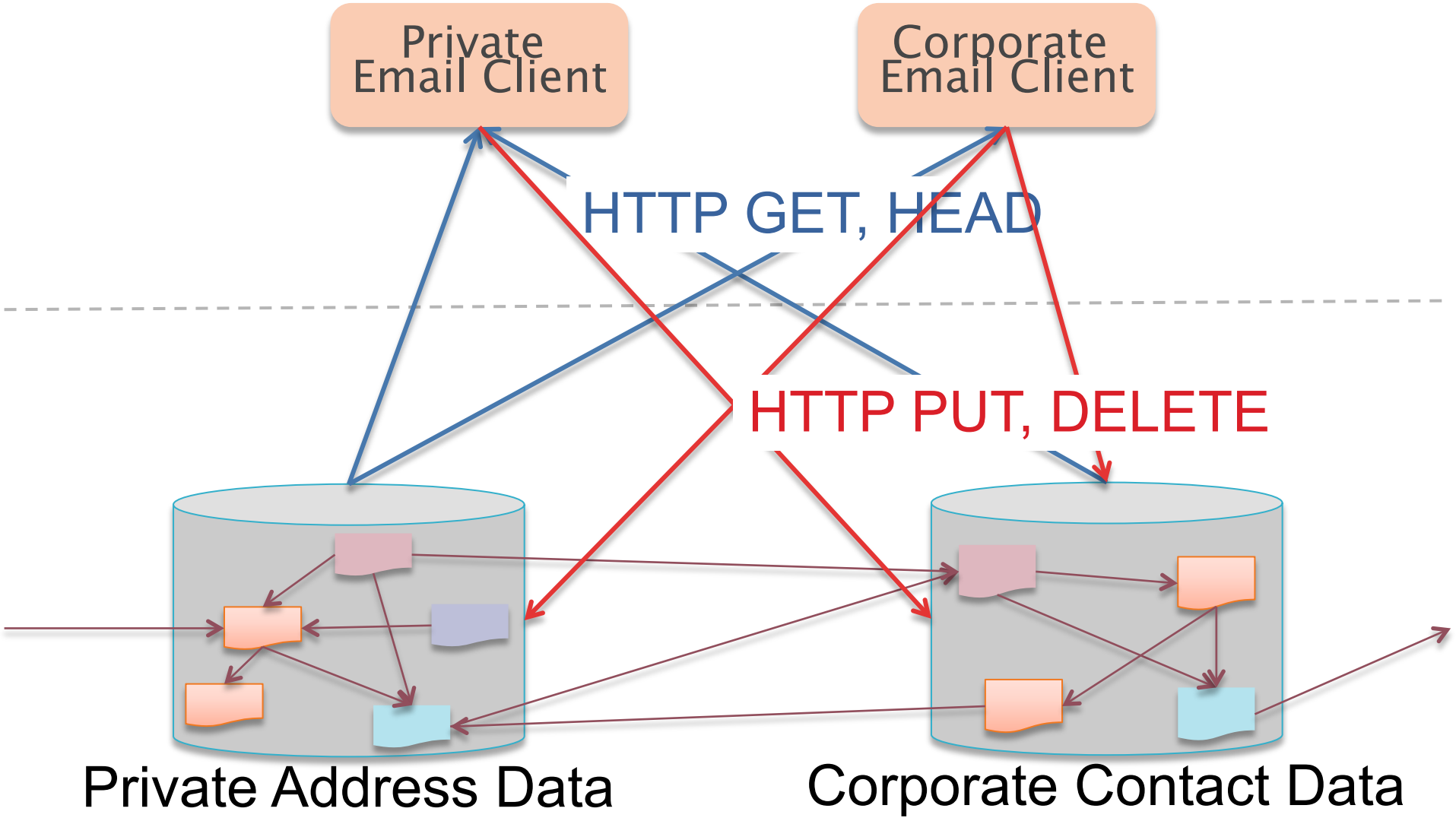
Example: Simple Application Integration via Linked Data



Example: Simple Application Integration via Linked Data

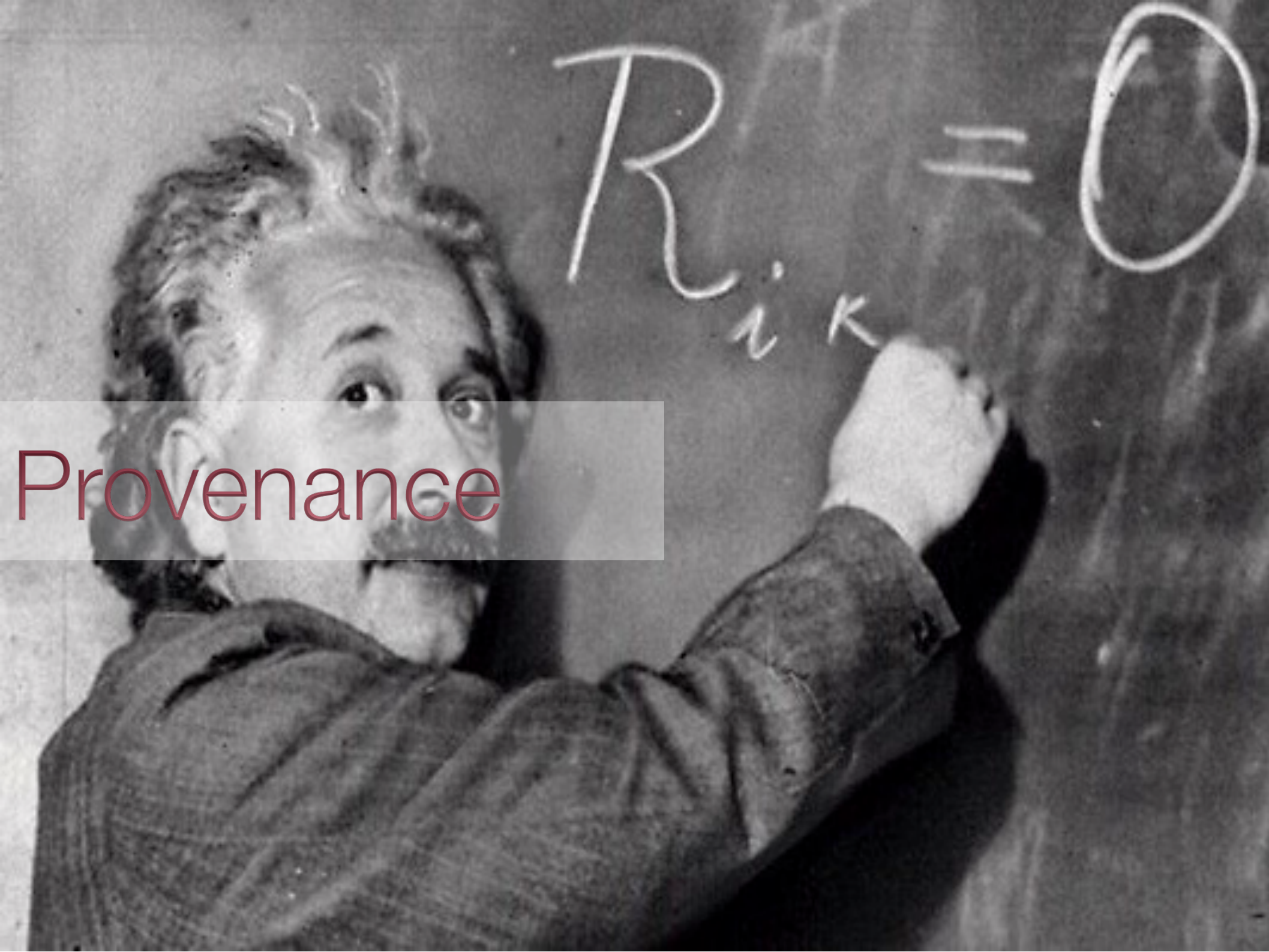


Example: Simple Application Integration via Linked Data



Linked Data Platform

- Define an “entry” level, HTTP/RESTful based infrastructure to publish, read, write, or modify linked data
  - typical usage: data intensive application in a browser, application integration using shared data…
- The infrastructure should be easy to implement and install
  - provides an “entry point” for Linked Data applications!
- The work has just started to flesh out the details…



Provenance

The goal is simple…

- We should be able to express all sorts of “meta” information on the data
  - who played what role in creating the data (author, reviewer, etc.)
  - view of the full revision chain of the data
  - in case of data integration: which part comes from which original data and under what process
  - what vocabularies/ontologies/rules were used to generate some portions of the data
  - etc.

…the solution is more complicated

- Requires a complete model describing the various constituents (actors, revisions, etc.)
- The model should be usable with RDF
- Has to find a balance between
  - simple (“scruffy”) provenance: easily usable and editable
  - complex (“complete”) provenance: allows for a detailed reporting of origins, versions, etc.
- That is the role of the Provenance Working Group (started in 2011)

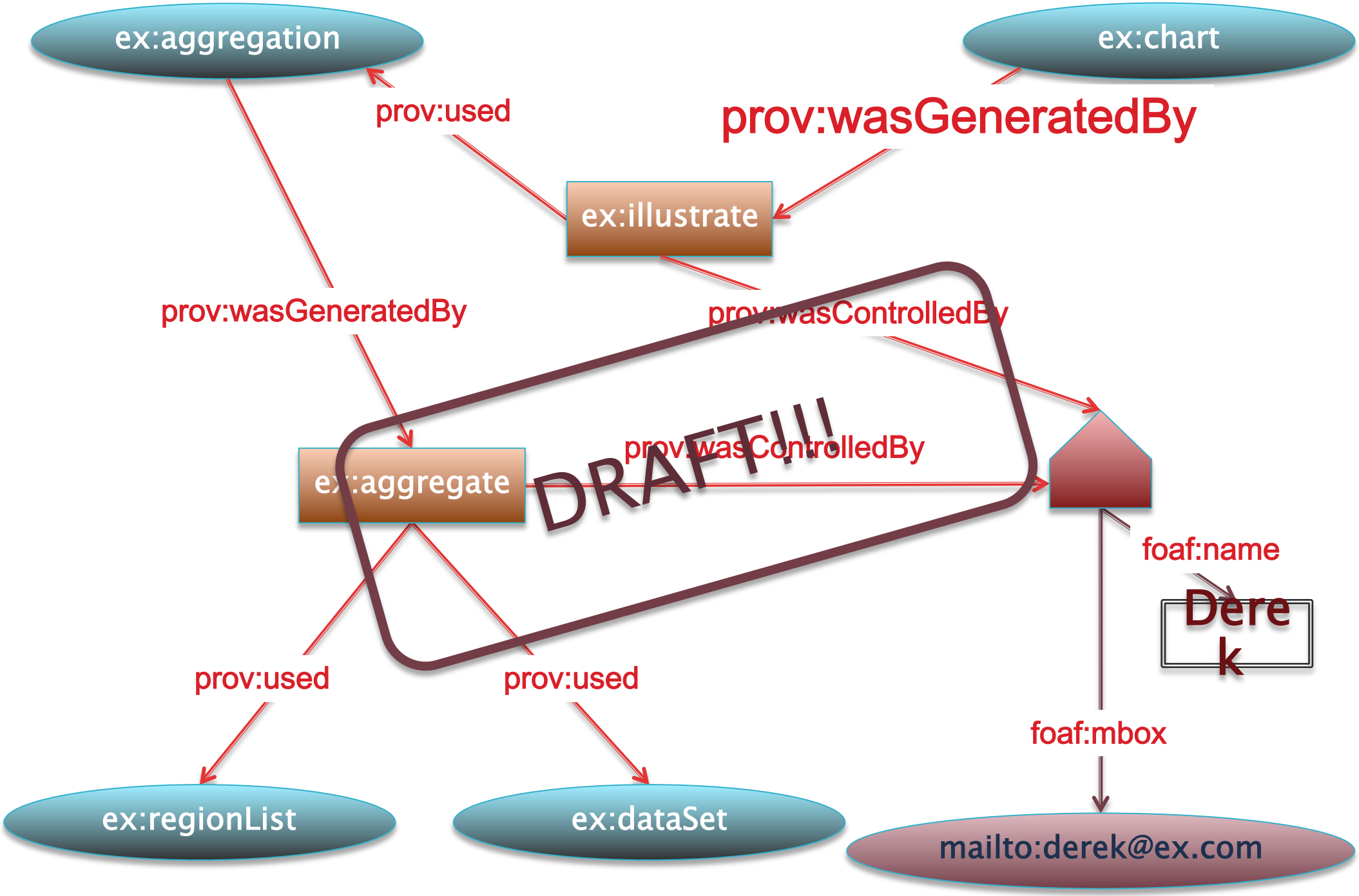

ex:aggregation
ex:chart
prov:used
prov:wasGeneratedBy
ex:illustrate
prov:wasGeneratedBy
prov:wasControlledBy
DRAFT!!!
ex:aggregate
prov:wasControlledBy
foaf:name
Dere
k
prov:used
prov:used
foaf:mbox
ex:regionList
ex:dataSet
mailto:derek@ex.com

Photo credit "reedster", Flickr

RDF data access

- The HTTP/RESTful approach is great to build up simple applications
- But as data grows, complexity of relationships grows
  - more sophisticated query possibilities are necessary!
- How do I query the RDF data?
  - e.g., how do I get to the DBpedia data?

Querying RDF graphs

- Remember the Python+RDFLib idiom:

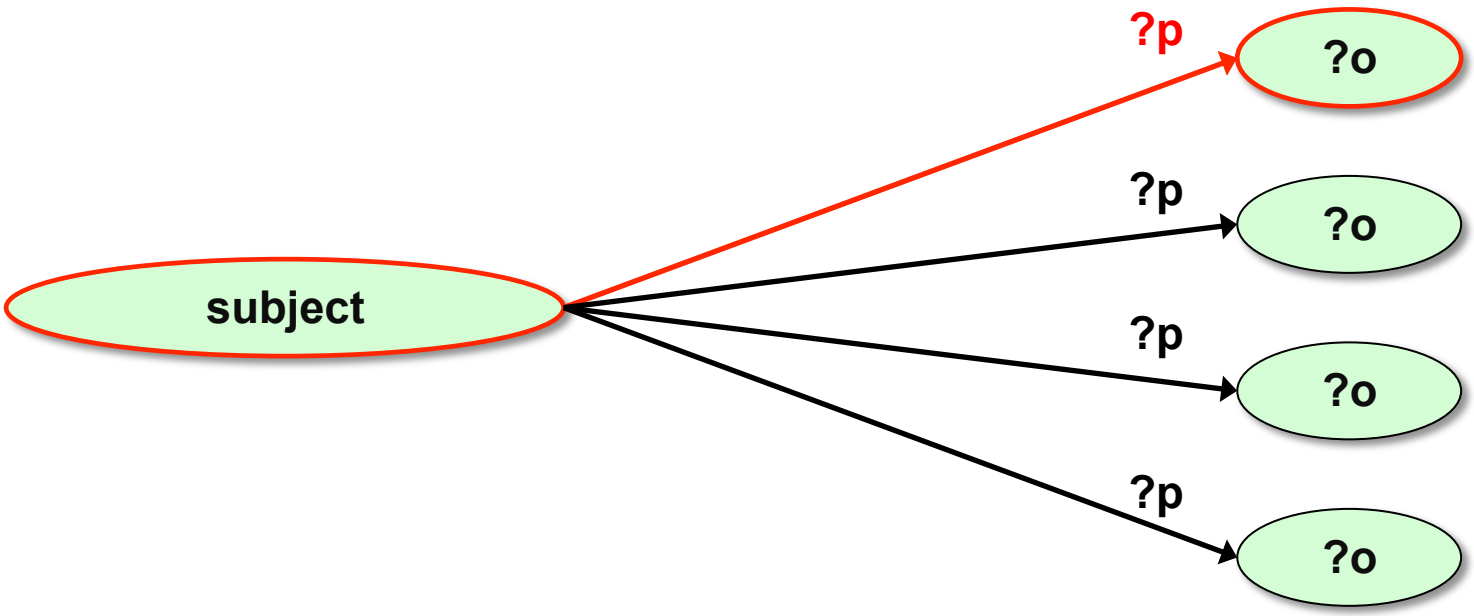
```
for (s,p,o) in graph.triples((subject,None,None)) :
  do_something(p,o);
```

Querying RDF graphs

- 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 Python+RDFLib example

```
for (s,p,o) in graph.triples((subject,None,None)) :
  do_something(p,o);
```



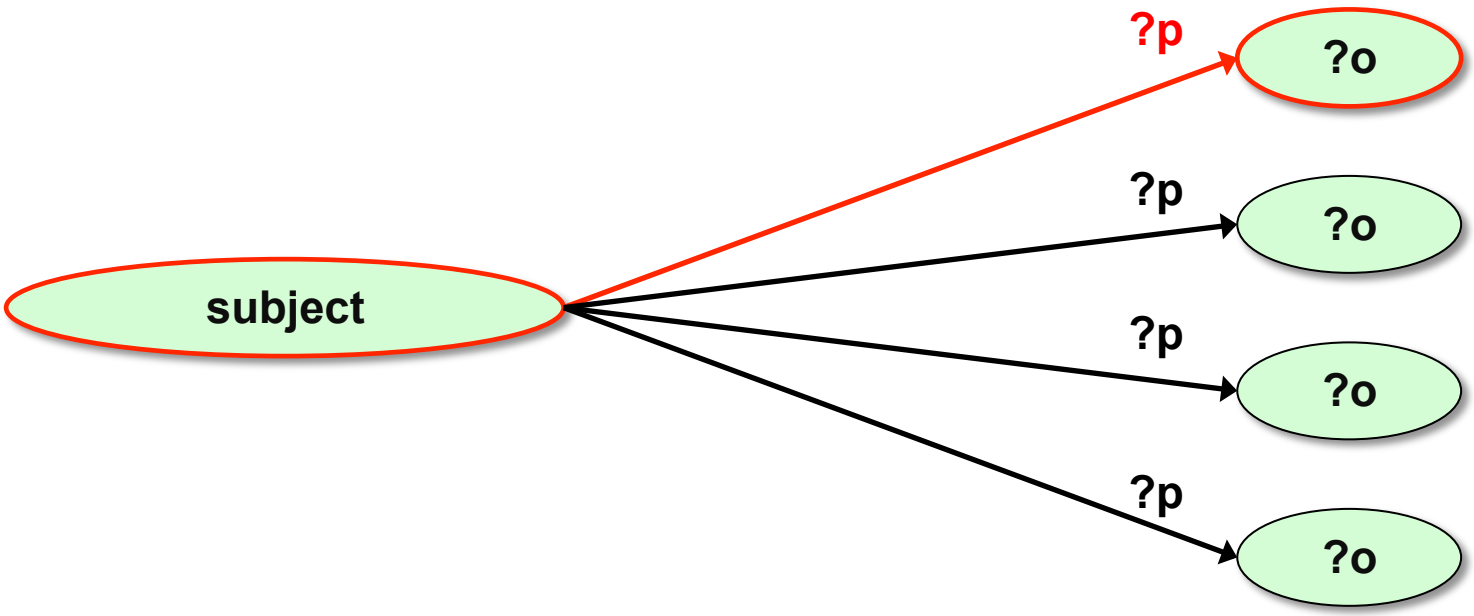
General: graph patterns

- 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 the bound resources

Our Python example in SPARQL

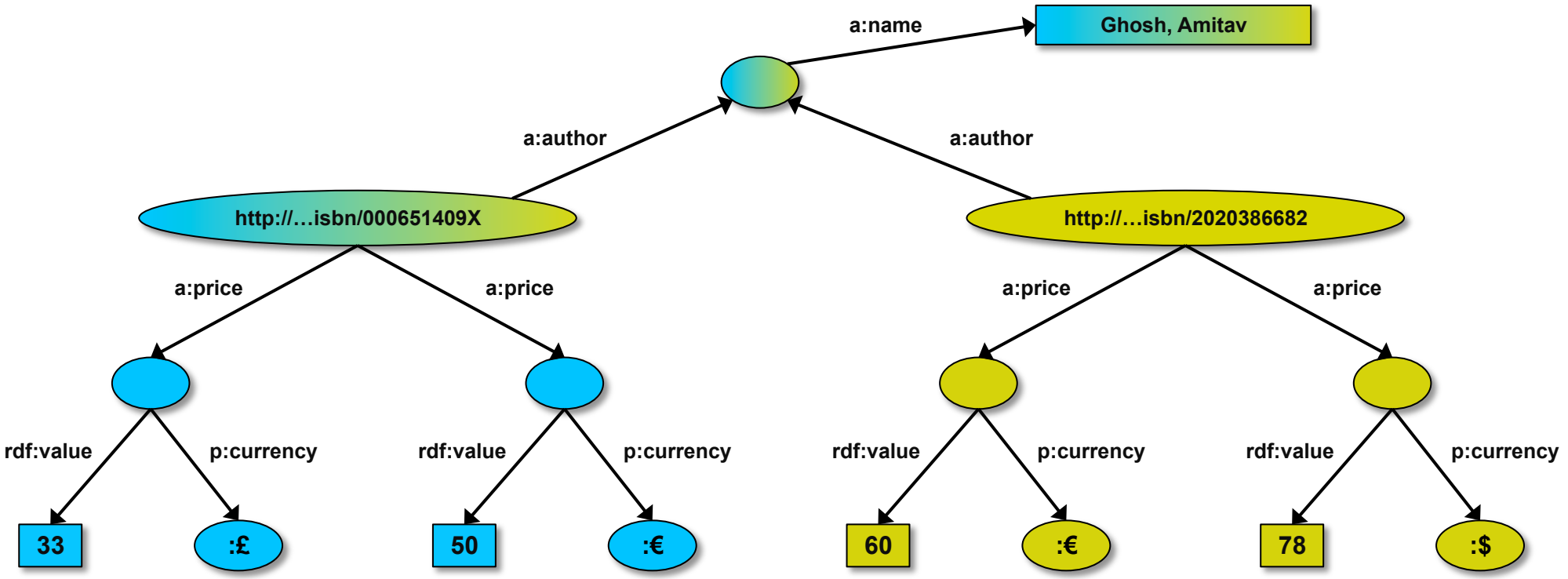
```
SELECT ?p ?o
WHERE {subject ?p ?o}
```

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



Simple SPARQL example

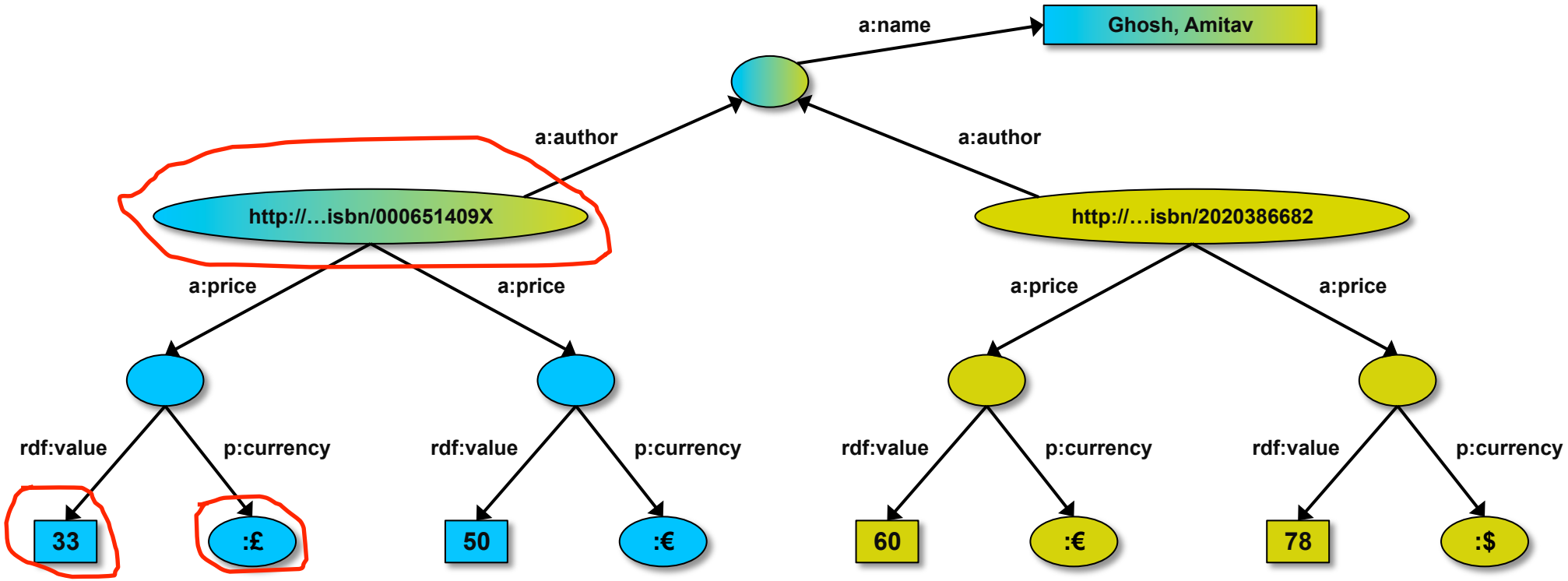
```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



Simple SPARQL example

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

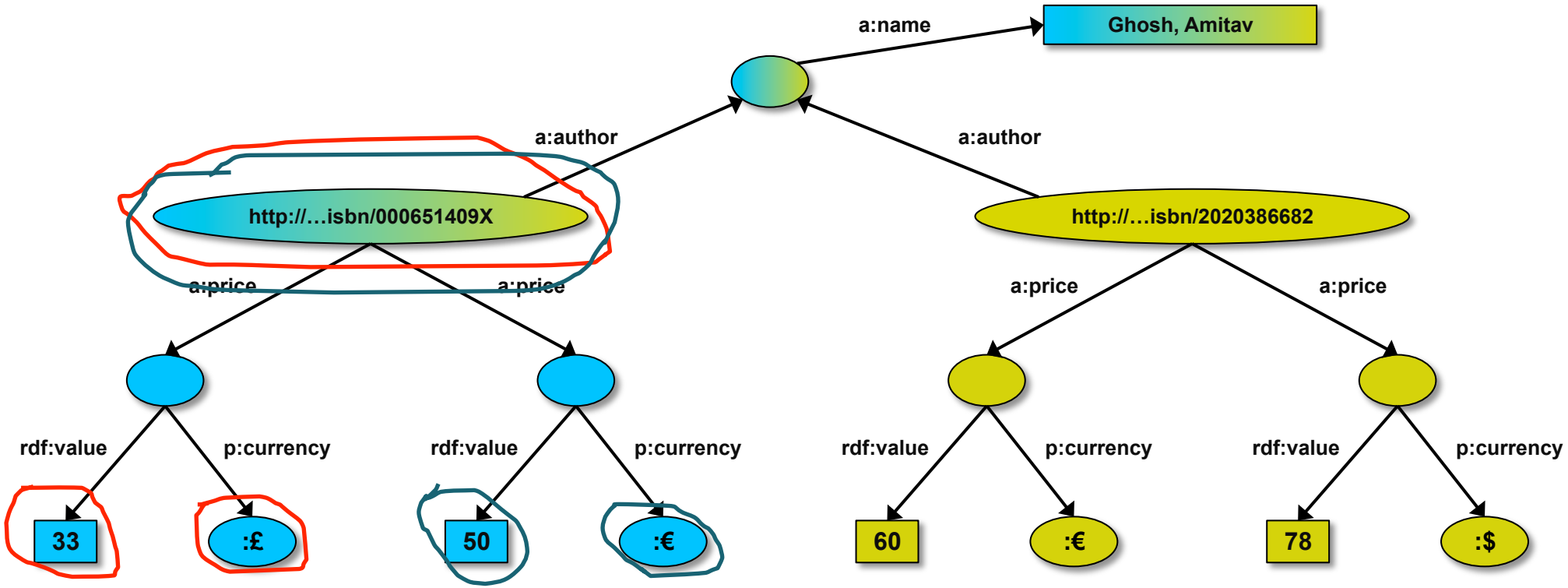
Returns: [<…409X>,33,:£]



Simple SPARQL example

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

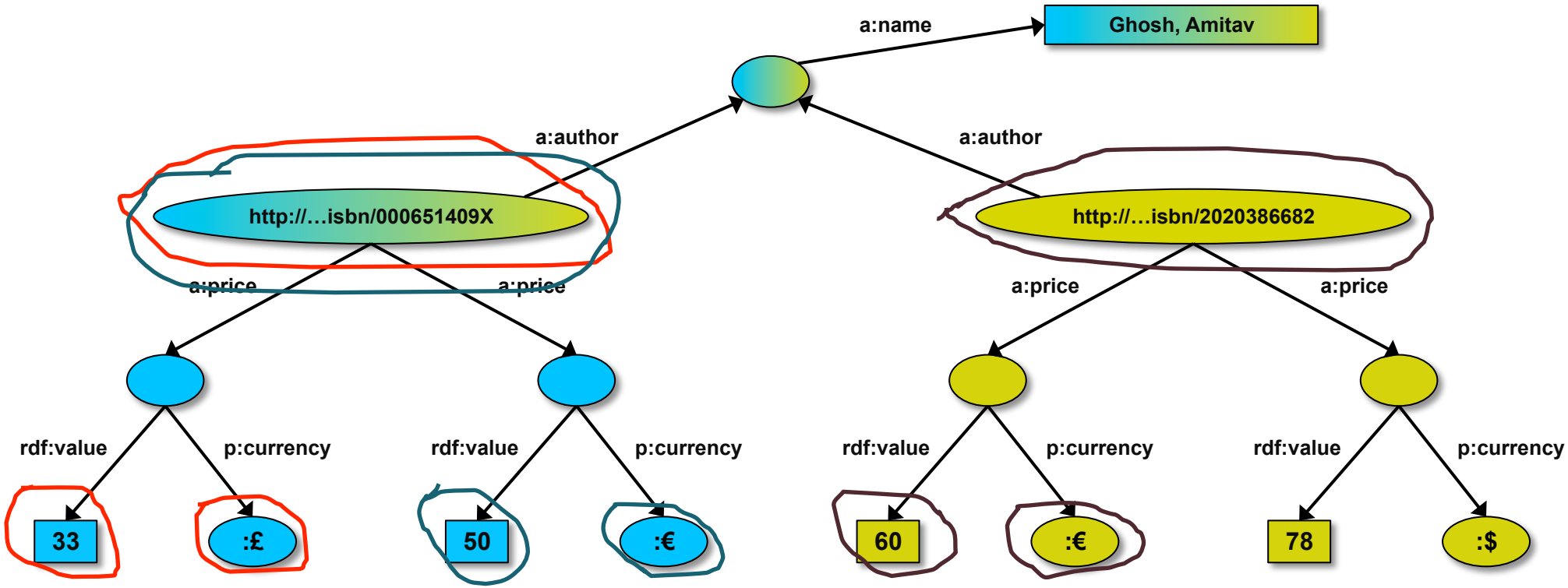
Returns: [<…409X>,33,:£], [<…409X>,50,:€]



Simple SPARQL example

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

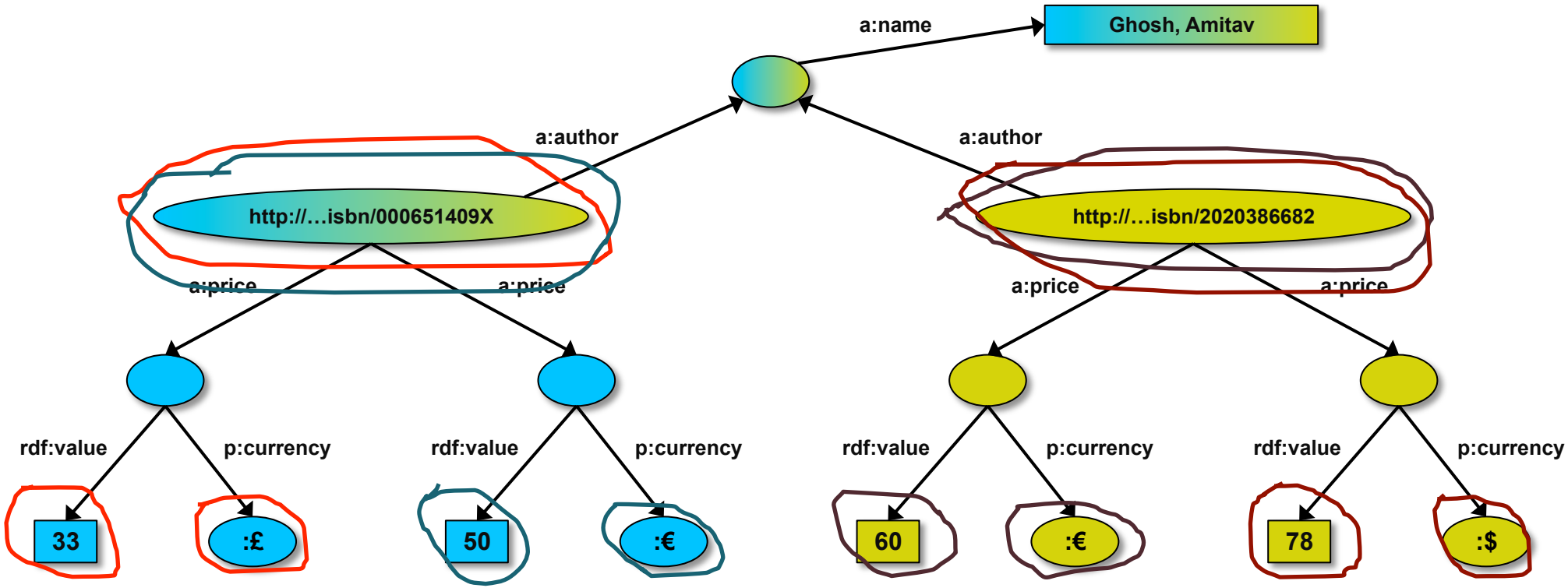
Returns: [<…409X>,33,:£], [<…409X>,50,:€],
[<…6682>,60,:€]



Simple SPARQL example

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

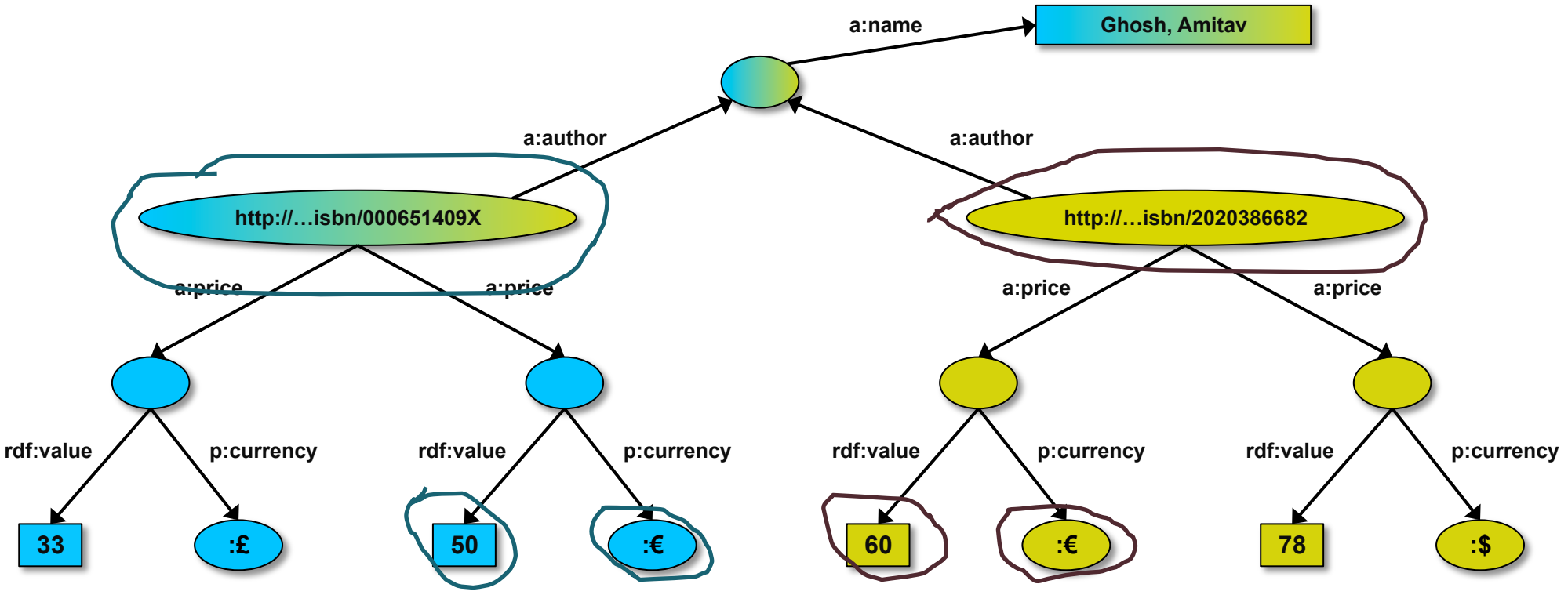
Returns: [<…409X>,33,:£], [<…409X>,50,:€],
[<…6682>,60,:€], [<…6682>,78,:$]



Pattern constraints

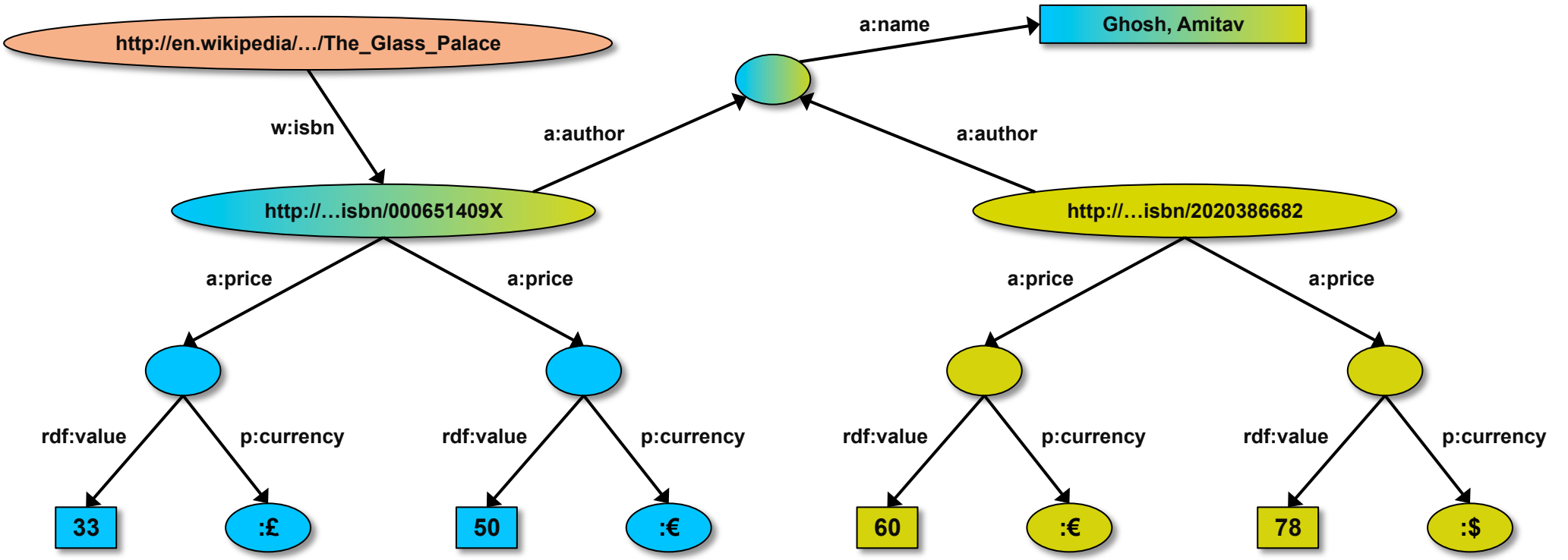
```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        FILTER(?currency == :€) }
```

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



Optional pattern

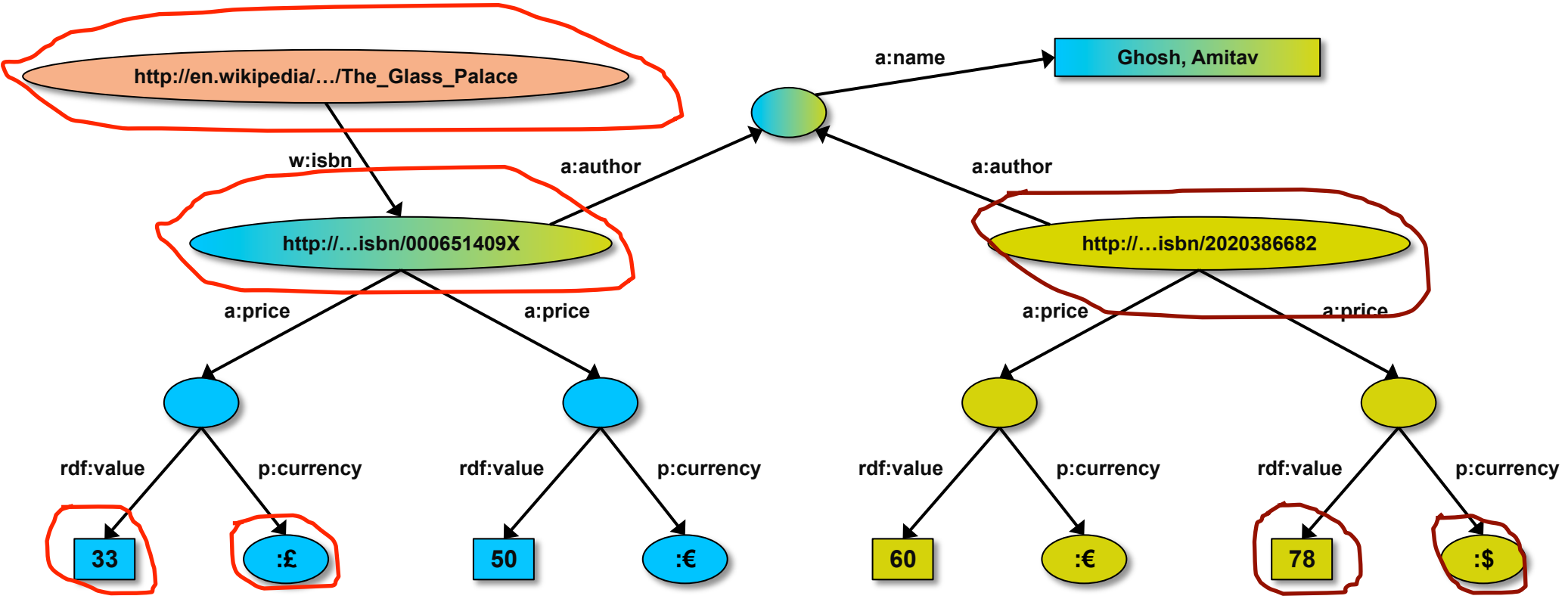
```
SELECT ?isbn ?price ?currency ?wiki
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        OPTIONAL ?wiki w:isbn ?isbn. }
```



Optional pattern

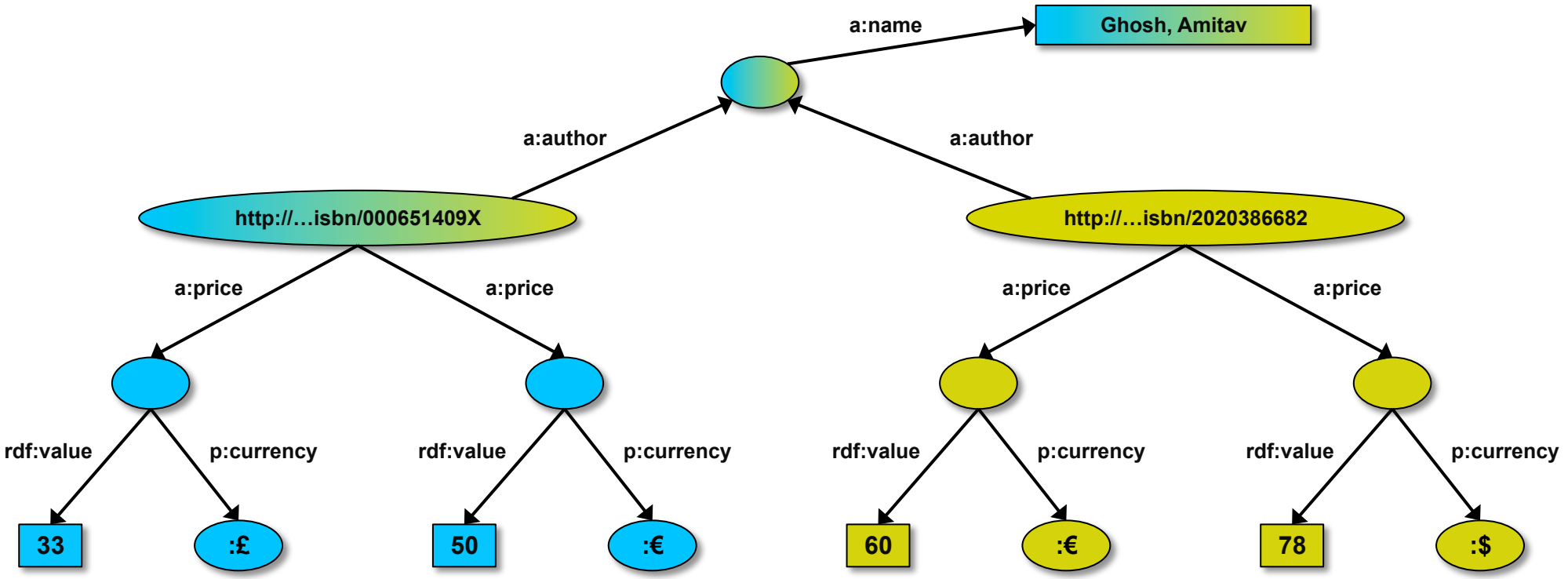
```
SELECT ?isbn ?price ?currency ?wiki
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        OPTIONAL ?wiki w:isbn ?isbn. }
```

Returns: [[<..09X>,33,:£,<…Palace>], … , [<..6682>,78,:$,]]



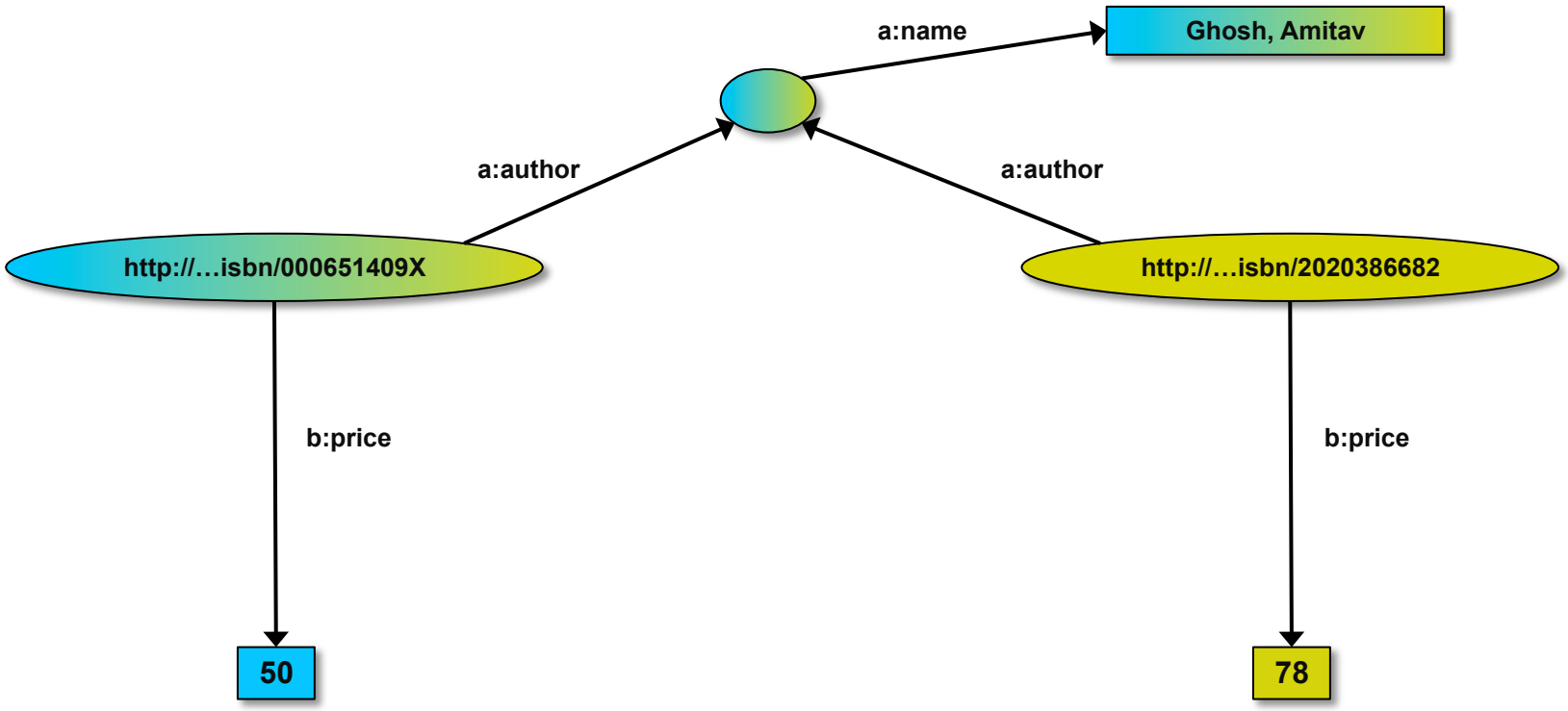
Construct a new graph

```
CONSTRUCT { ?isbn b:price ?price.
            ?isbn a:author ?y. ?y a:name ?name . }
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        ?isbn a:author ?y. ?y a:name ?name .
        FILTER(?currency == :€) }
```



Construct a new graph

```
CONSTRUCT { ?isbn b:price ?price.
            ?isbn a:author ?y. ?y a:name ?name . }
WHERE { ?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.
        ?isbn a:author ?y. ?y a:name ?name .
        FILTER(?currency == :€) }
```



Other SPARQL features

- Limit the number of returned results; remove duplicates, sort them, …
- Specify several data sources (via URI-s) within the query
- Construct a graph combining a separate pattern and the query results
- Use datatypes and/or language tags when matching a pattern
- Aggregation of the results (min, max, average, etc.)
- Path expressions (a bit like regular expressions)

SPARQL usage in practice

- SPARQL is usually used over the network
  - http request is sent to a SPARQL endpoint
  - result is the result of the SELECT, the CONSTRUCT,…
- 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

Remote query/reply example

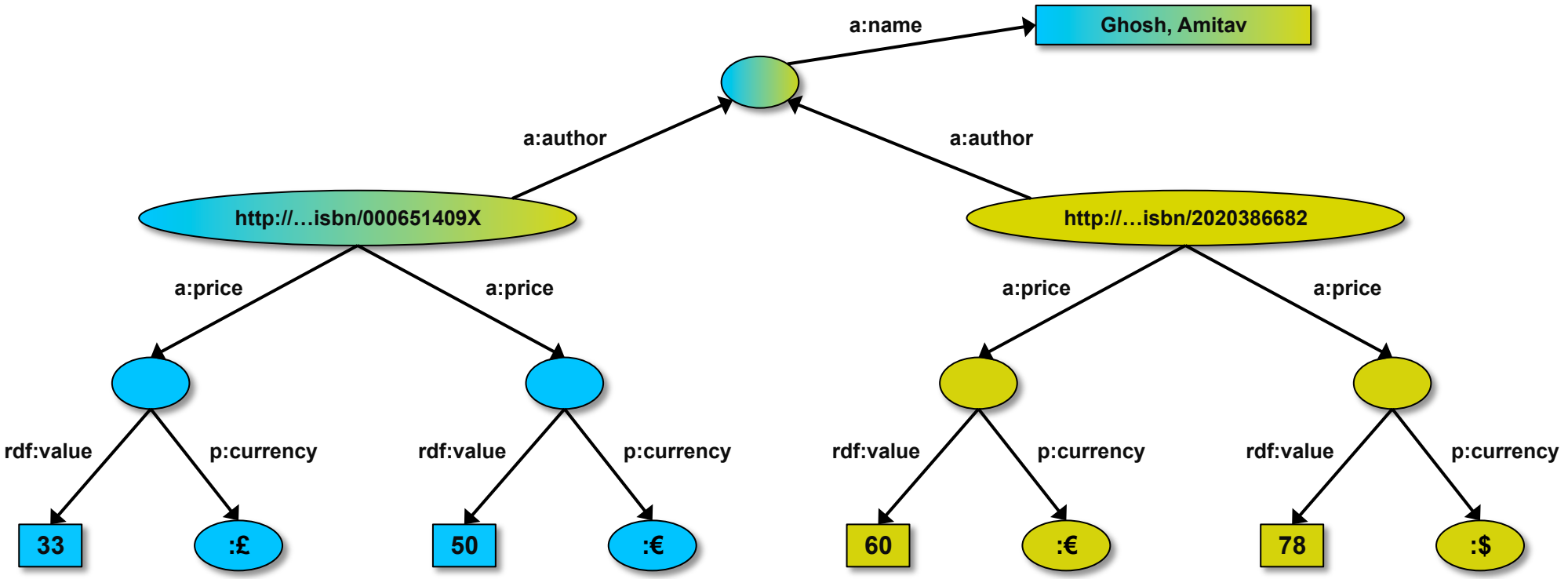
```
GET /qps?&query=SELECT+:…+WHERE:+… HTTP/1.1
User-Agent: my-sparql-client/0.0
Host: my.example

HTTP/1.1 200 OK
Server: my-sparql-server/0.0
Content-Type: application/sparql-results+xml

<?xml version="1.0" encoding="UTF-8"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#>
  <head>
    <variable name="a"/>
    ...
  </head>
  <results>
    <result ordered="false" distinct="false">
      <binding name="a"><uri>http:…</uri></binding>
      ...
    </result>
    <result> ... </result>
  </results>
</sparql>
```

SPARQL 1.1 Update

- SPARQL CONSTRUCT returns a new, modified graph
  - the original data remains unchanged!
- SPARQL 1.1 Update *modifies the original dataset!*

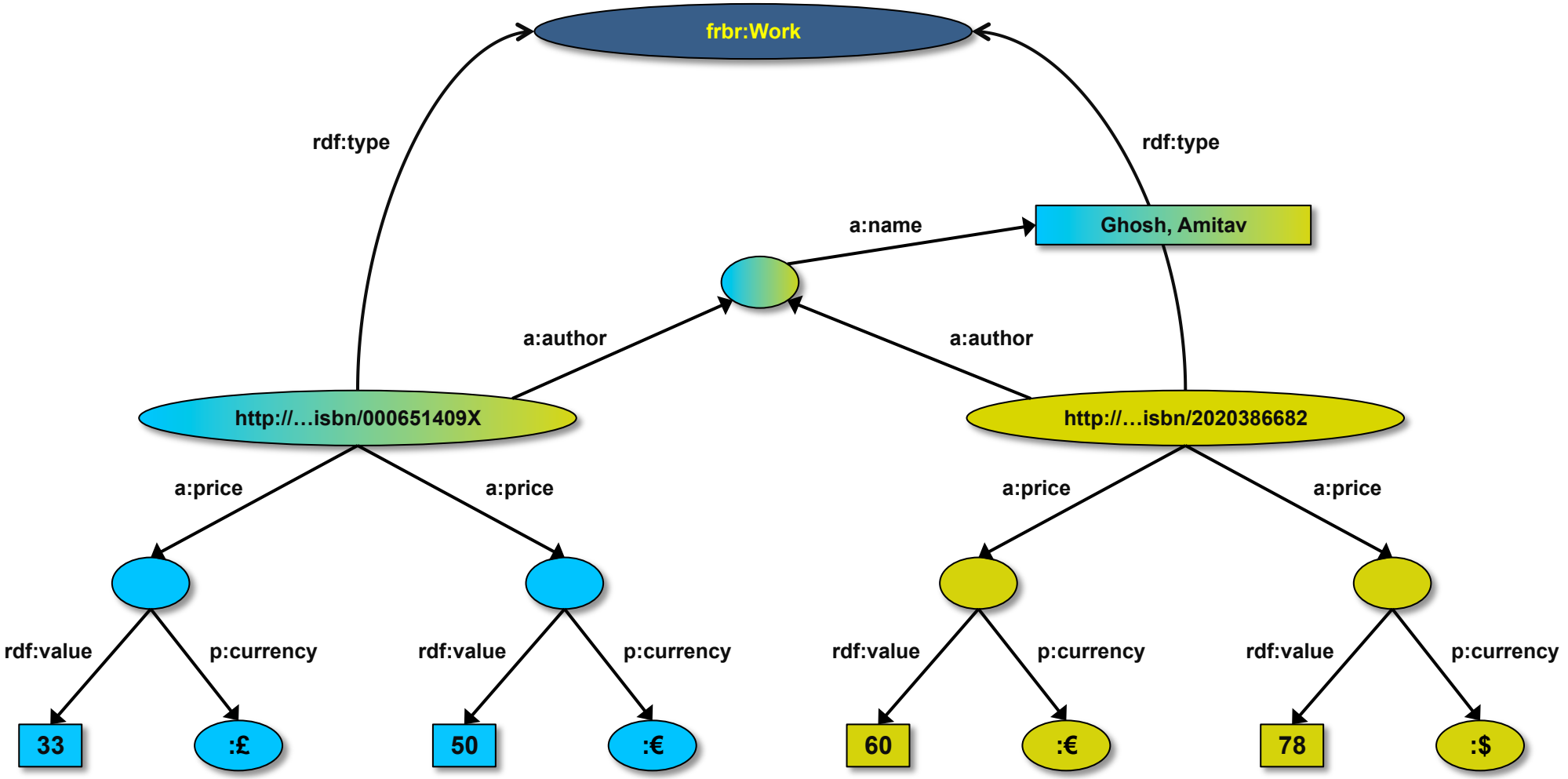
Update: insert

```
INSERT {?isbn rdf:type frbr:Work}
WHERE  {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



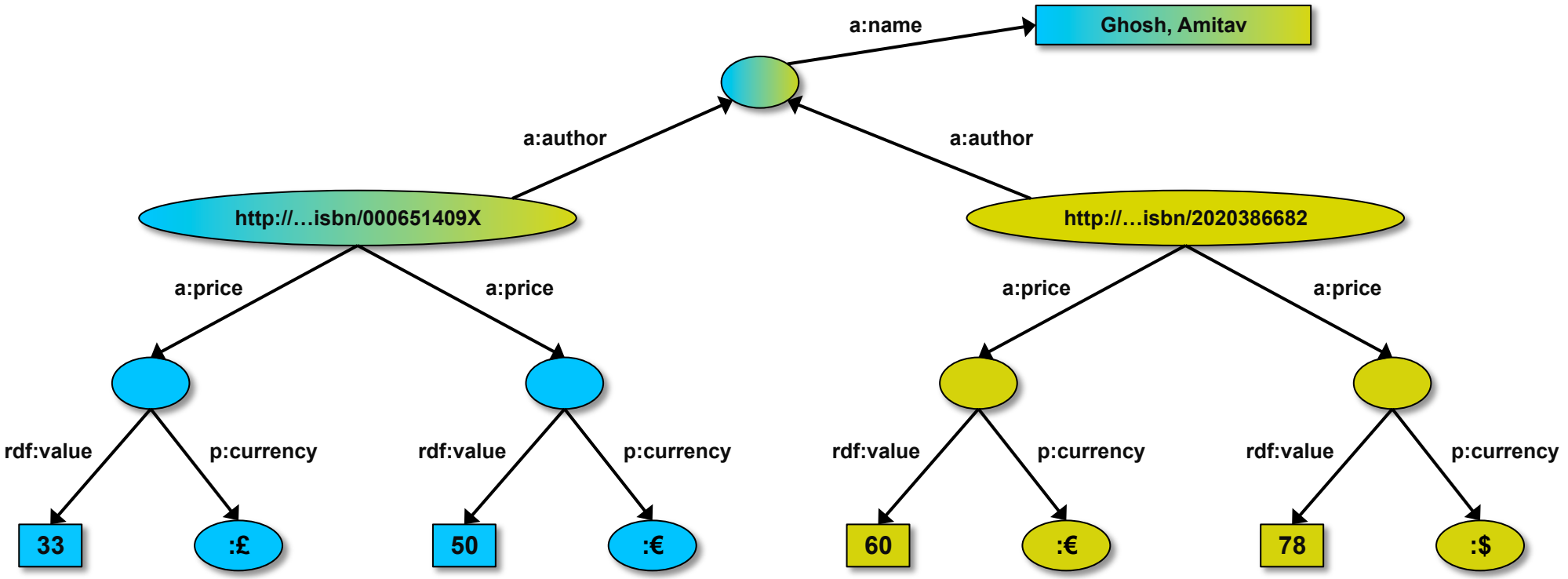
Update: insert

```
INSERT {?isbn rdf:type frbr:Work}
WHERE  {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



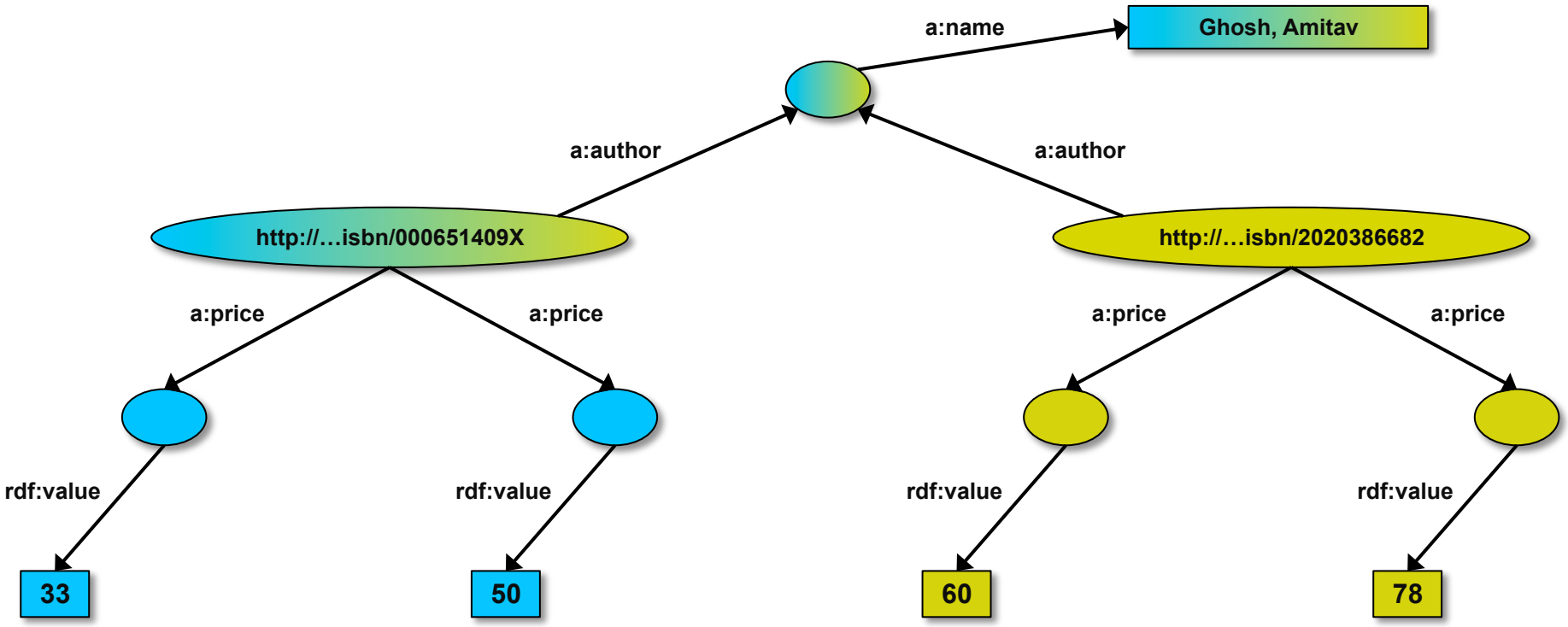
Update: delete

```
DELETE {?x p:currency ?currency}
WHERE  {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```

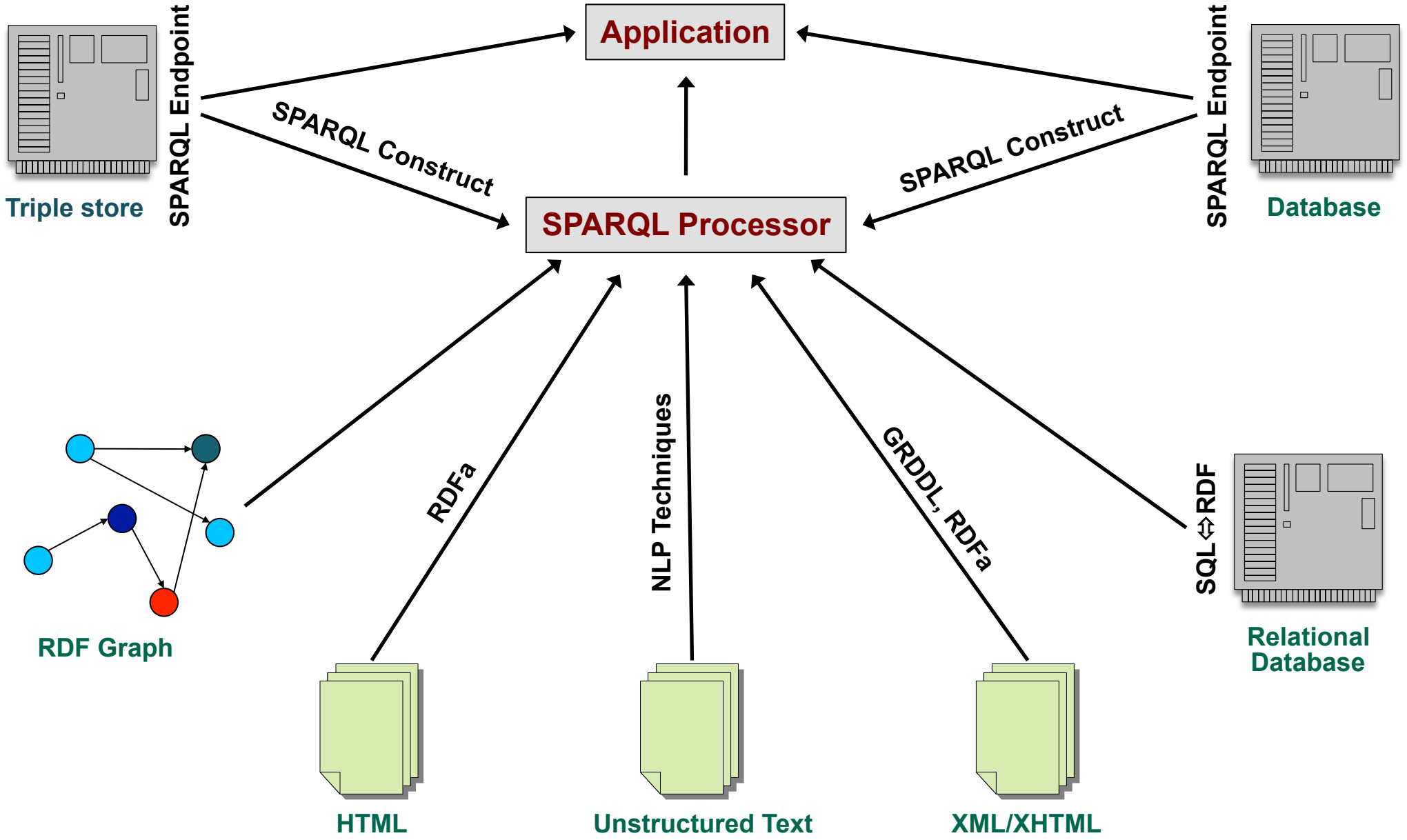


Update: delete

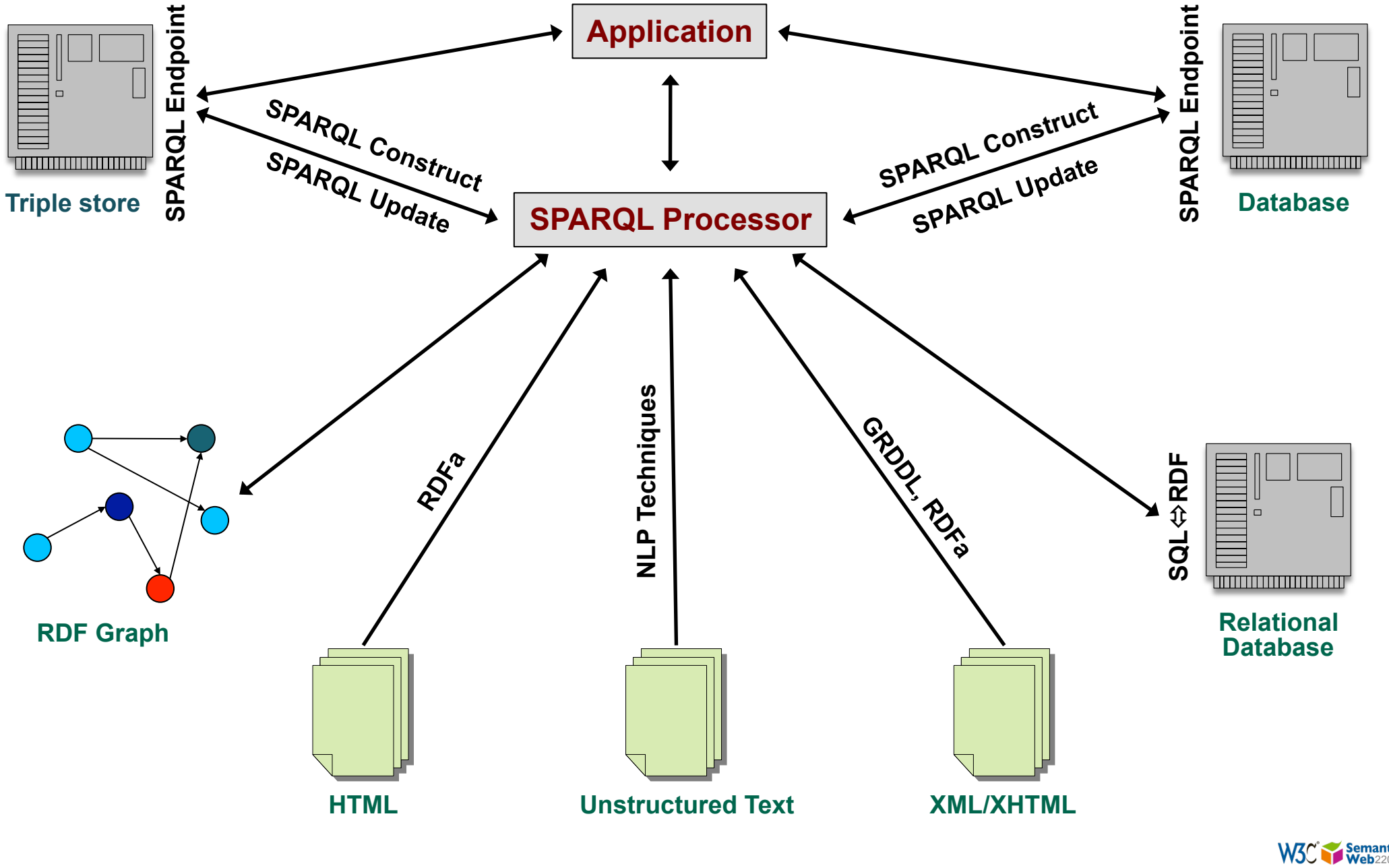
```
DELETE {?x p:currency ?currency}
WHERE  {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



SPARQL as a unifying point



SPARQL 1.1 as a unifying point



Remember the Direct Mapping issue?

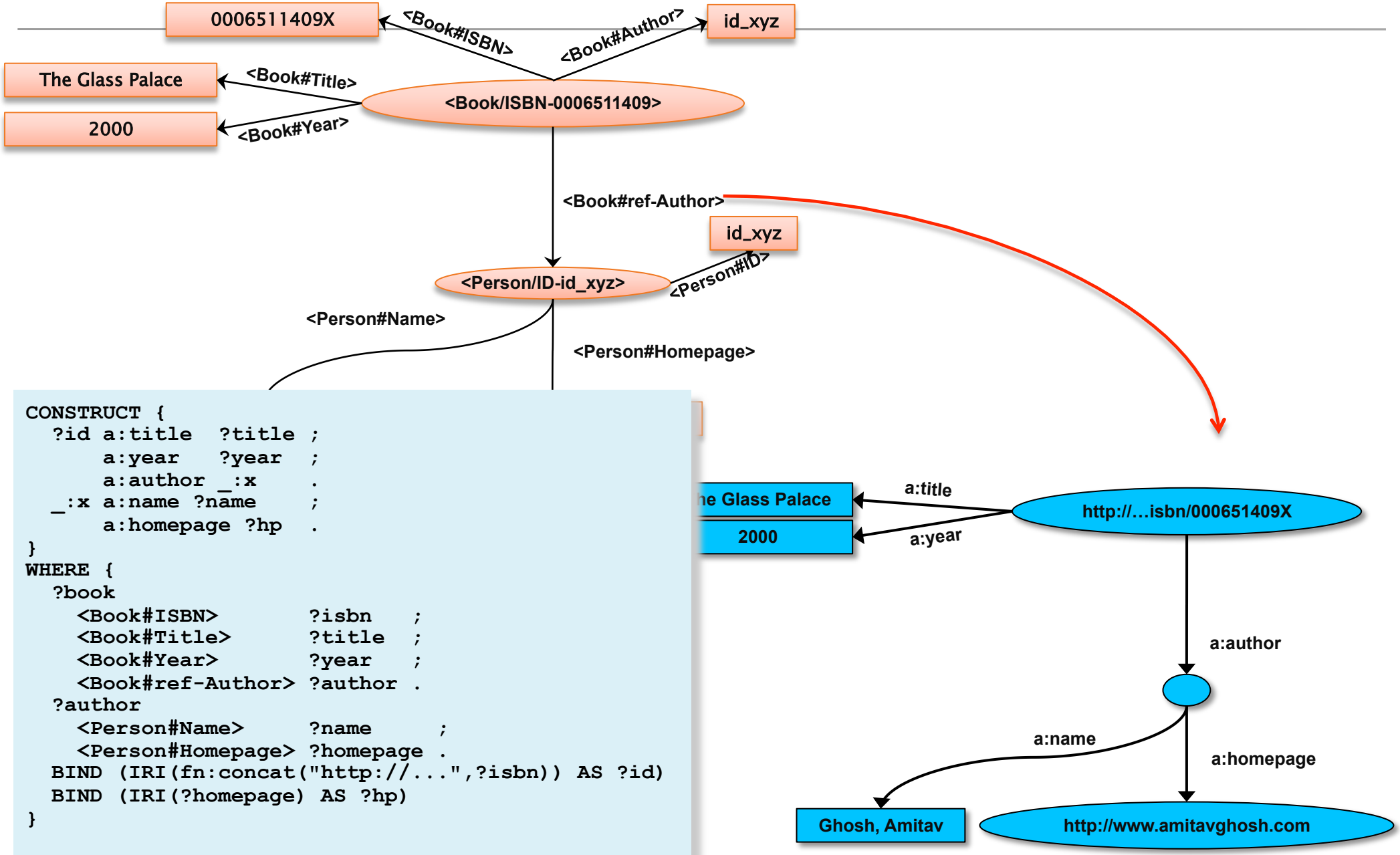


Photo credit "Indy Reading Coalition", Wordpress.com

Vocabularies

- Data integration needs agreements on
  - terms
    - “translator”, “author”
  - categories used
    - “Person”, “literature”
  - relationships among those
    - “an author is also a Person…”, “historical fiction is a narrower term than fiction”
    - ie, new relationships can be deduced

Vocabularies

- There is a need for “languages” to define such vocabularies
  - to define those vocabularies
  - to assign clear “semantics” on how new relationships can be deduced

But what about RDFS?

- Indeed RDFS *is* such framework:
  - there is typing, subtyping
  - properties can be put in a hierarchy
  - datatypes can be defined
- RDFS is enough for many vocabularies
  - we have seen some of those…
- But not for all!

Three technologies have emerged

- To re-use thesauri, glossaries, etc: SKOS
- To define more complex vocabularies with a strong logical underpinning: OWL
- Generic framework to define rules on terms and data: RIF

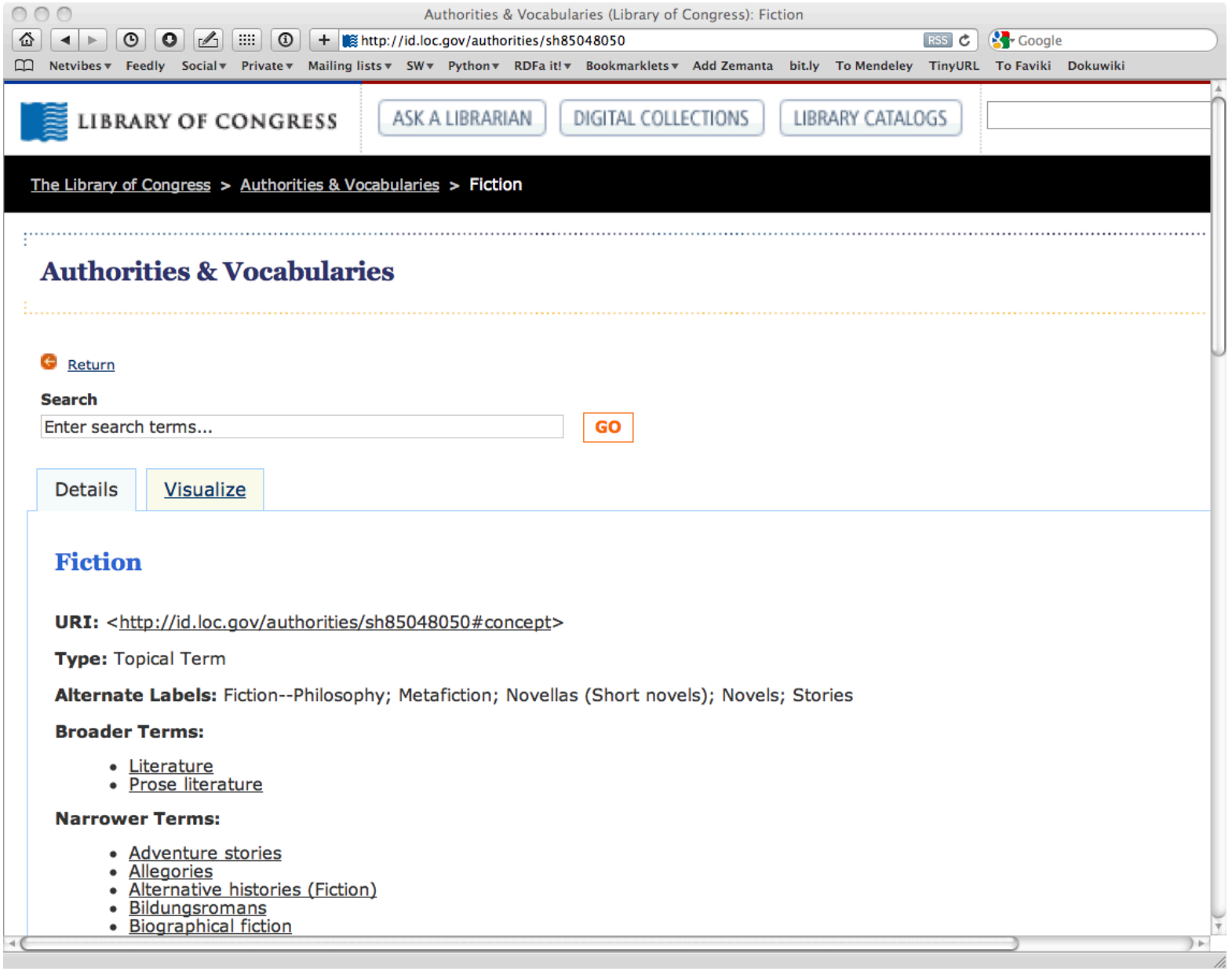
Thesauri, glossaries (SKOS)

Photo credit "scarletgreen", Flickr

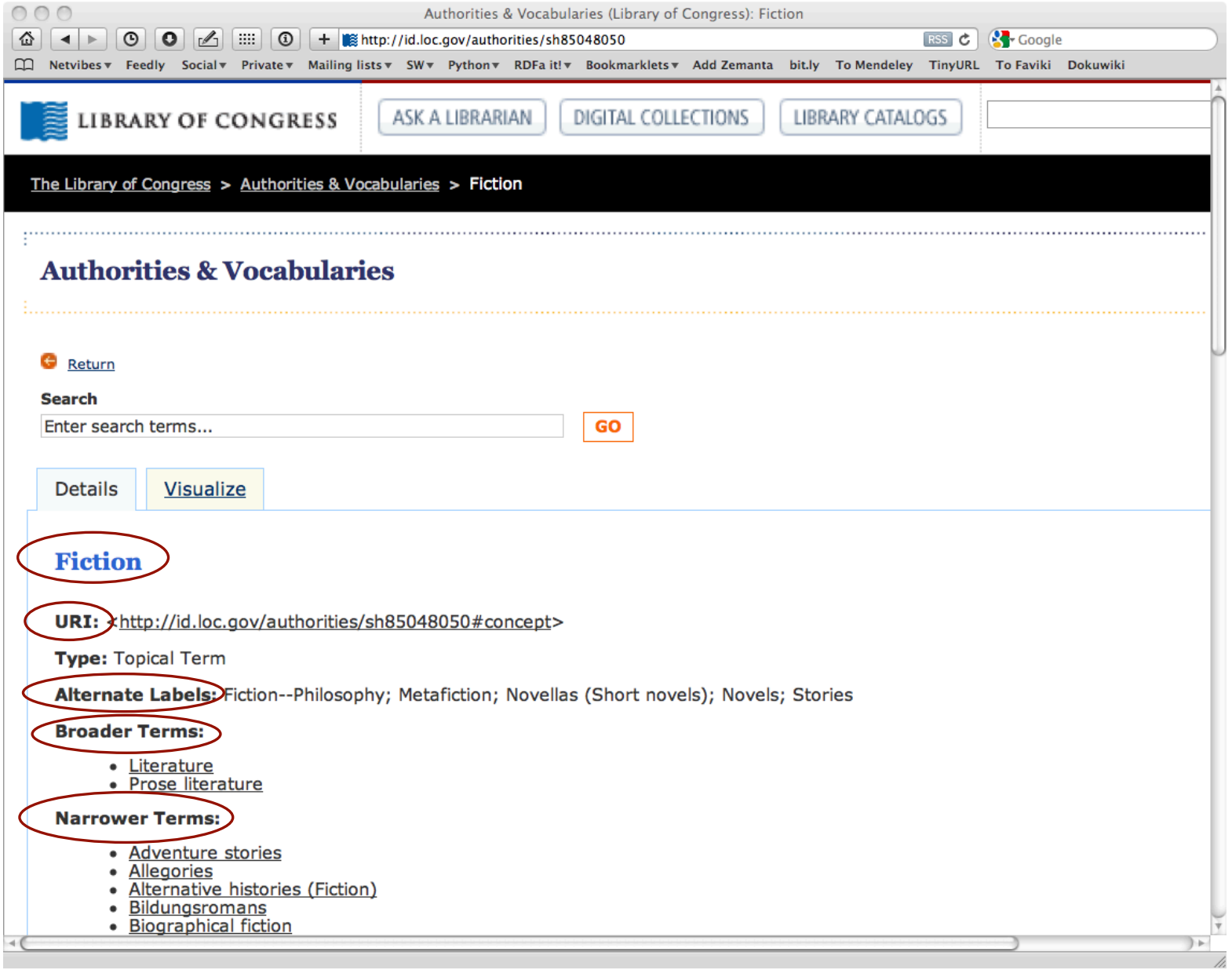
SKOS

- Represent and share classifications, glossaries, thesauri, etc
  - for example:
    - Dewey Decimal Classification, Art and Architecture Thesaurus, ACM classification of keywords and terms…
    - classification/formalization of Web 2.0 type tags
- Define classes and properties to add those structures to an RDF universe
  - allow for a quick port of this traditional data, combine it with other data

Example: the term “Fiction”, as defined by the Library of Congress



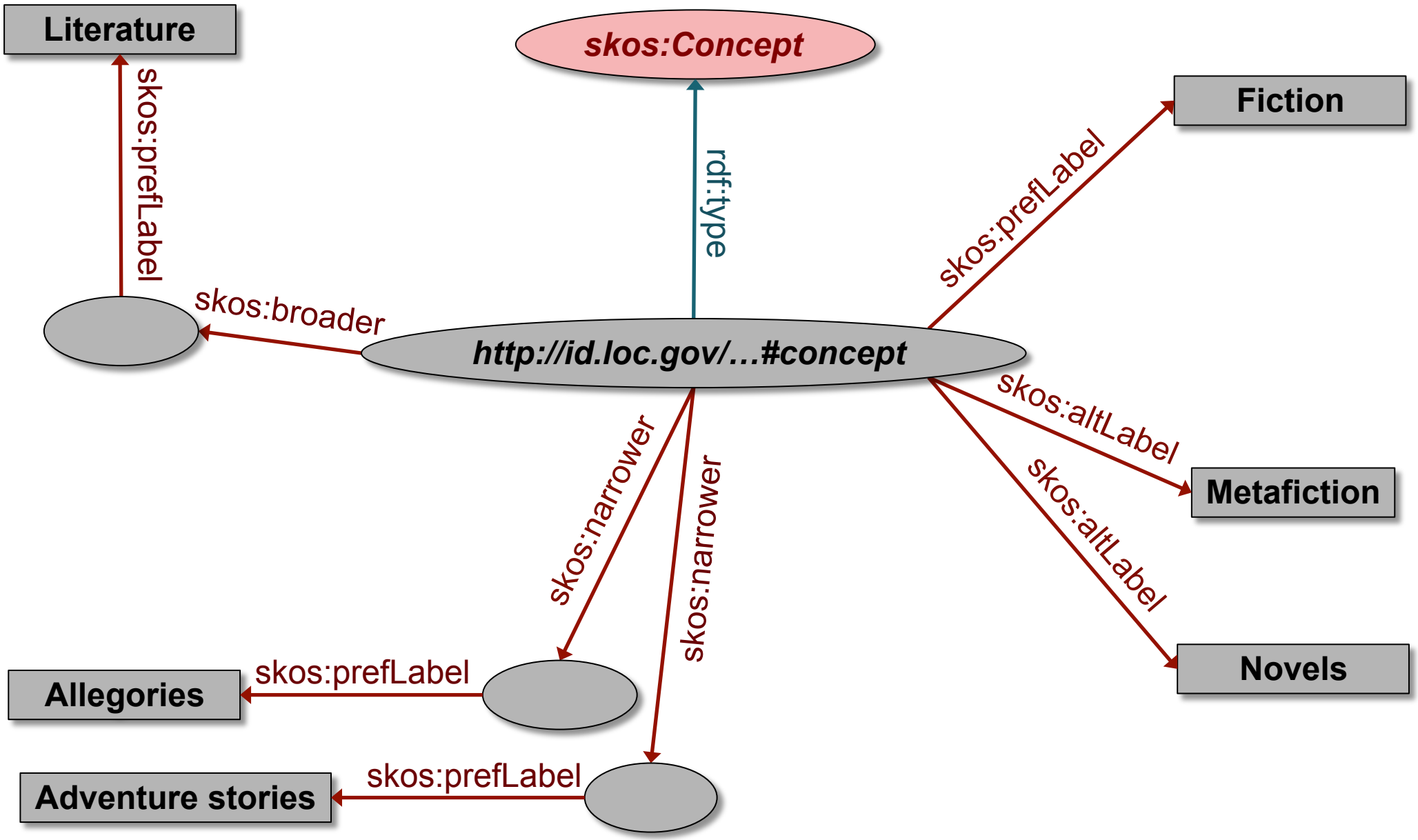
Example: the term “Fiction”, as defined by the Library of Congress



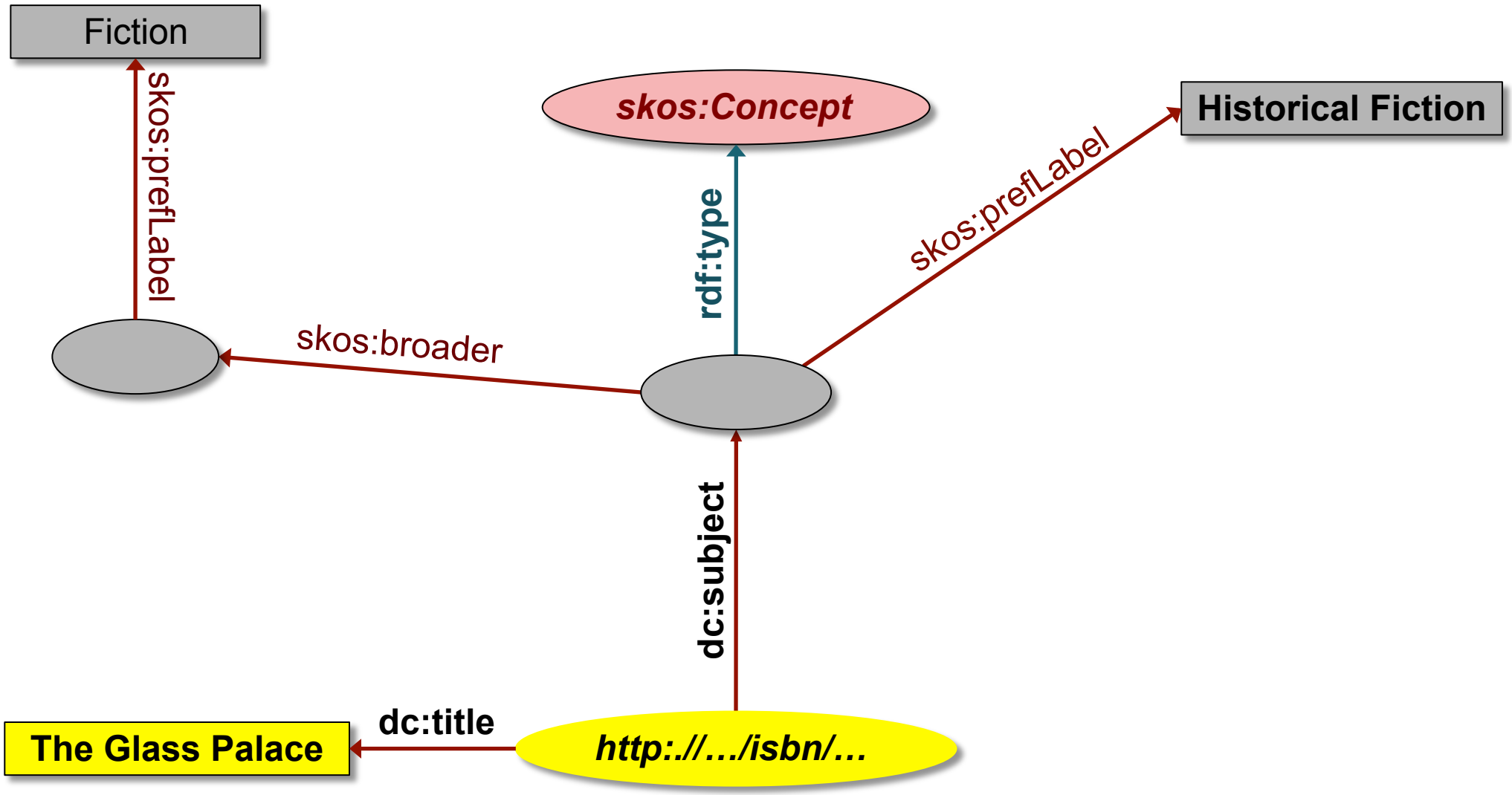
Thesauri have identical structures…

- The structure of the LOC page is fairly typical
  - label, alternate label, narrower, broader, …
  - there is even an ISO standard for these
- SKOS provides a basic structure to create an RDF representation of these

LOC's "Fiction" in SKOS/RDF



Usage of the LOC graph



Same serialized

```
<http://…/isbn/000651409X>
    dc:title "The Glass Palace"@en;
    dc:subject <http://id.loc.gov/authorities/sh85061165#concept>;
    ...

<http://id.loc.gov/authorities/sh85061165#concept>
    a    skos:Concept;
    skos:prefLabel "Historical Fiction"@en;
    skos:broader <http://id.loc.gov/authorities/sh85048050#concept>;
    ...

<http://id.loc.gov/authorities/sh85048050#concept>
    a skos:Concept;
    skos:prefLabel "Fiction"@en;
    skos:narrower <http://id.loc.gov/authorities/sh85061165#concept>;
    ...
```

SKOS terms overview

- Classes and Properties:
  - Basic description (Concept, ConceptScheme,…)
  - Labeling (prefLabel, altLabel,…)
  - Documentation (definition, historyNote,…)
  - Semantic relations (broader, narrower, related,…)
  - Collections (Collection, OrderedCollection,…)
  - Concept mappings (broadMatch, narrowMatch,…)

Importance of SKOS

- SKOS provides a simple bridge between the “print world” and the (Semantic) Web
- Thesauri, glossaries, etc, from the library community can be made available
  - LOC is a good example
- SKOS can also be used to organize, eg, tags, annotate other vocabularies, …

Importance of SKOS

- Anybody in the World can refer to common concepts
  - they mean the same for everybody
- Applications may exploit the relationships among concepts
  - eg, SPARQL queries may be issued on the library data+LOC

Photo credit "Rickydavid", Flickr

SKOS is not enough…

- SKOS may be used to provide simple vocabularies
- But it is not a complete solution
  - it concentrates on the concepts only
  - no characterization of properties in general
  - simple from a logical perspective
    - i.e., only a few inferences are possible

Application may want more…

- 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"

- The term ontologies is used in this respect:
  - Ie, there is a need for Web Ontology Languages
  - 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”) published in 2009
  - this tutorial presupposes OWL 2

OWL is complex…

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

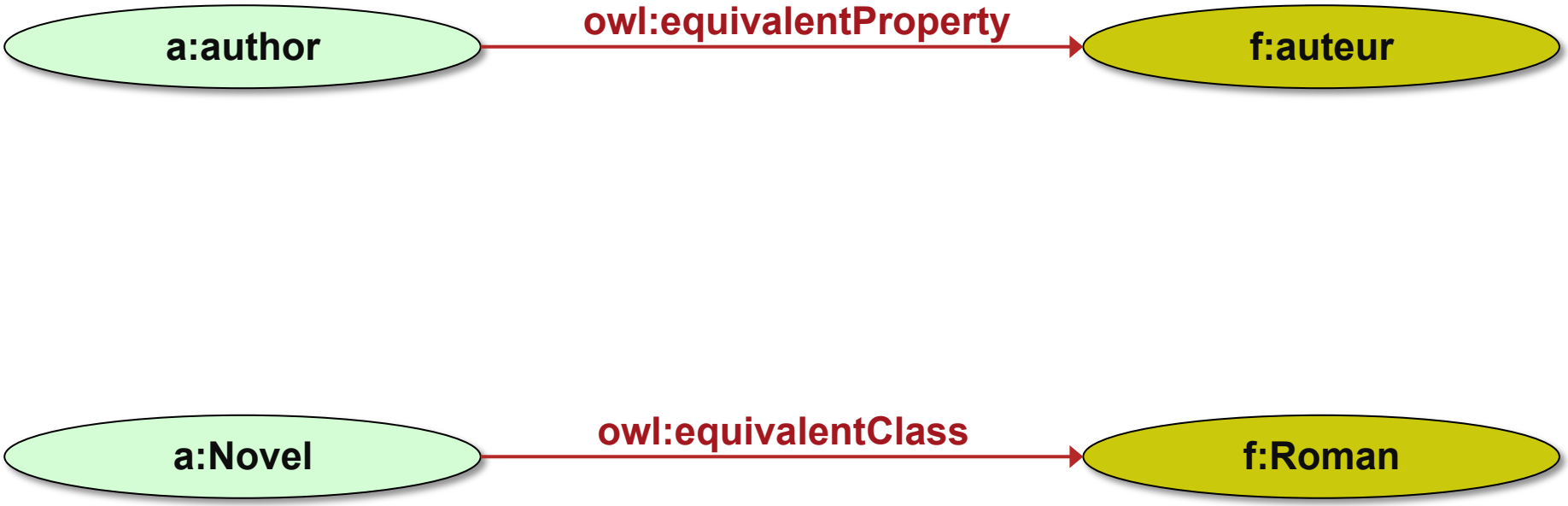
Term equivalences

- 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

Term equivalences

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

Other example: 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>;
```

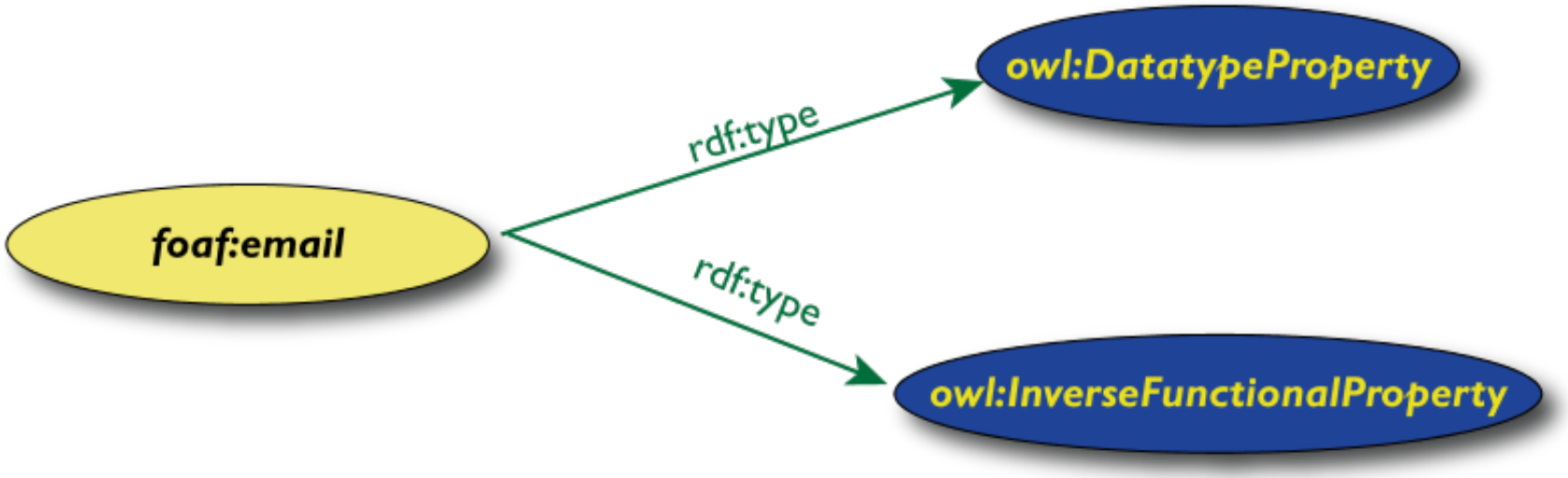
- This is the main mechanism of “Linking” in the Linked Open Data project

Property characterization

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

Characterization example

- “foaf:email” may be defined as “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.
```

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".
```

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>.
```

Inverse properties

- There may be an inverse relationship among properties, eg:

```
<somebook> ex:author <somebody>.
ex:author owl:inverseOf ex:authorOf.
```

yields, in OWL:

```
<somebody> ex:authorOf <somebook>.
```

Property chains

- 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

- 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

"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

```
: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, 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

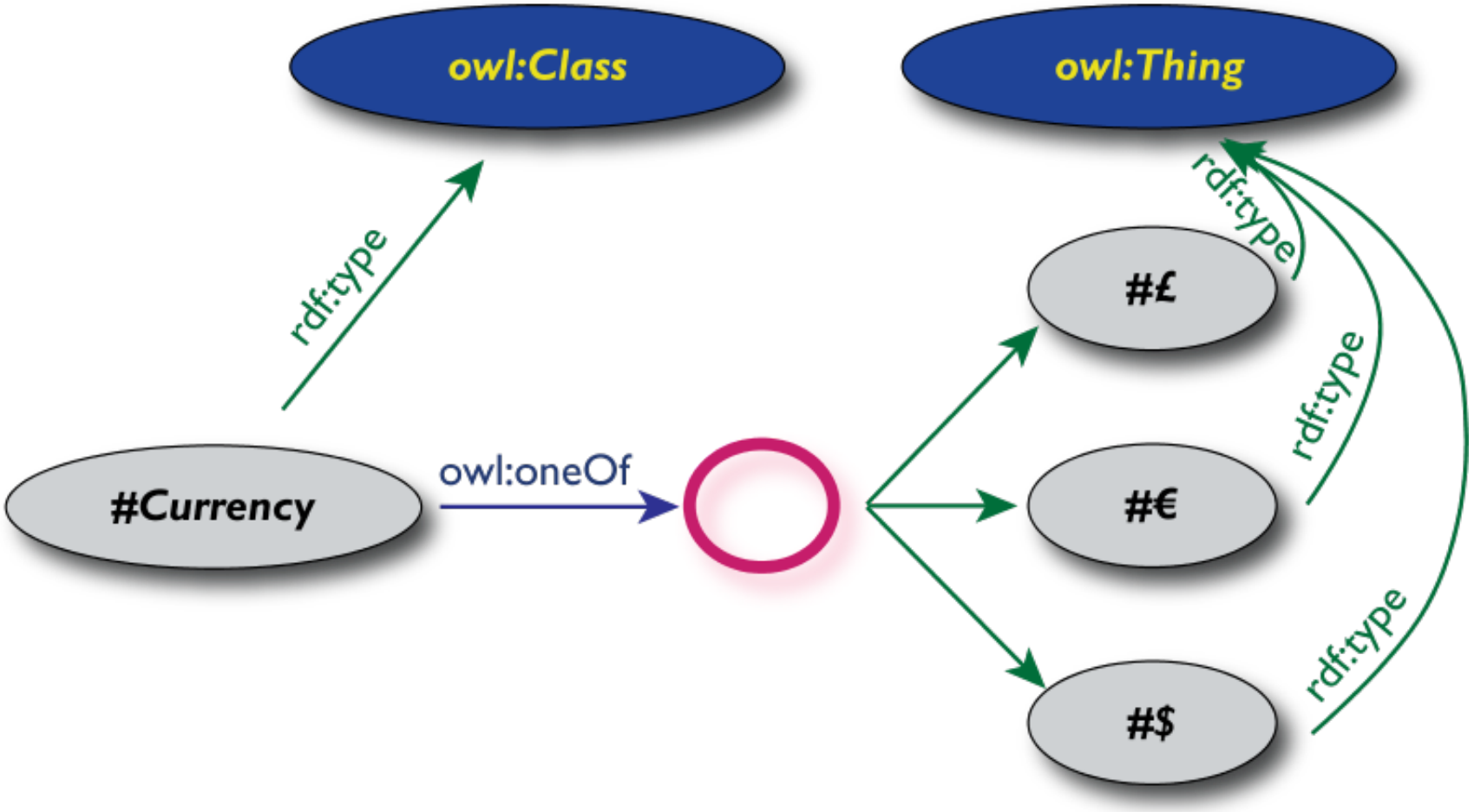
Classes in OWL (cont)

- OWL makes a stronger conceptual distinction between classes and individuals
  - there is a separate term for owl:Class, to make the difference
  - 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 .
```

OWL classes can be “enumerated”

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



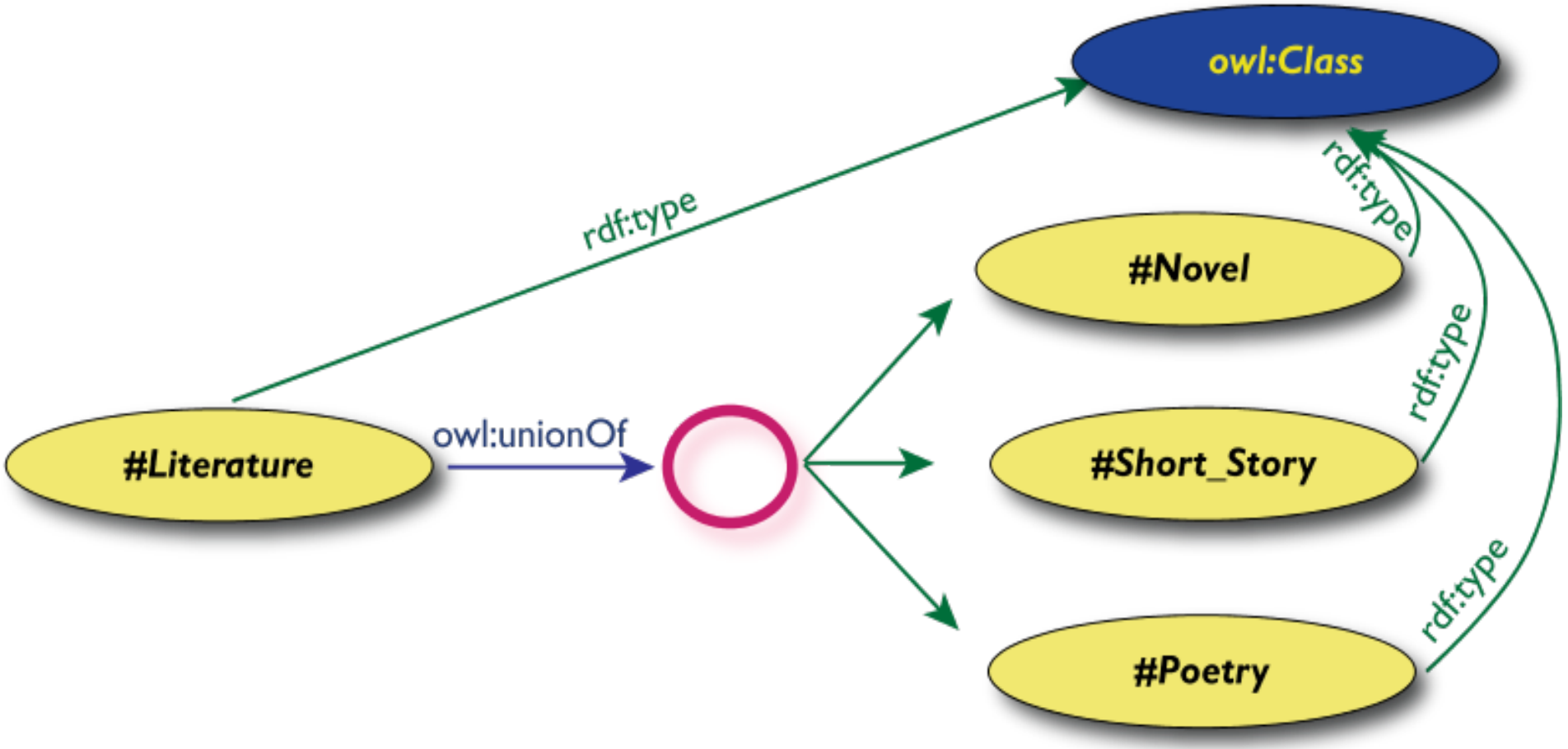
Same serialized

```
<owl:Class rdf:ID="Currency">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:ID="£"/>
    <owl:Thing rdf:ID="€"/>
    <owl:Thing rdf:ID="$"/>
    …
  </owl:oneOf>
</owl:Class>
```

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

Union of classes

- Essentially, like a set-theoretical union:



Same serialized

```
<owl:Class rdf:ID="Literature">
   <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#Novel"/>
       <owl:Class rdf:about="#Short_Story"/>
       <owl:Class rdf:about="#Poetry"/>
       …
   </owl:unionOf>
</owl:Class>
```

```
: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).
```

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 relationship 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
  - some major issues
    - the way classes (i.e., “concepts”) are defined
    - handling of datatypes
- 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 (super)class
- 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 (€, £, 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 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 <someCurrency> .
```

then the following holds:

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

Other restrictions

- owl:allValuesFrom could be replaced by:
  - owl:someValuesFrom
    - e.g., I could have said: there should be a price given in at least one of those currencies
  - owl:hasValue, when restricted to one specific value
  - owl:hasSelf, for local reflexivity

Similar concept: cardinality restriction

- In a property restriction, the goal was to restrict the possible values of a property
- In a cardinality restriction, the number of relations with that property is restricted
  - “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 rdfs:subClassOf [
      rdf:type              owl:Restriction;
      owl:onProperty        p:price;
      owl:minCardinality    "1"^^xsd:integer.
   ].
```

- could also be “owl:cardinality” or “owl:maxCardinality”

Qualified Cardinality Restriction

- Combining cardinality and the “all value” restriction
  - “there should be exactly two listed price tags with currency value”

```
:Listed_Price rdf:type owl:Class;
    rdfs:subClassOf [
      rdf:type                    owl:Restriction;
      owl:onProperty              p:currency;
      owl:onClass                 :Currency;
      owl:qualifiedCardinality    "2"^^xsd:integer.
   ].
```

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”

Datatype restrictions

- For each datatype, there are possible restriction “facets”: min and max for numeric types, length for strings, etc
- These facets can be used to define new datatypes

Definition of a numeric interval

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

Typical usage of 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 ]
           )
      ]
   ].
```

i.e.: an affordable book's price is between 5.0 and 15.0

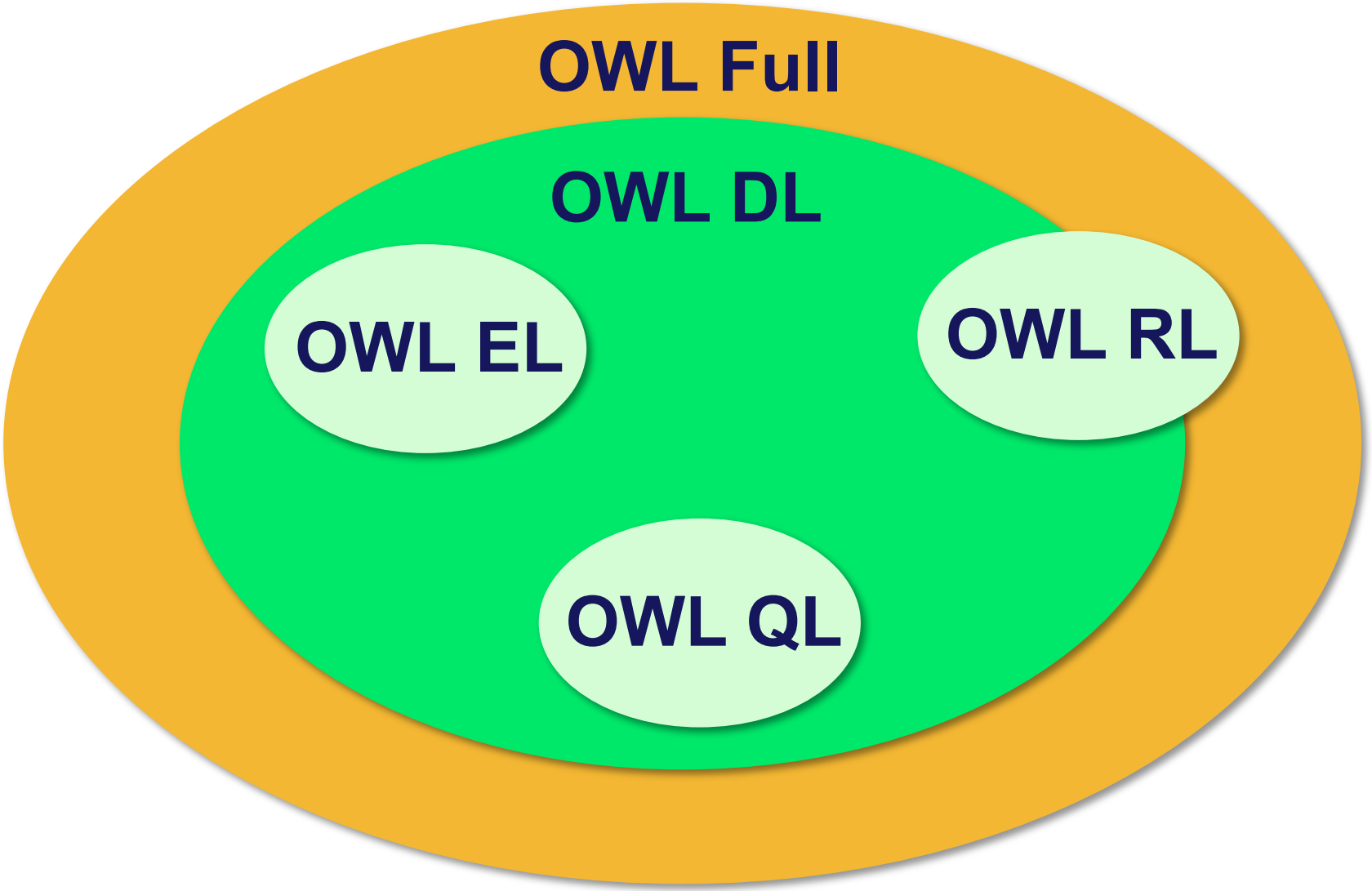
But: OWL is hard!

- 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” or profiles

- 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: expressiveness vs. implementability

OWL Species



OWL Full

- No constraints on any of the constructs
  - owl:Class is equivalent to 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.
- Extension of RDFS in all respects
- But: an OWL Full ontology may be, eg, inconsistent!

Example for a possible OWL Full problem

- Here is a syntactically valid but inconsistent ontology:

```
:A rdf:type owl:Class;
   owl:equivalentClass [
      rdf:type           owl:Restriction;
      owl:onProperty     rdf:type;
      owl:allValuesFrom :B.
   ].
:B rdf:type owl:Class;
   owl:complementOf :A.

:C rdf:type :A .
```

if **c** is of type **A** then it must be in **B**, but then it is in the complement of **A**, ie, it is not of type **A**…

OWL Full usage

- Nevertheless OWL Full is important
  - it gives a generic framework to express many things
- Some application just need to express and interchange terms (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 (eg, SKOS!)
  - thereby ensuring a manageable inference procedure

OWL DL

- A number of restrictions are defined
  - RDFS and OWL terms are reserved
    - no statements on RDFS and OWL resources
  - user's object properties must be among individuals only
  - no characterization of datatype properties
  - the same symbol, when used for an individual and a class does not mean full identity
  - ...
- But: well known inference algorithms exist!

Examples for restrictions

- Given the statement:

```
<q> rdf:type <A>.              # A is a class, q is an individual
```

the followings are not “legal” OWL DL:

```
<A> ex:something <B>.          # properties are for individuals only

<q> ex:something <s>.          # same property cannot be used as object…
<p> ex:something "54".         # … and datatype property
```

Example for restriction on conceptual identity

- Same symbol may be used both for a class and an instance
- But not all “natural” inferences can be drawn
  - eg, although the following is ok:

```
q rdf:type A.          # A is a class, q is an individual
A owl:sameAs B.        # A and B are equals as individuals
```

when using OWL DL, this does *not* yield

```
q rdf:type B.
```

“DL” stands for “Description Logic”

- An area in knowledge representation
  - a special type of “structured” First Order Logic (logic with safety guards…)
  - formalism based on “concepts” (i.e., classes), “roles” (i.e., properties), and “individuals”

OWL DL and Description Logic

- OWL DL can be interpreted as a variant of Description Logic
  - for connoisseurs: OWL (2) DL ≈ $\mathcal{SROIQ}$(D)
- Hence the results of this particular area of logic are directly applicable

Description Logic Formalism

- Traditional DL has its own terminology:
  - named objects or concepts ⇔ definition of classes, relationships among classes
  - roles ⇔ properties
  - (terminological) axioms ⇔ subclass and subproperty relationships
  - facts or assertions ⇔ statements on individuals (owl:Thing-s)

Description Logic Formalism

- There is also a compact mathematical notation for axioms, assertions, etc:
  - Literature $\equiv$ Novel $\sqcup$ Short_Story $\sqcup$ Poetry
  - Listed_Price $\sqsubseteq \forall$ currency.Currencies
- You may see these in papers, books…

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 also defines “profiles”

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

OWL 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 (beyond those of DL)
  - no cardinality restrictions, fewer property restrictions
  - no inverse, reflexive, disjoint, symmetric, asymmetric, functional or inverse functional properties
  - class disjunction
  - …

OWL 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 (beyond those of DL)
  - functional and inverse functional properties, sameAs, keys
  - fewer property restrictions
  - no cardinality restrictions
  - transitive properties, property chains
  - …

OWL 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
  - …

Some more on OWL RL

- Goal: to be implementable through rule engines
- Usage follows a similar approach to RDFS:
  - merge the ontology and the instance data into an RDF graph
  - use the rule engine to add new triples (as long as it is possible)
- This application model is very important for RDF based applications

What can be done in OWL RL?

- Many features are available:
  - identity of classes, instances, properties
  - subproperties, subclasses, domains, ranges
  - union and intersection of classes (though with some restrictions)
  - property characterizations (functional, symmetric, etc)
  - property chains
  - keys
  - some property restrictions (but not all inferences are possible)
- All examples so far could be inferred with OWL RL!

What cannot be done in OWL RL?

- There are restrictions on what can be a sub and superclass. Eg, the following is not manageable:

```
B rdf:type owl:Class;
  owl:unionOf (P Q R).

A rdfs:subClassOf B .
```

- Some features are not available or are restricted:
  - not all datatypes are available
  - no datatype restrictions
  - no minimum or exact cardinality restrictions
  - maximum cardinality only 0 or 1

What cannot be done in OWL RL?

- Some “natural” conclusions cannot be drawn, eg:

```
A rdf:type owl:Class;
  owl:intersectionOf (U V S).
```

does *not* yield:

```
A rdfs:subClassOf U .
```

Alternative syntaxes for OWL

- OWL constructs in RDF can be fairly verbose
- There are alternative syntaxes to express ontologies
  - direct XML encoding of ontologies (OWL/XML)
  - “functional” syntax
  - “Manchester” syntax
- The official exchange syntax is RDF (RDF/XML)
  - all other syntaxes are optional for tools

Functional syntax example

```
:£ rdf:type owl:Thing.
:€ rdf:type owl:Thing.
:$ rdf:type owl:Thing.

my:Currency rdf:type owl:Class;
    owl:oneOf (:€ :£ :$).

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

is equal to:

```
Declaration(NamedIndividual(:£))
Declaration(NamedIndividual(:€))
Declaration(NamedIndividual(:$))

Declaration(Class(:Currency))
EquivalentClasses(:Currency one of(:€ :£ :$))

SubClassOf(my:Listed_Price AllValuesFrom(p:currency my:Currency))
```

Manchester syntax example

```
:£ rdf:type owl:Thing.
:€ rdf:type owl:Thing.
:$ rdf:type owl:Thing.

my:Currency rdf:type owl:Class;
    owl:oneOf (:€ :£ :$).

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

is equal to:

```
Individual: :€
Individual: :£
Individual: :$

Class: my:Currency EquivalentTo {:€ :£ :$}

Class: my:Listed_Price that p:currency only my:Currency
```

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

Ontology development (cont.)

- Ontologies should be shared and reused
  - can be via the simple namespace mechanisms…
  - …or via explicit imports
- Applications can also be developed with very small ontologies, though

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

Ginger Lemon Ice Cream

#1

1 1/2 TB cornstarch
2 TB milk

#2

Zest 2 lemons

#3

GINGER SYRUP:
1/3 cup water
1/3 cup sugar
3 TB grated fresh ginger

#4

ICE CREAM BASE:
2 cups milk
1.5 cups heavy cream
2/3 cup honey/sugar
2 tsp minced ginger
1/8 TB light corn syrup

#5

3 TB cream cheese
1/8 tsp salt

#6

1 TB lemon juice
chopped candied
ginger pieces

Rules (RIF)

1. **Prepare #1 and #2 - set aside.**
2. **Make #3 Ginger Syrup - boil until all dissolved. Keep simmering for a few minutes. Set aside**
3. **In a medium saucepan, cook #4 on moderate to high heat until simmering.**
4. **Remove saucepan from heat, stir in #3 and #2. Return to low heat for a few minutes.**
5. **Remove saucepan from heat, add in #1. Return to low heat for a few minutes, stir until thickened.**
6. **Slowly strain into Bowl #5. Add 1/8 tsp kosher salt and whisk for a while.**
7. **Put in fridge for 20+ minutes until cool.**
8. **Churn in ice cream maker for 20 minutes or until thickened.**
9. **In the last few minutes, slowly add #6. Churn for 5 more minutes.**
10. **Transfer to container. Freeze for a few hours or overnight.**
11. **ENJOY!**

Photo credit "Lili.chin", Flickr

Why rules on the Semantic Web?

- Some conditions may be complicated in ontologies (ie, OWL)
  - e.g., Horn rules: (P1 & P2 & …) → C
- In many cases applications just want 2-3 rules to complete integration
- I.e., rules may be an alternative to (OWL based) ontologies

Rules

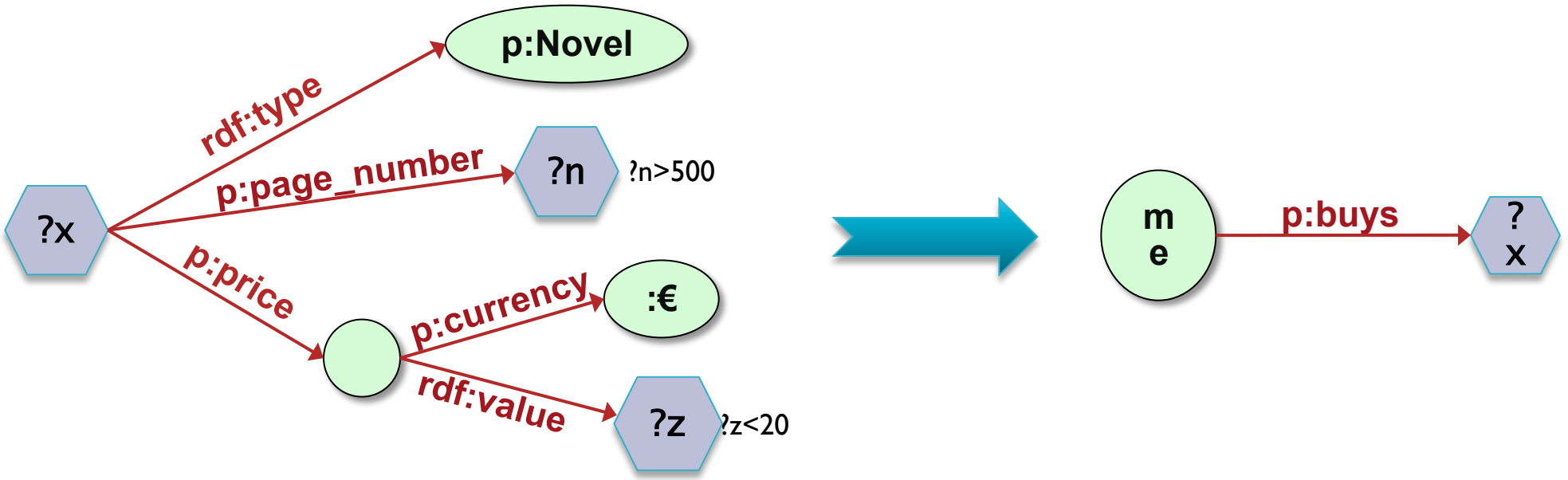
- There is also a long history of rule languages and rule-based systems
  - e.g., logic programming (Prolog), production rules
- Lots of small and large rule systems (from mail filters to expert systems)
- Hundreds of niche markets

Things you may want to express

- An example from our bookshop integration:
  - “I buy a novel with over 500 pages if it costs less than €20”
  - something like (in an ad-hoc syntax):

```
{
  ?x rdf:type p:Novel;
     p:page_number ?n;
     p:price [
         p:currency :€;
         rdf:value  ?z
     ].
  ?n > "500"^^xsd:integer.
  ?z < "20.0"^^xsd:double.
}
=>
{ <me> p:buys ?x }
```

Things you may want to express



RIF (Rule Interchange Format)

- The goals of the RIF work:
  - define simple rule language(s) for the (Semantic) Web
  - define interchange formats for rule based systems
- RIF defines several “dialects” of languages

RIF Core

- The simplest RIF “dialect”
- A Core document is
  - some directives like import, prefix settings for URI-s, etc
  - a sequence of logical implications
- RIF is not bound to RDF only
  - eg, relationships may involve more than 2 entities

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)
  )
)
```

This infers the following relationship:

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

Expressivity of RIF Core

- Formally: definite Horn without function symbols, a.k.a. “Datalog”
  - e.g., p(a,b,c) is fine, but p(f(a),b,c) is not
- Includes some extra features
  - built-in datatypes and predicates
  - notion of “local names”, a bit like RDF’s blank nodes
  - …

Expressivity of RIF Core

- There are also “safeness measures”
  - eg, variable in a consequent should be in the antecedent
    - this secures a straightforward implementation strategy (“forward chaining”)
- RIF Core is the simplest rule dialect
  - but covers a large percentage of RDF related use cases…

RIF Syntaxes

- RIF defines
  - a “presentation syntax”
    - a bit like the functional syntax for OWL
  - a standard XML syntax to encode and exchange the rules
  - there is a draft for expressing Core in RDF
    - just like OWL is represented in RDF

What about RDF and RIF?

- Typical scenario:
  - the “data” of the application is available in RDF
  - rules on that data is described using RIF
  - the two sets are “bound” (eg, RIF “imports” the data)
    - if RIF is encoded in RDF, the two sets may be in the same file
  - a RIF processor produces new relationships

To make RIF/RDF work

- Some technical issues should be settled:
  - RDF facts/triples have to be representable in RIF
  - various constructions (typing, datatypes, lists) should be aligned
  - the formal semantics of the two worlds should be compatible
- There is a separate document that brings these together

Expressing RDF triples in RIF

- RIF has a “frame-based” syntax:

```
s[p->x q->z]
```

- This is used to represent the RDF triples:

```
s p x .
s q z .
```

- The rest is fairly obvious

RIF/RDF harmonization

RDF		RIF
s p o .	⇔	s[p->o]
s rdf:type A .	⇔	A # B
(A B C)	⇔	List(A B C)
datatypes	⇔	datatypes
_:a (blank nodes)	⇔	_a (local names)

Remember the what we wanted from Rules?

```
{
  ?x rdf:type p:Novel;
     p:page_number ?n;
     p:price [
         p:currency :€;
         rdf:value  ?z
     ].
  ?n > "500"^^xsd:integer.
  ?z < "20.0"^^xsd:double.
}
=>
{ <me> p:buys ?x }
```

The same with RIF Presentation syntax

```
Document (
  Prefix …
  Group (
    Forall ?x ?n ?z (
      <me>[p:buys->?x] :-
        And(
          ?x rdf:type p:Novel
          ?x[p:page_number->?n p:price->_abc]
          _abc[p:currency->:€ rdf:value->?z]
          External(pred:numeric-greater-than(?n "500"^^xsd:integer))
          External(pred:numeric-less-than(?z "20.0"^^xsd:double))
        )
    )
  )
)
```

Discovering new relationships…

```
Forall ?x ?n ?z (
  <me>[p:buys->?x] :-
    And(
      ?x # p:Novel
      ?x[p:page_number->?n p:price->_abc]
      _abc[p:currency->:€ rdf:value->?z]
      External( pred:numeric-greater-than(?n "500"^^xsd:integer) )
      External( pred:numeric-less-than(?z "20.0"^^xsd:double) )
    )
)
```

Discovering new relationships…

```
Forall ?x ?n ?z (
  <me>[p:buys->?x] :-
    And(
      ?x # p:Novel
      ?x[p:page_number->?n p:price->_abc]
      _abc[p:currency->p:€ rdf:value->?z]
      External( pred:numeric-greater-than(?n "500"^^xsd:integer) )
      External( pred:numeric-less-than(?z "20.0"^^xsd:double) )
    )
)
```

combined with:

```
<http://…/isbn/…> a p:Novel;
    p:page_number "600"^^xsd:integer ;
    p:price [ rdf:value "15.0"^^xsd:double ; p:currency :€ ] .
```

Discovering new relationships…

```
Forall ?x ?n ?z (
  <me>[p:buys->?x] :-
    And(
      ?x # p:Novel
      ?x[p:page_number->?n p:price->_abc]
      _abc[p:currency->p:€ rdf:value->?z]
      External( pred:numeric-greater-than(?n "500"^^xsd:integer) )
      External( pred:numeric-less-than(?z "20.0"^^xsd:double) )
    )
)
```

combined with:

```
<http://…/isbn/…> a p:Novel;
    p:page_number "600"^^xsd:integer ;
    p:price [ rdf:value "15.0"^^xsd:double ; p:currency :€ ] .
```

yields:

```
<me> p:buys <http://…/isbn/…> .
```

A word on the syntax

- The RIF Presentation syntax is… only syntax
- It can express more than what RDF needs
- Hopefully, a syntax will emerge with
  - close to one of the RDF syntaxes with a better integration of rules
  - can be mapped on Core implementations

RIF vs. OWL?

- OWL concentrates on “taxonomic reasoning”
  - i.e., if you have large knowledge bases, ontologies, use OWL
- Rules concentrate on reasoning problems within the data
  - i.e., if your knowledge base is simple but lots of data, use rules
- But these are thumb rules only…

At the end of the day…

- Using rules vs. ontologies may largely depend on
  - available tools
  - personal technical experience and expertise
  - taste…

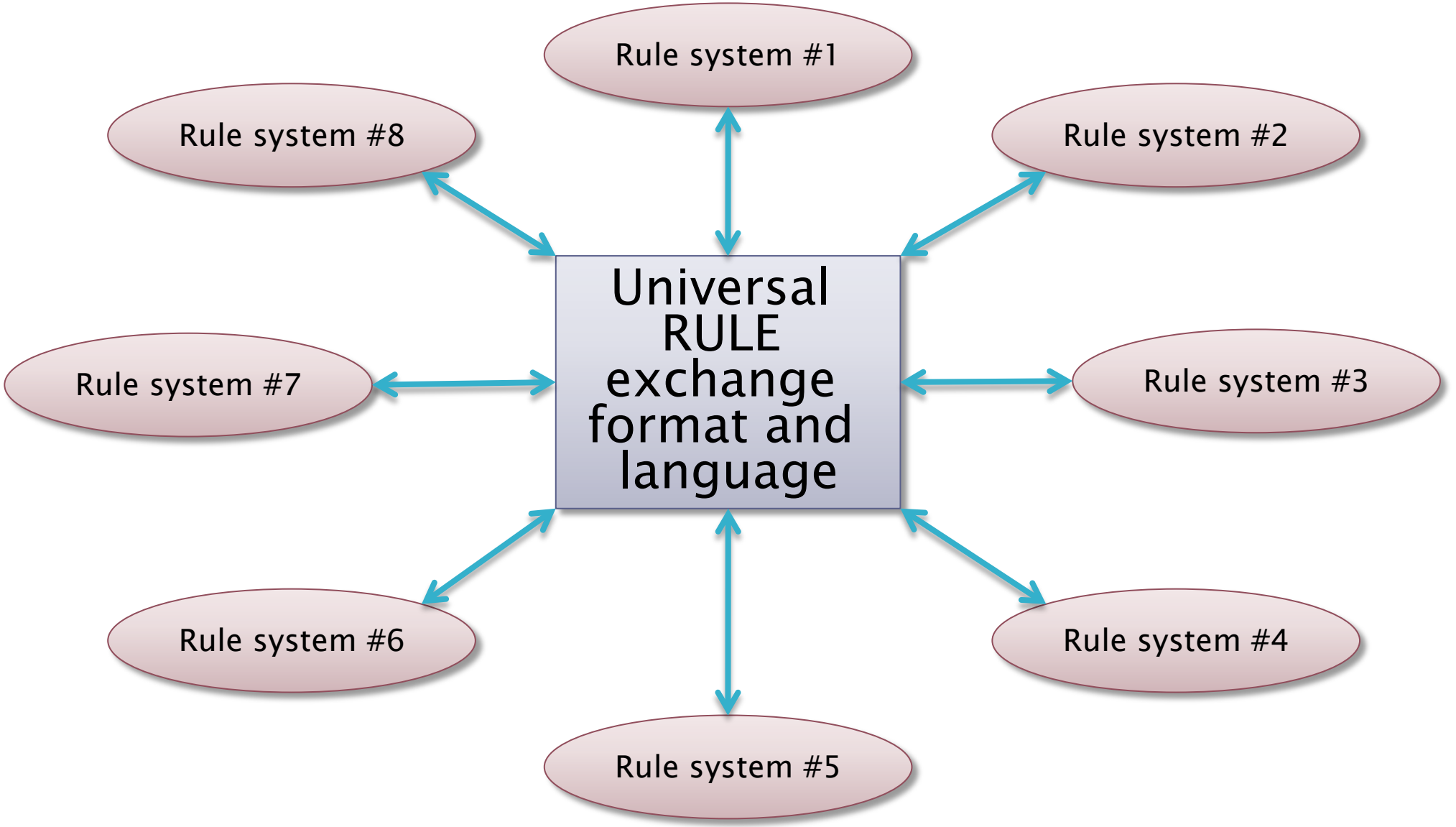
What about OWL RL?

- OWL RL stands for “Rule Language”…
- OWL RL is in the intersection of RIF Core and OWL
  - inferences in OWL RL can be expressed with rules
    - the rules are precisely described in the OWL specification
  - there are OWL RL implementations that are based on RIF

Why other RIF dialects?

- 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)

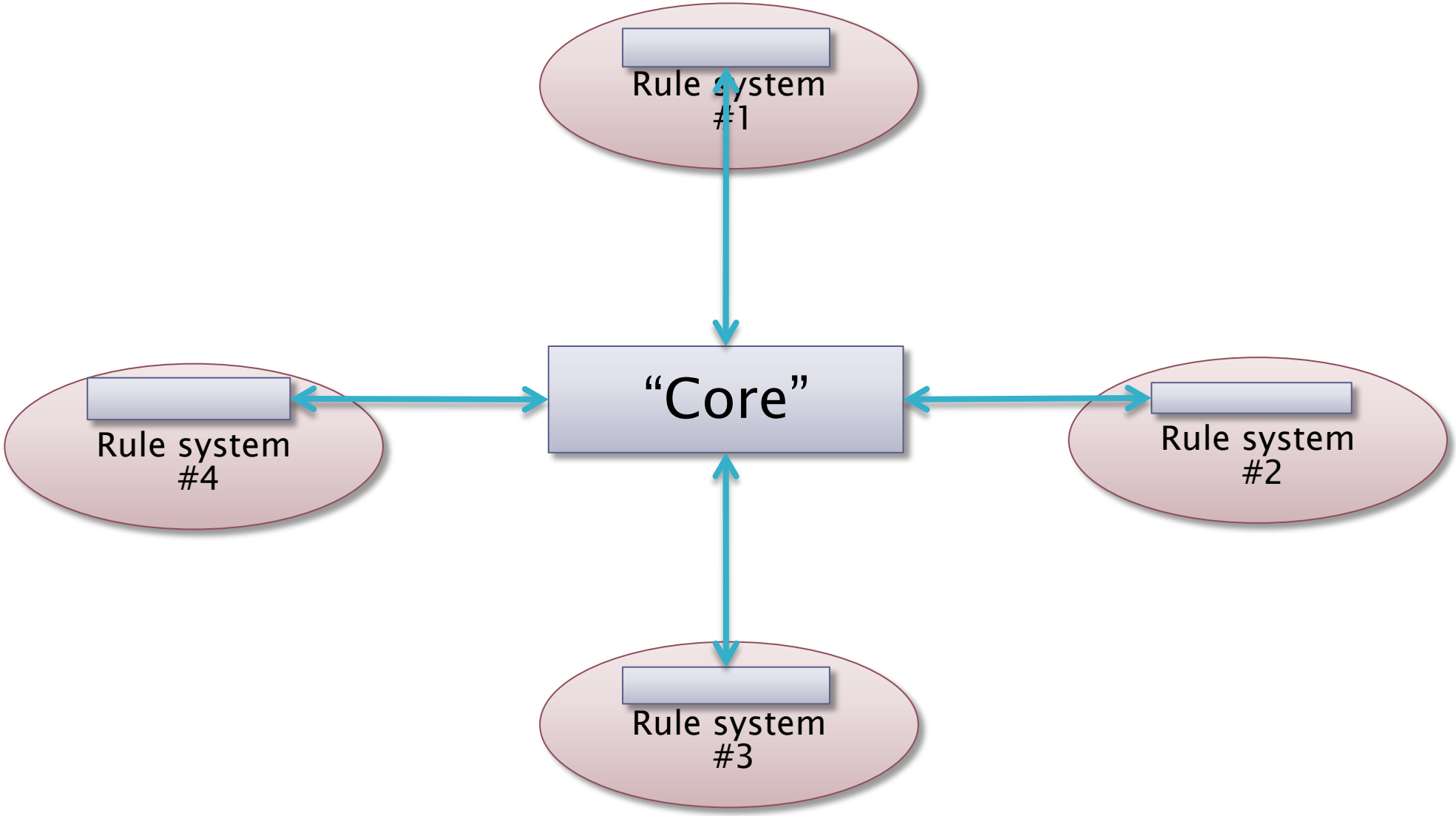
In an ideal world



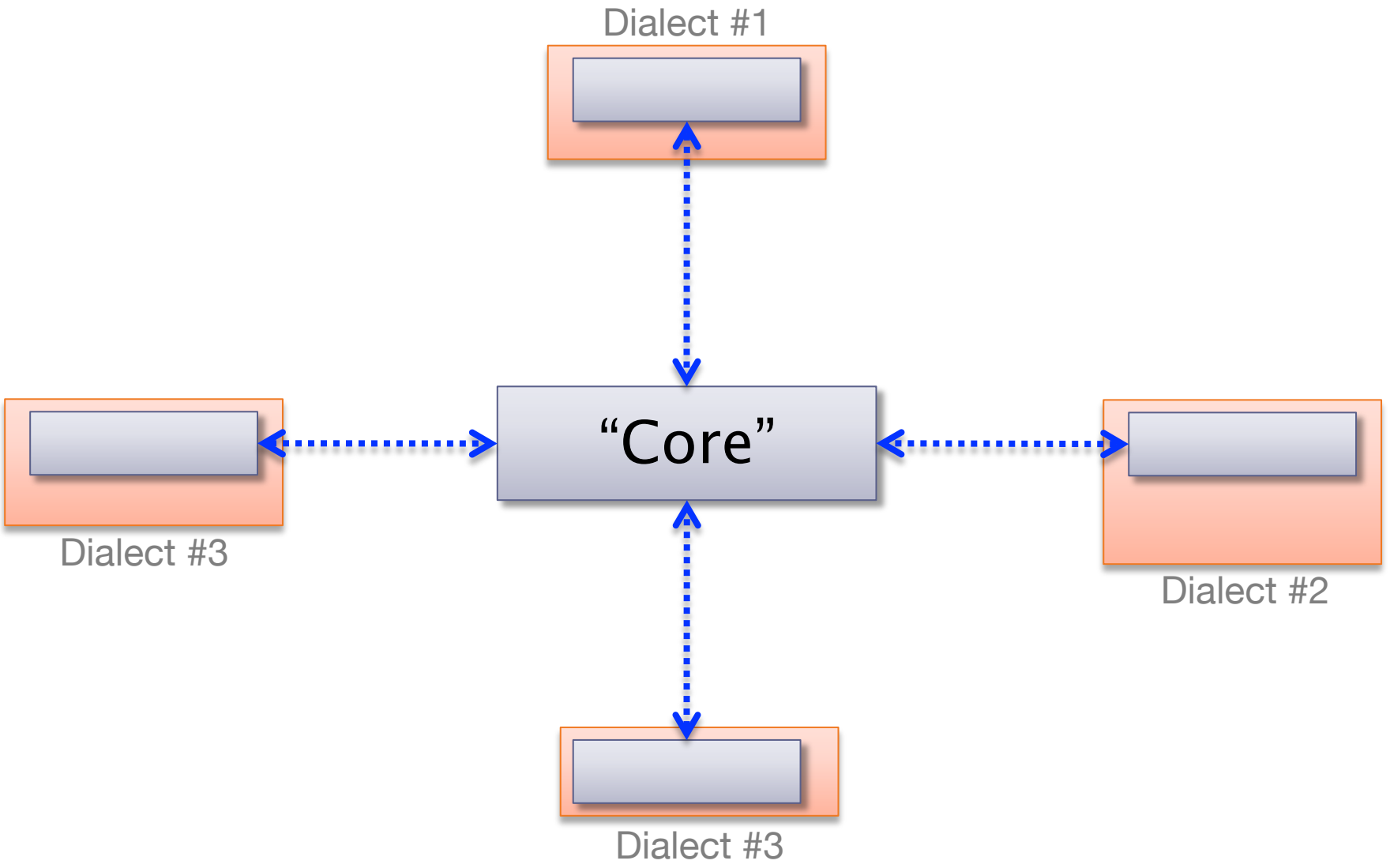
In the real World…

- 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”

A partial interchange…

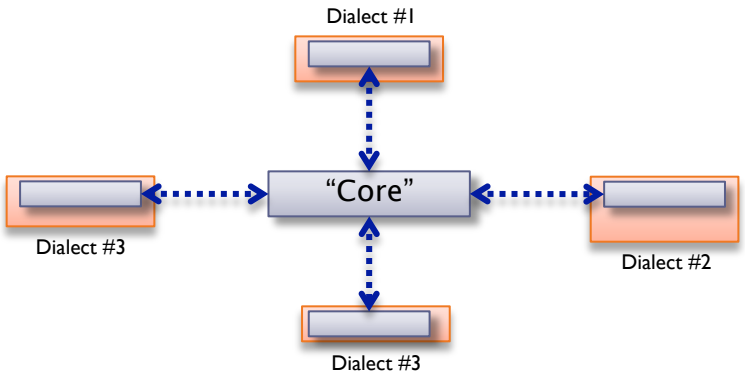


RIF “dialects”…

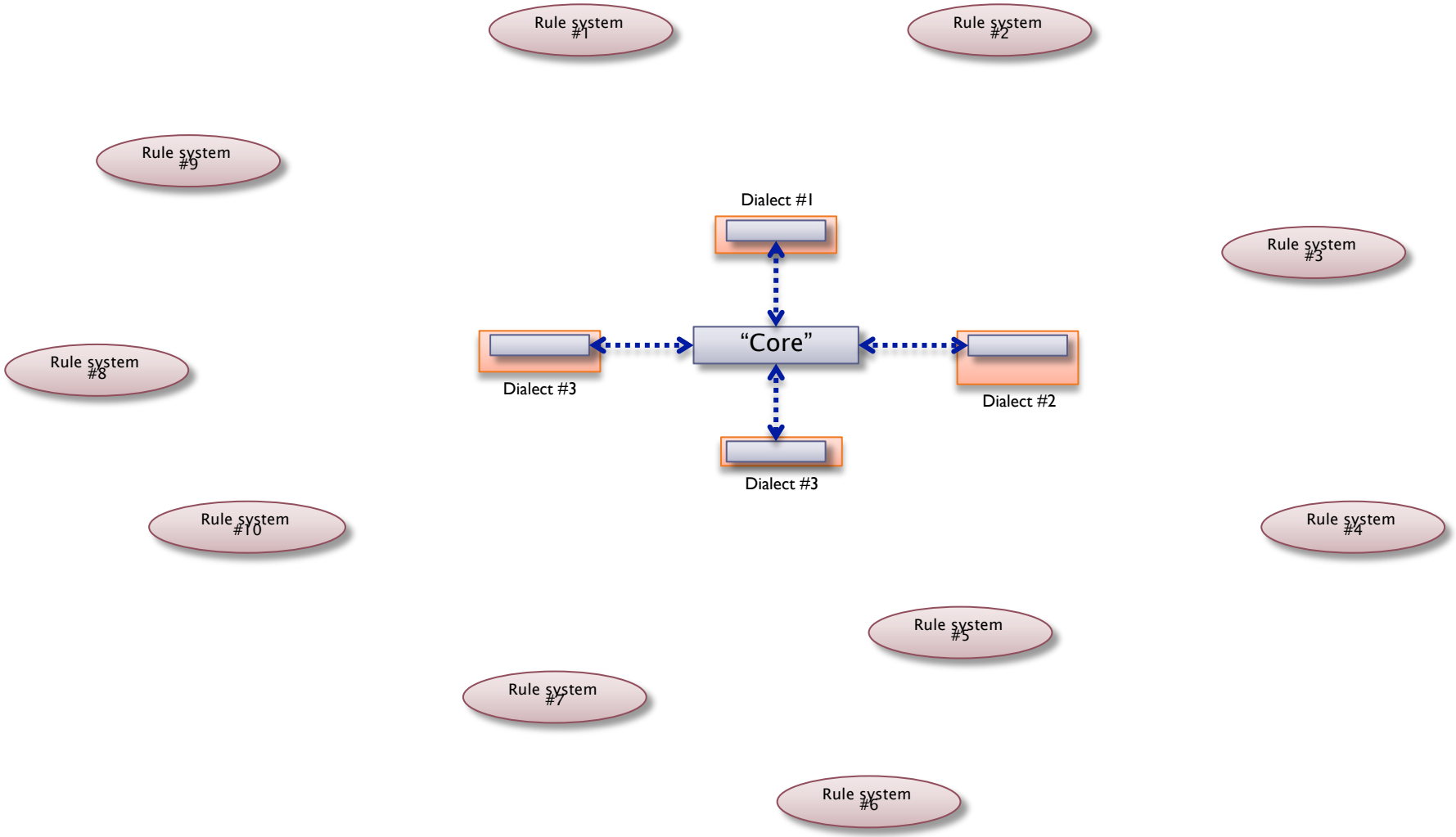


- Possible dialects: F-logic, fuzzy logic, …

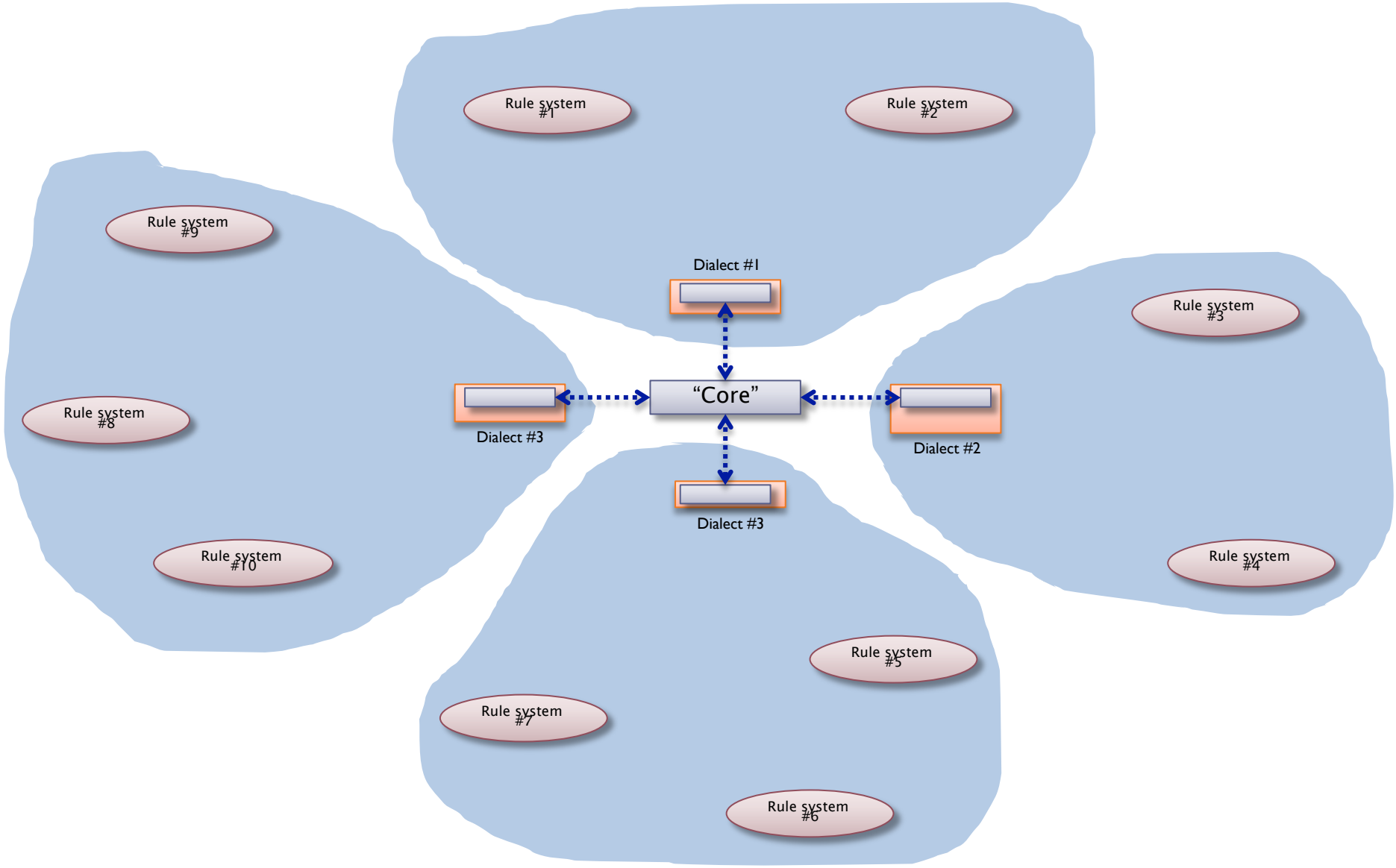
The role of dialects



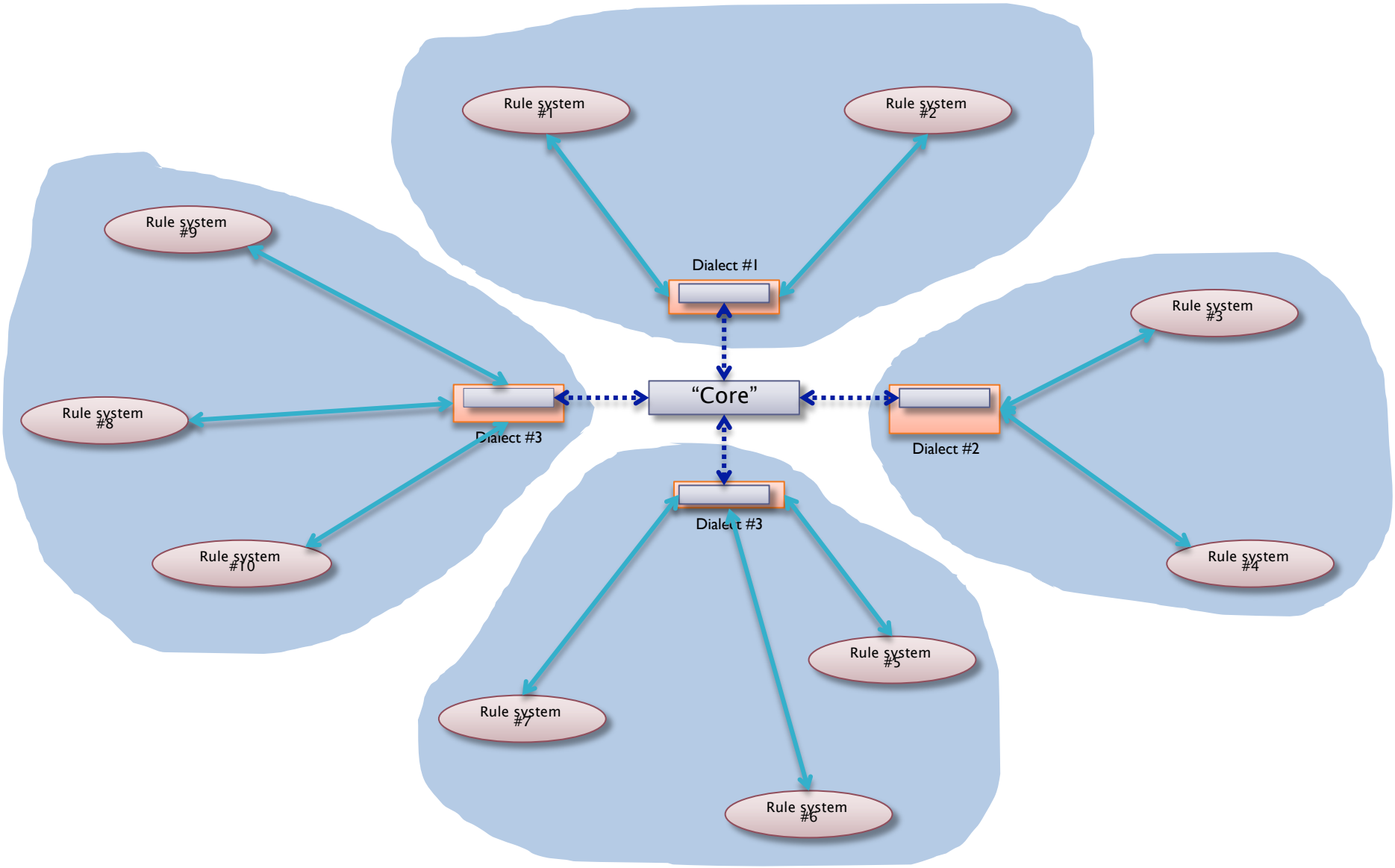
The role of dialects



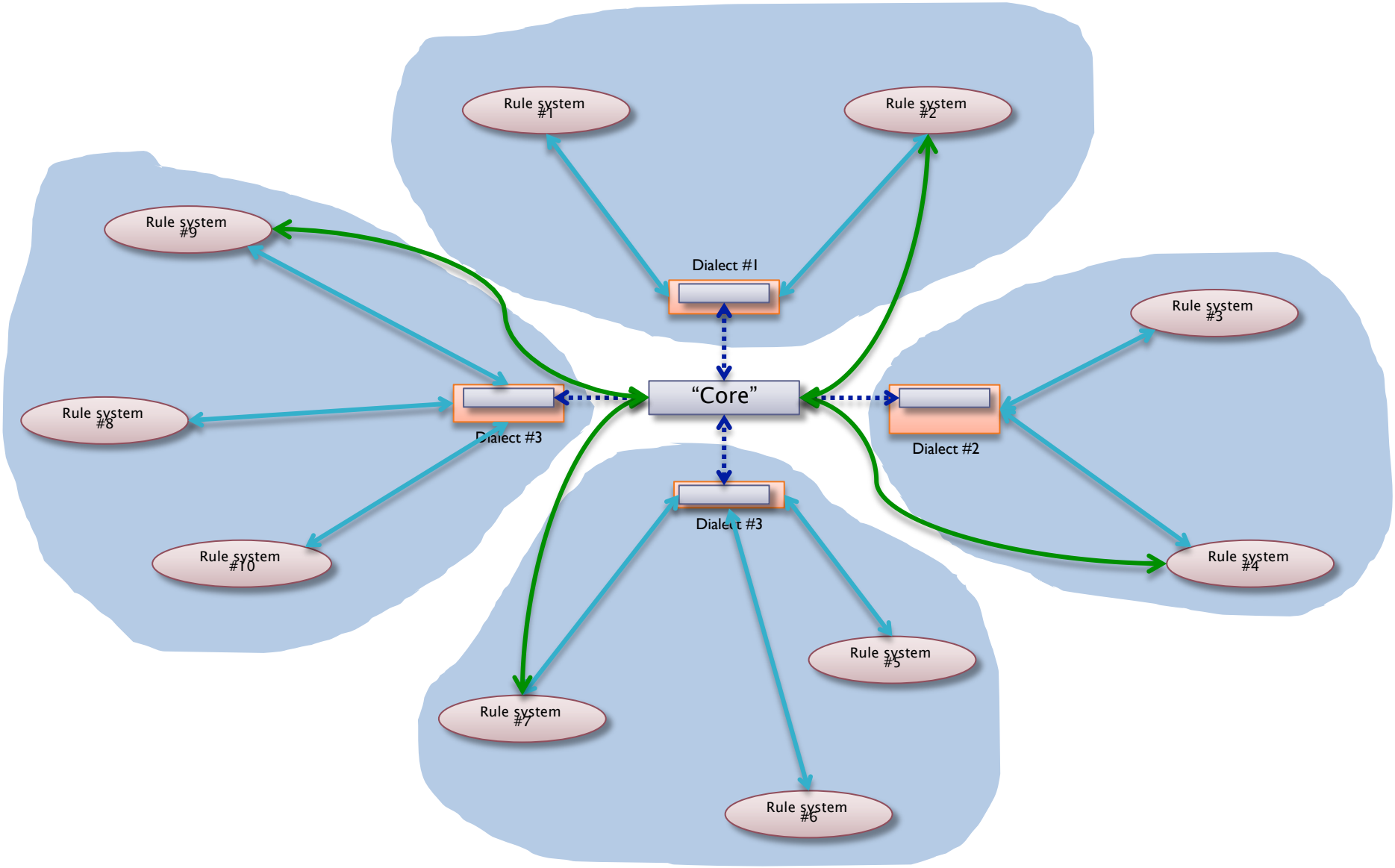
The role of dialects



The role of dialects



The role of dialects



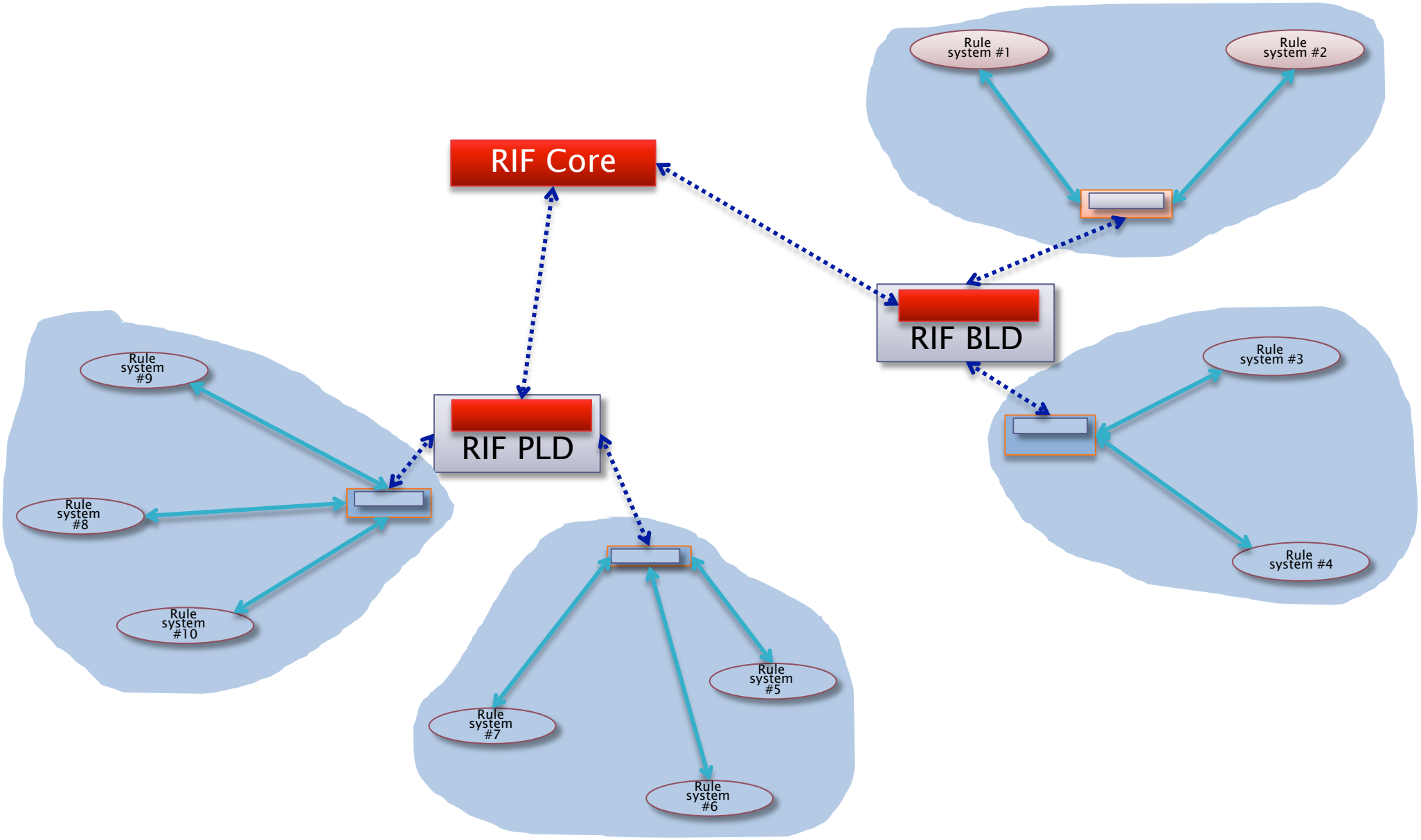
However…

- 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
  - the common RIF Core binding these two

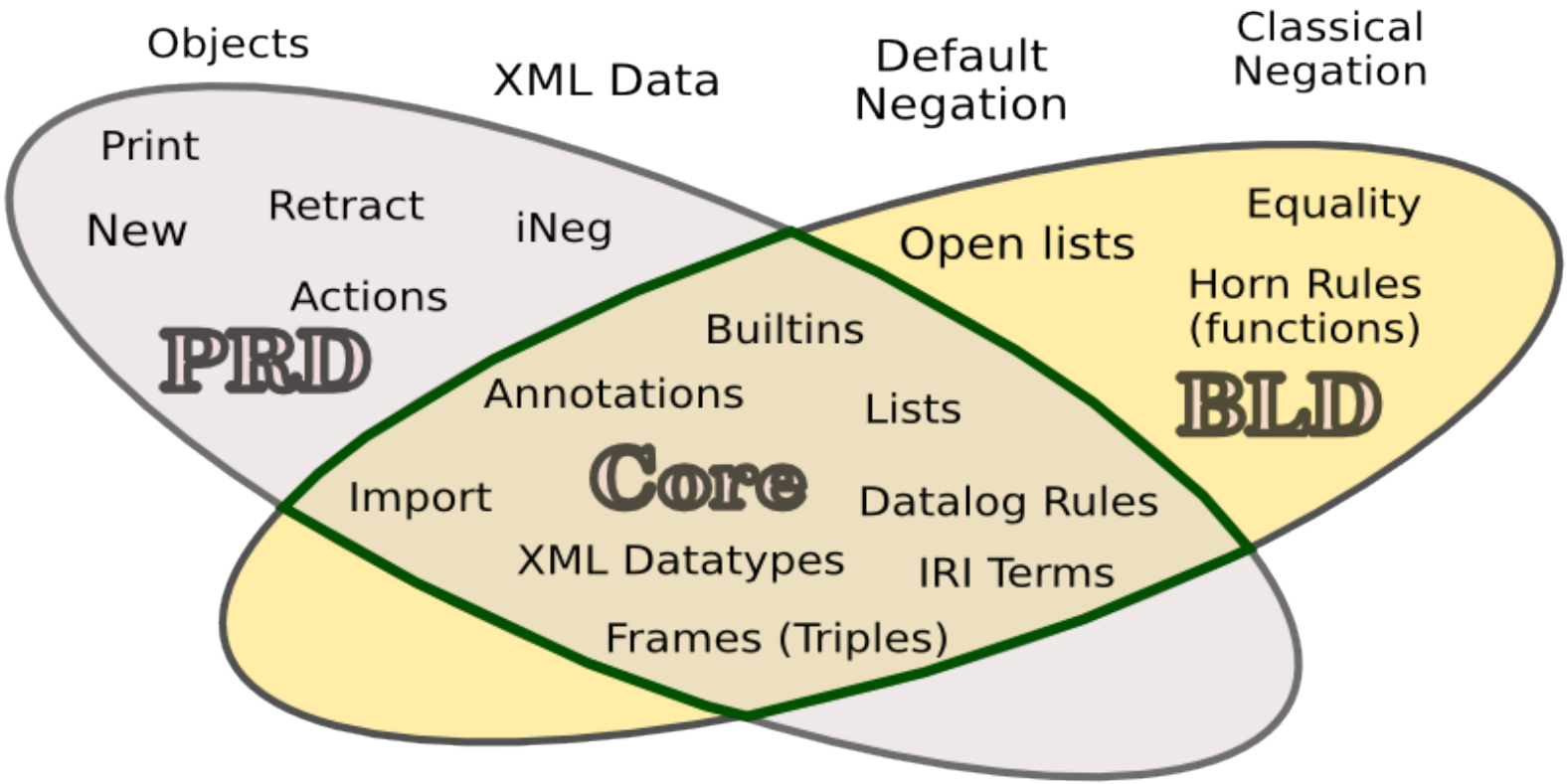
Schematically…

- The “BLD (Basic Logic Dialect)” is of the form:
  - “if condition true then this is true”
  - conditions may include functions, hierarchies
- The “PLD (Production Logic Dialect)” is of the form:
  - “if condition is true then do something”
- The “Core”: shared subset of major language

Hierarchy of cores

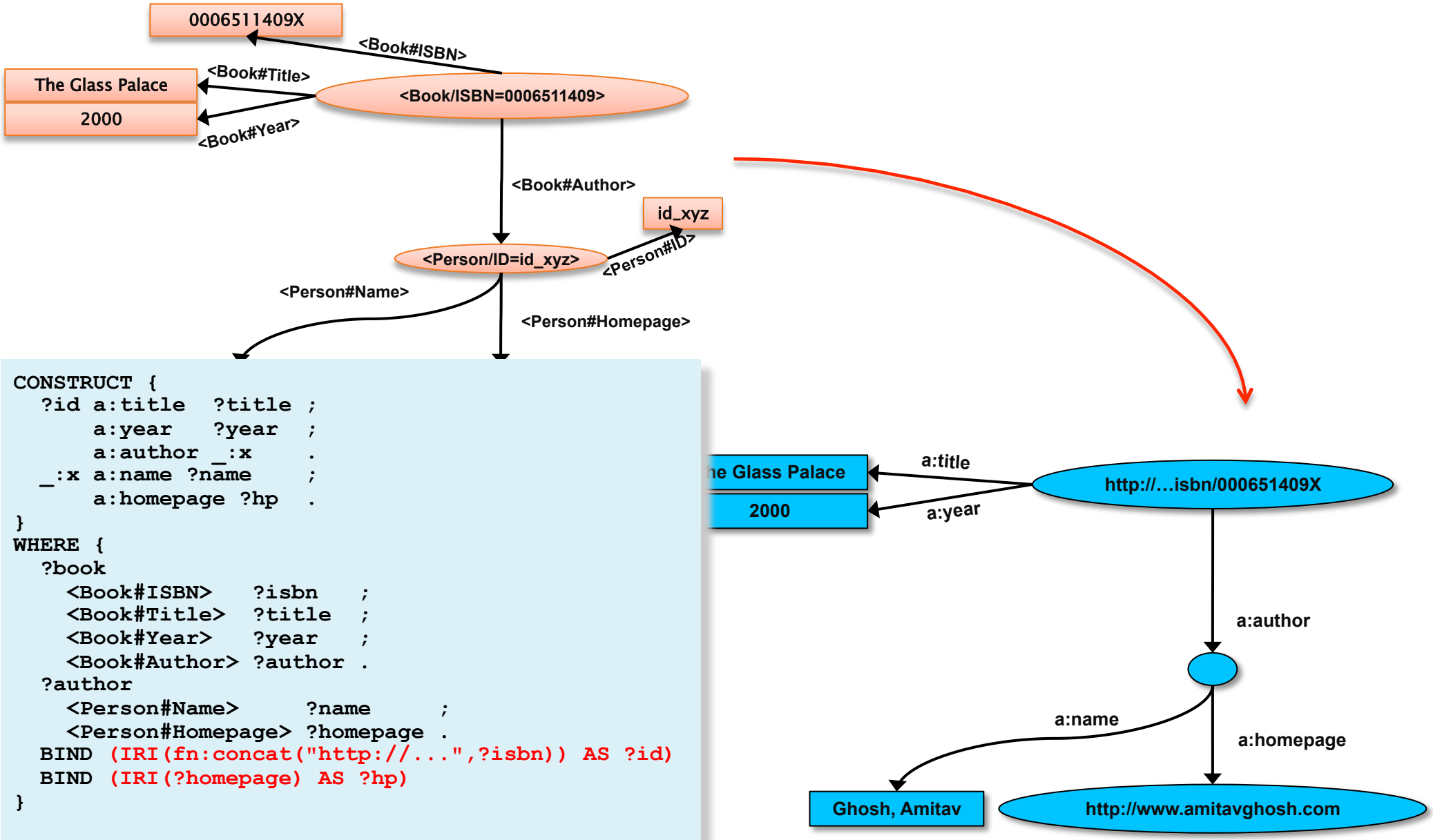


A rough division of features...

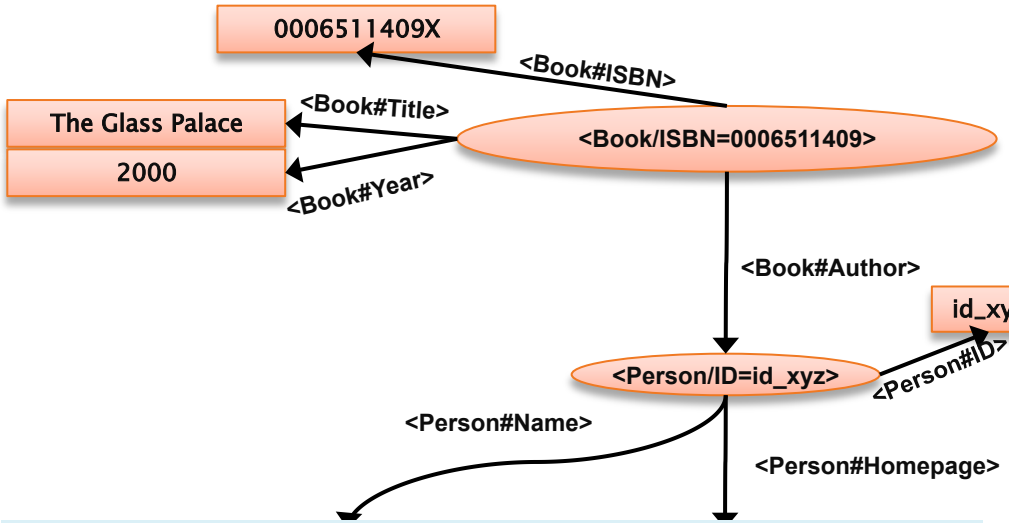


Courtesy of Sandro Hawke, W3C

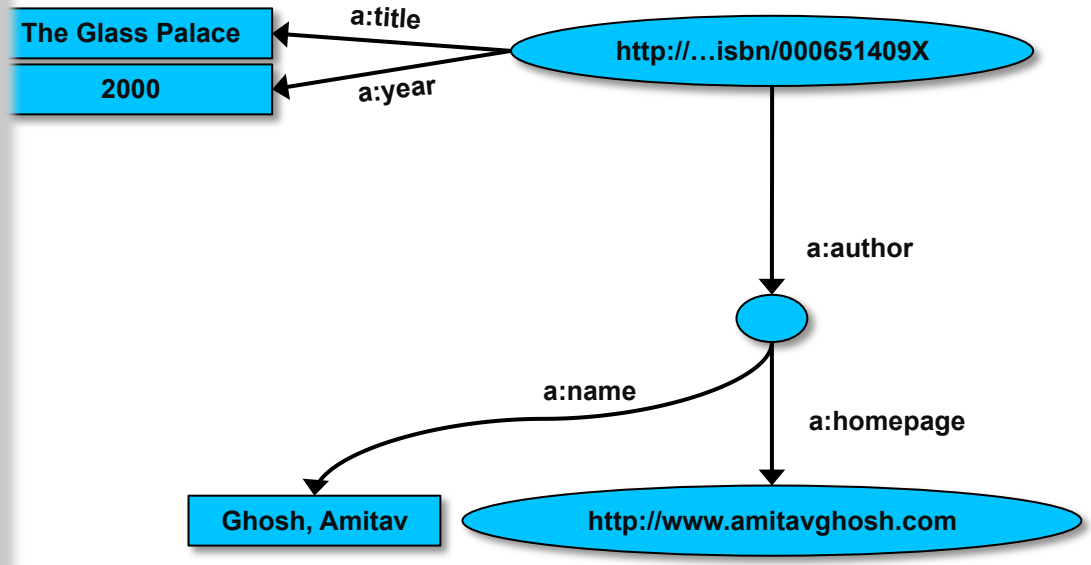
Remember this?



Same with RIF Core



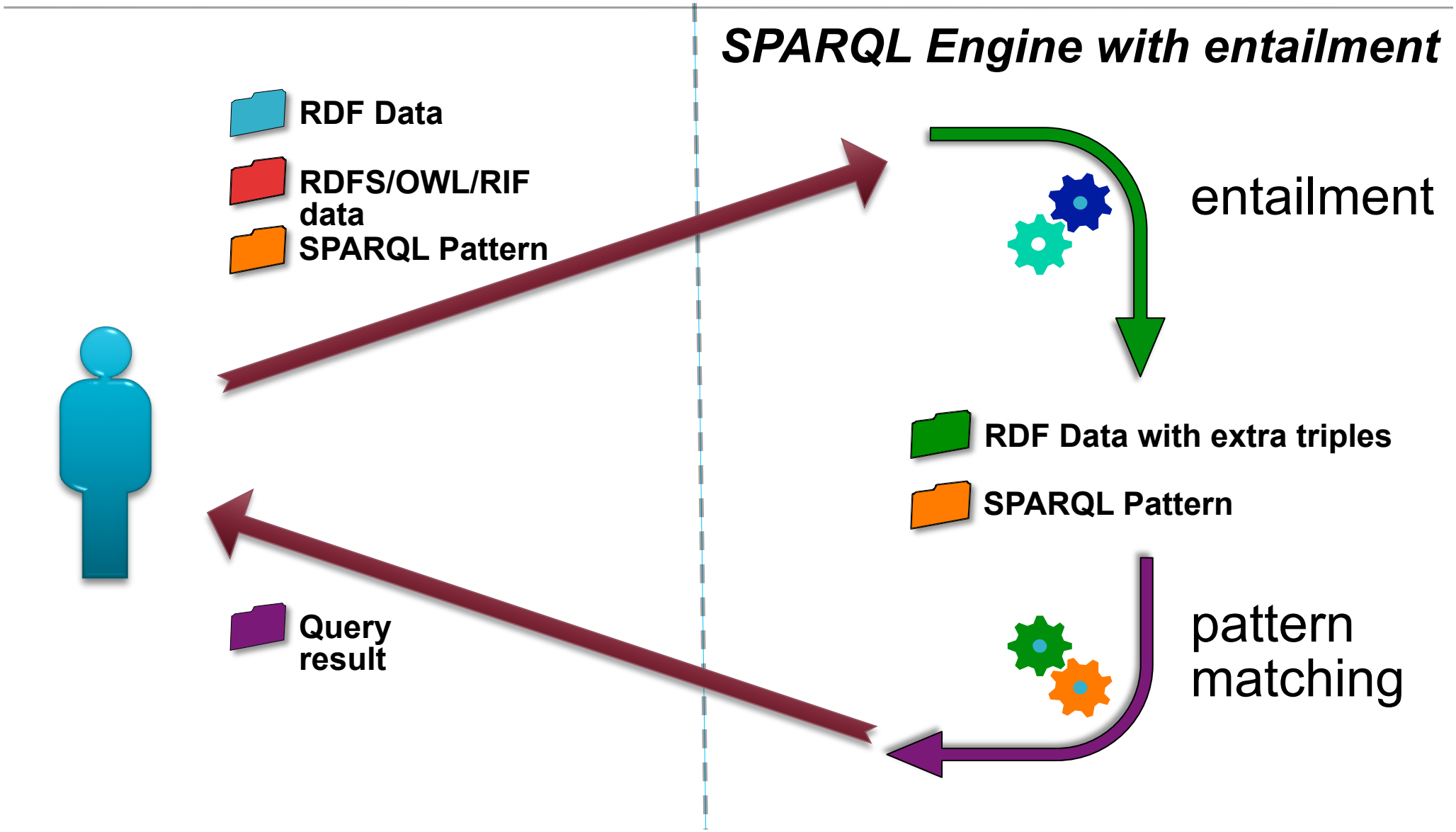

```
?id[a:year->?year a:title->?title
    a:author->?author] :-
  And(
    ?book[<Book#ISBN>   -> ?isbn
          <Book#Title>  -> ?title
          <Book#Year>   -> ?year
          <Book#Author> -> ?author]
    External(pred:iri-string(?id
      External(func:concat("http://..." ?isbn))))
    )
)
?author[a:name->?name a:homepage->?hp] :-
  And(
    ?author[<Person#Name>     -> ?name
            <Person#Homepage> -> ?homepage]
    External(pred:iri-string(?hp ?homepage))
  )
)
```



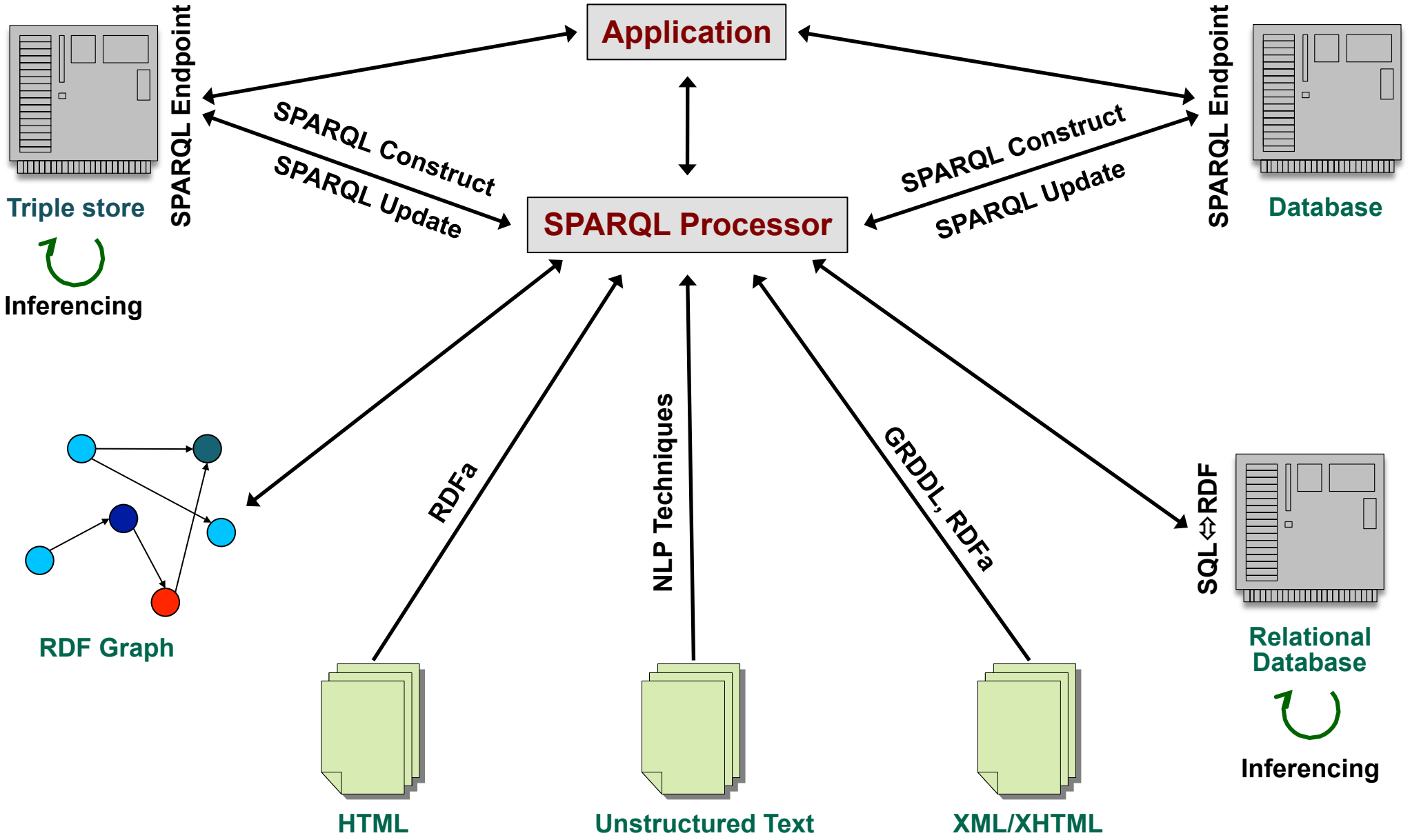
Vocabularies and SPARQL

- Question: how does SPARQL queries and vocabularies work together?
  - RDFS, OWL, and RIF produce new relationships
  - on what data do we query?
- Answer: in current SPARQL, that is not defined ☹
- But, in SPARQL 1.1 it is…☺

SPARQL 1.1 and RDFS/OWL/RIF

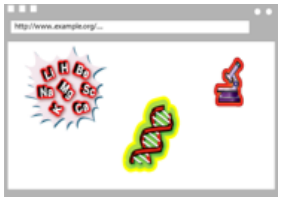


SPARQL 1.1 as a unifying point



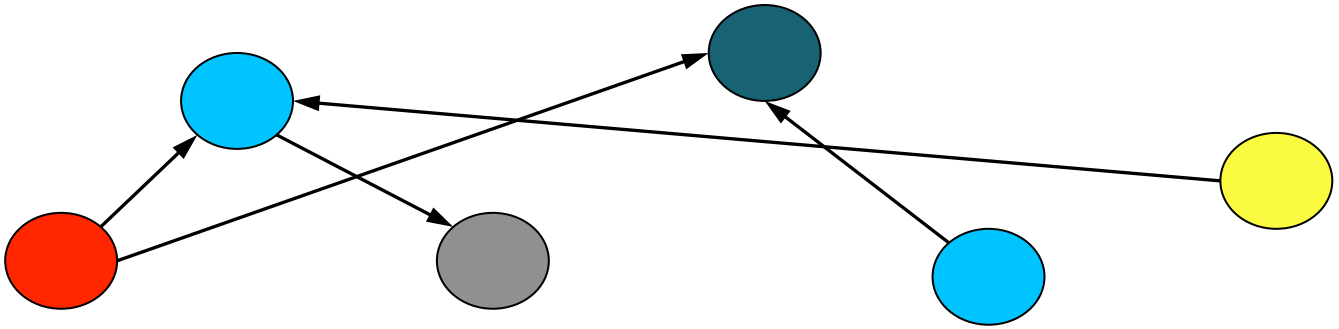

What have we achieved?
(putting all together)

Remember the integration example?



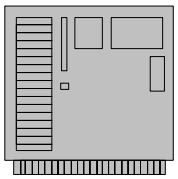
Applications

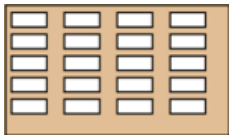
Manipulate
Query
…

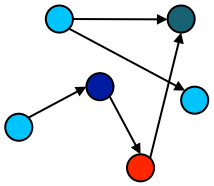


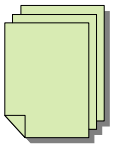
Data represented in abstract format

Map,
Expose,
…



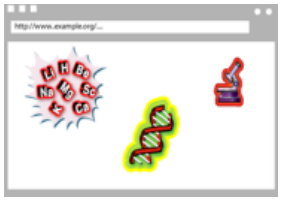






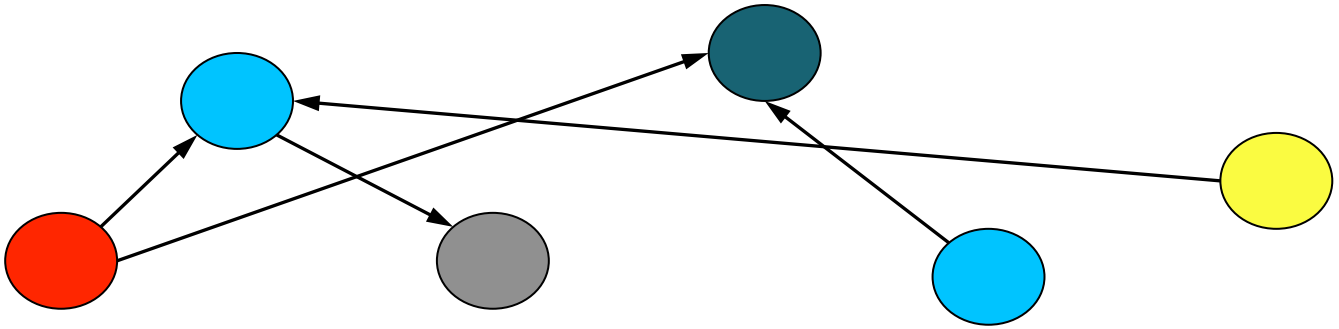
Data in various formats

Same with what we learned



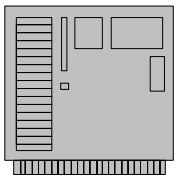
Applications

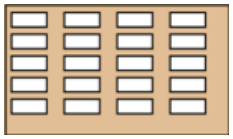
RESTful API
SPARQL,
Inferences
…

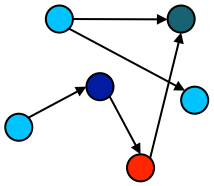


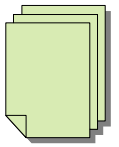
Data represented in RDF with extra knowledge (RDFS, SKOS, RIF, OWL,…)

R2RML,
RDFa,
…



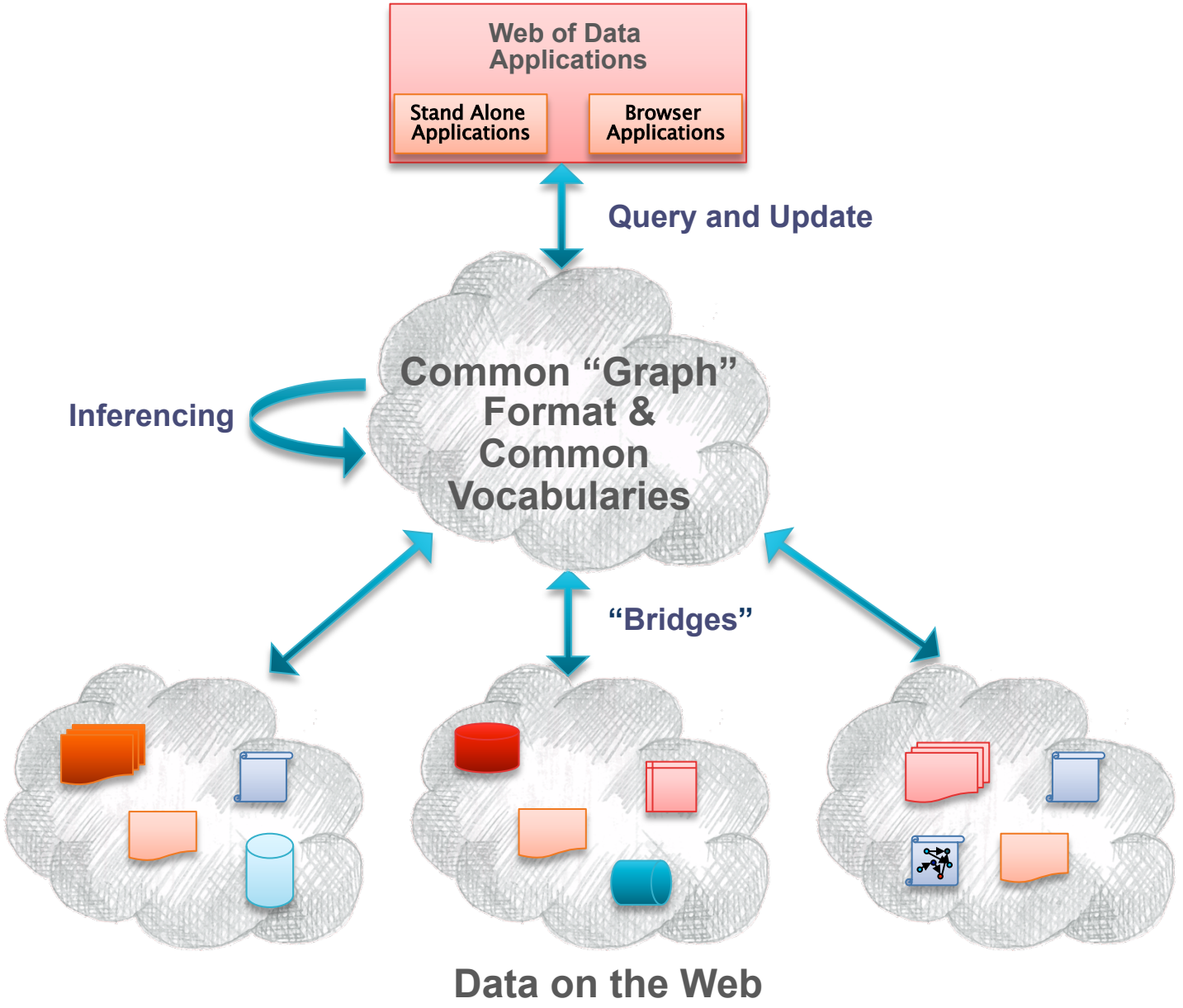




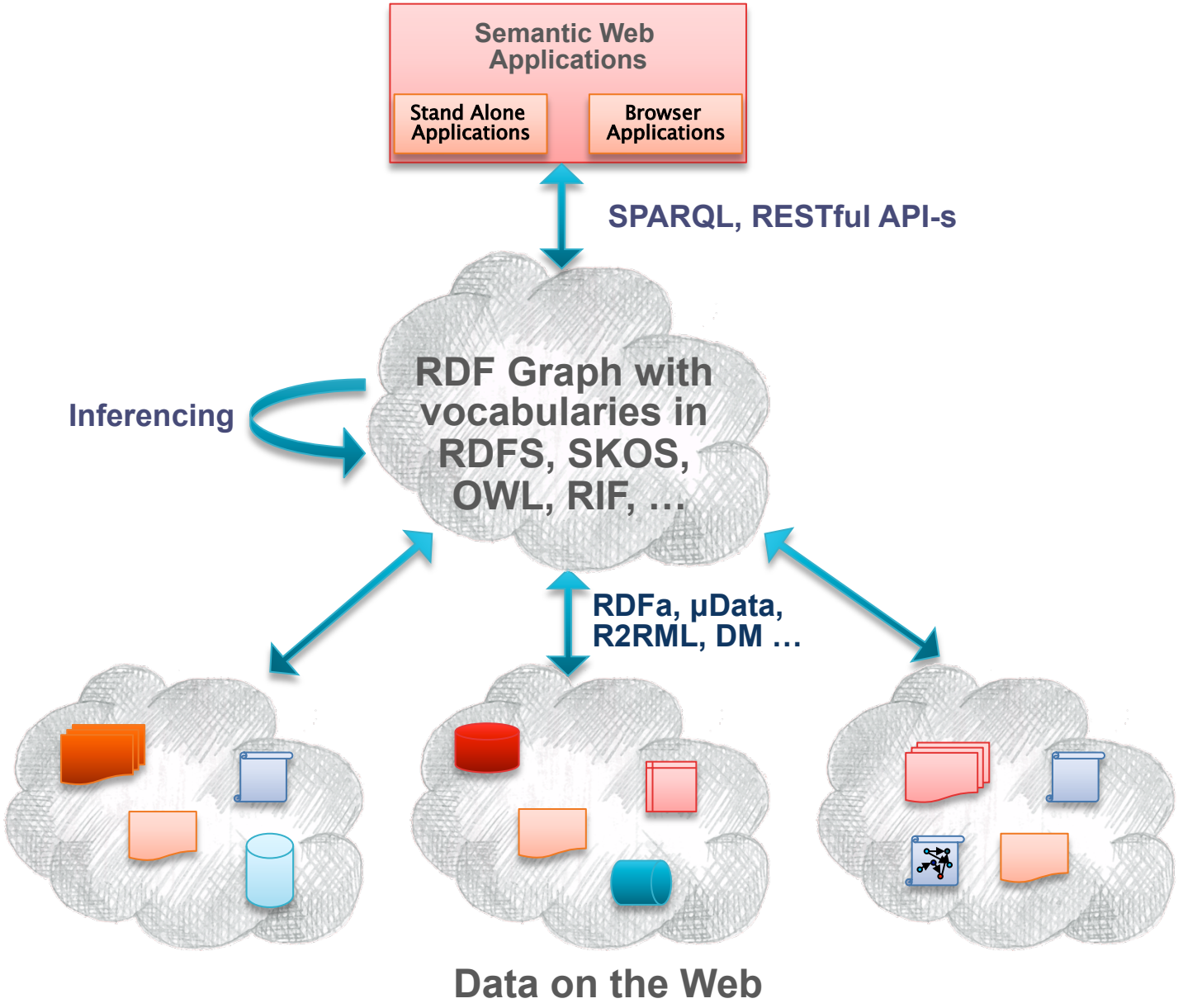


Data in various formats

Remember the integration example?



The same with what we learned




What else may be on the horizon?

Some items, issues, questions, …

- Quality constraints on graphs
  - “may I be sure that certain patterns are present in a graph?”
- Quality control of the content of the graphs
  - are links really what they should be?
- User interfaces for exploring large datasets
- Access control issues on a Web of Data

Some items, issues, questions, …

- Ontology merging, alignment, term equivalences, versioning, development, …
- Scaling:
  - what does it mean to “reason” and “infer” on graphs with millions of triples?
  - how can one systematically “explore” a huge Web of Data

Some items, issues, questions, …

- Naming:
  - the SW infrastructure relies on unique naming of “things” via URI-s
  - lots of discussions are happening that touch upon general Web architecture, too
    - HTTP URI-s or other URN-s?
    - URI-s for “informational resources” and “non informational resources”
    - how to ensure that URI-s used on the SW are dereferencable
    - etc.

Some items, issues, questions, …

- A different aspect of naming: what is the URI for a specific entity (regardless of the technical details)
  - what is the unique URI for, e.g., Bach's Well-Tempered Clavier?
    - obviously important for music ontologies an data
  - who has the authority or the means to define and maintain such URI-s?
  - should we define characterizing properties for these and use owl:sameAs instead of a URI?
- The traditional library community may be of a big help in this area

Some items, issues, questions, …

- Uncertainty:
  - Fuzzy logic
    - look at alternatives of Description Logic based on fuzzy logic
    - alternatively, extend RDF(S) with fuzzy notions
  - Probabilistic statements
    - have an OWL class membership with a specific probability
    - combine reasoners with Bayesian networks
  - A W3C Incubator Group has issued a report on the current status, possibilities, directions, etc.
  - possible RIF dialect for fuzzy logic, for example?

Available documents, resources

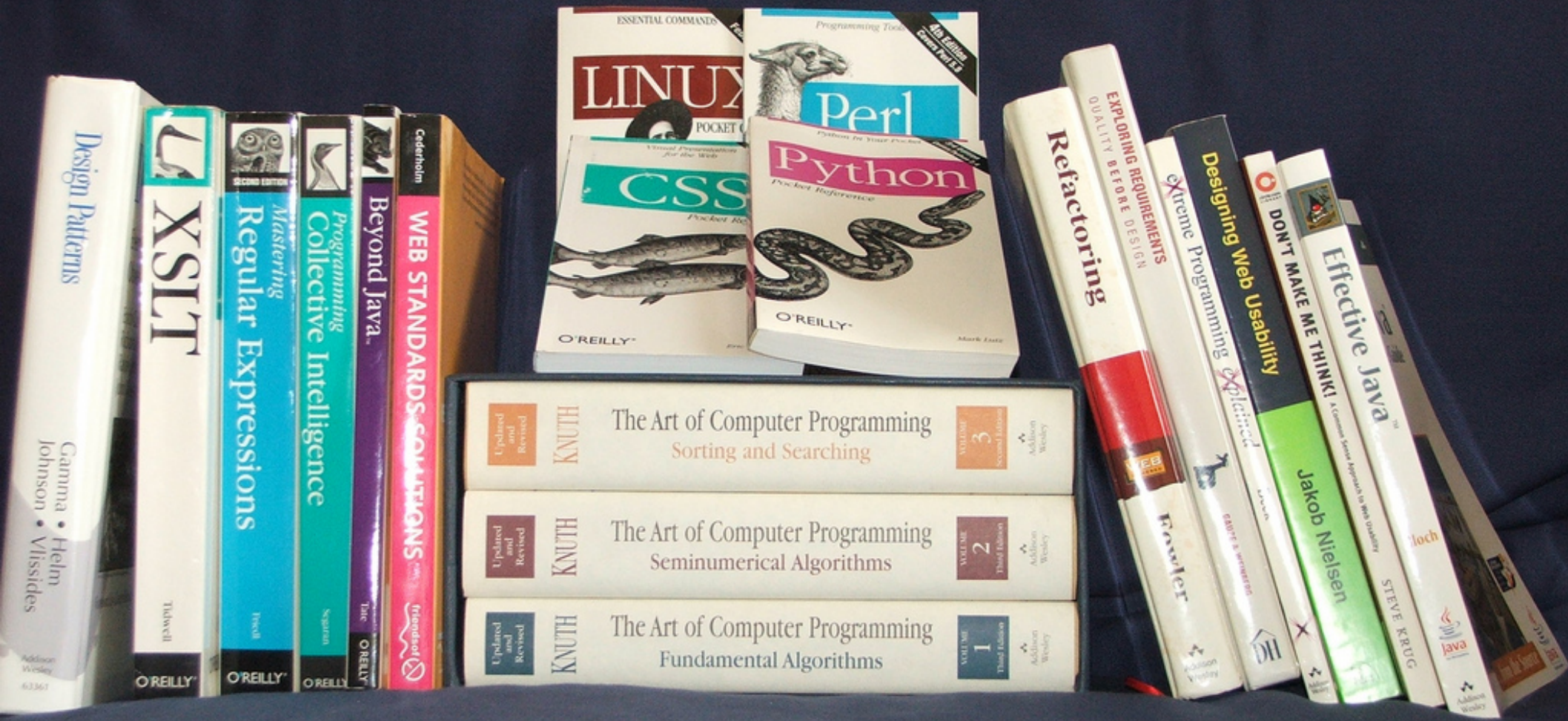


Photo credit "Abizern", Flickr

Available specifications: Primers, Guides

- The “RDF Primer” and the “OWL Guide” give a formal introduction to RDF(S) and OWL
- SKOS has its separate “SKOS Primer”
- GRDDL Primer and RDFa Primer have been published; RIF Primer is on its way
- The W3C Semantic Web Activity Wiki has links to all the specifications

“Core” vocabularies

- There are also a number “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

- T. Heath and C. Bizer: *Linked Data: Evolving the Web into a Global Data Space*, 2011
- M. Watson: *Practical Semantic Web and Linked data Applications*, 2010
- P. Hitzler, R. Sebastian, and M. Krötzsch: *Foundation of Semantic Web Technologies*, 2009
- G. Antoniu and F. van Harmelen: *Semantic Web Primer, 2nd edition,* 2008
- D. Allemang and J. Hendler: *Semantic Web for the Working Ontologist*, 2008
- …

See the separate Wiki page collecting book references

Further information

- Planet RDF aggregates a number of SW blogs:
  - http://planetrdf.com/
- Semantic Web Interest Group
  - a forum developers with a publicly archived 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/
- Linked Data mailing list
  - a forum concentrating on linked data with a public archive
  - anybody can sign up on the list
    - http://lists.w3.org/Archives/Public/public-lod/

Great community…

Some good lessons

computas

- New standards (e.g. SPARQL), proposals for standardization (e.g. SPARUL), new tools (e.g. Jena), open source (e.g. Tomcat, Apache), lack of good documentation all say **high risk!!!!**
- **However, the support and maintenance from the W3C community and open source developers (e.g. Jena team) has been impressive, the support through IRC channels, mailing lists etc has been invaluable for the project.**

From a presentation given by David Norheim, Computas AS, ESTC2008 Conference, Vienna, Austria

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

- Some names:
  - Jena, AllegroGraph, Mulgara, Sesame, flickurl, 4Store, Stardog, …
  - TopBraid Suite, Virtuoso environment, Falcon, Drupal 7, Redland, Pellet, …
  - Disco, Oracle 11g, RacerPro, IODT, Ontobroker, OWLIM, Talis Platform, …
  - RDF Gateway, RDFLib, Open Anzo, DartGrid, Zitgist, Ontotext, Protégé, …
  - Thetus publisher, SemanticWorks, SWI-Prolog, RDFStore…
  - …
- Categories:
  - Triple Stores
  - Inference engines
  - Converters
  - Search engines
  - Middleware
  - CMS
  - Semantic Web browsers
  - Development environments
  - Semantic Wikis
  - …

More on http://www.w3.org/2001/sw/wiki/Tools

Deployment, applications

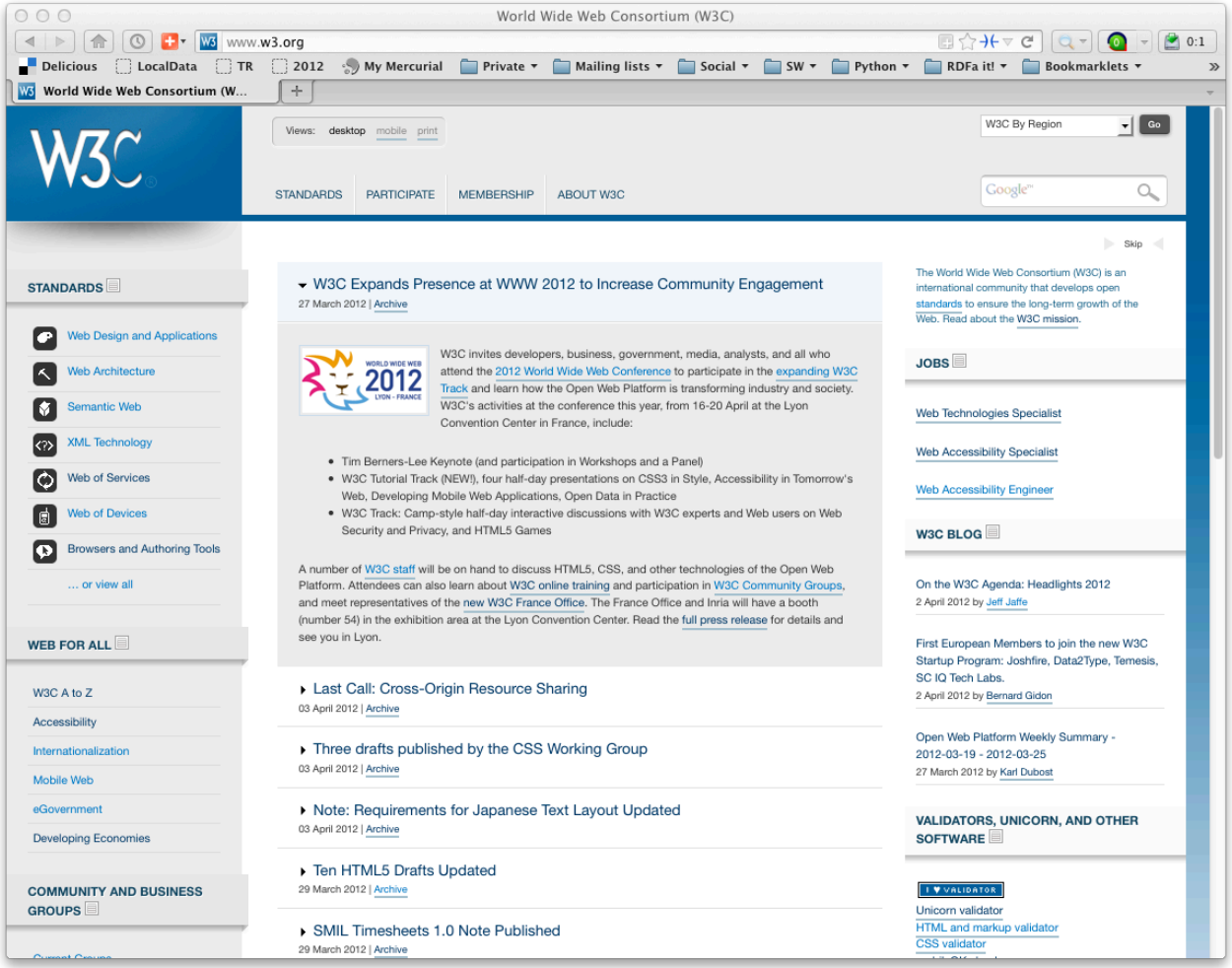
See the separate slide set…

Conclusions

To remember…

- *Data* on the Web is a major challenge
  - technologies are needed to use them, to interact with them, to integrate them
- Semantic Web technologies, Linked Data principles and practices, etc., should play a major role in publishing and using Data on the Web

Lot remain to be done…



- Lots of issues to be solved
- But… W3C needs experts!
  - consider joining W3C, as well as the work done there!

Thank you for your attention

These slides are also available on the Web:

http://www.w3.org/People/Ivan/CorePresentations/SWTutorial/