Introduction to the Semantic Web (tutorial)

2009 Semantic Technology Conference
San Jose, California, USA
June 15, 2009

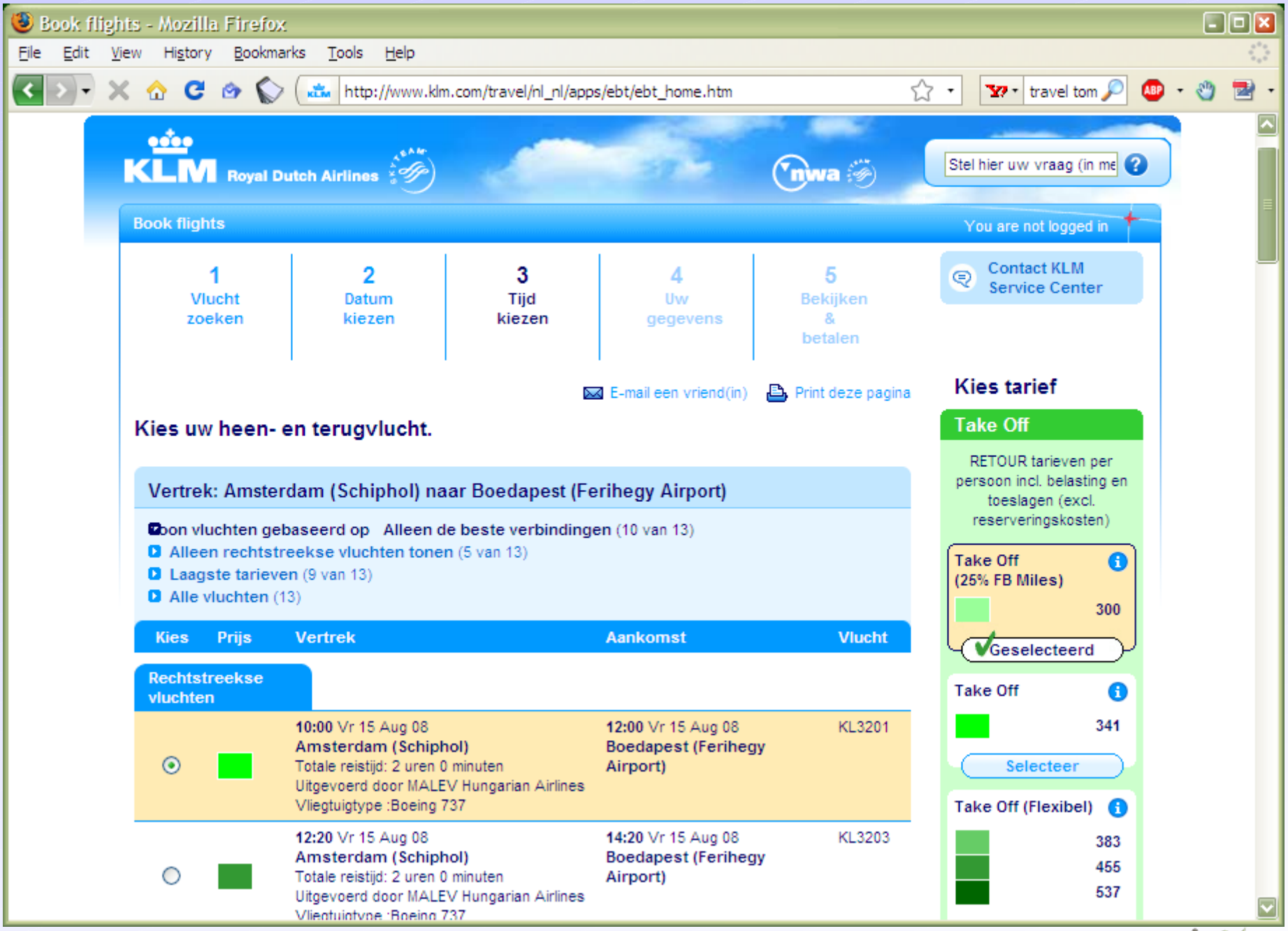
Ivan Herman, W3C
ivan@w3.org

Introduction

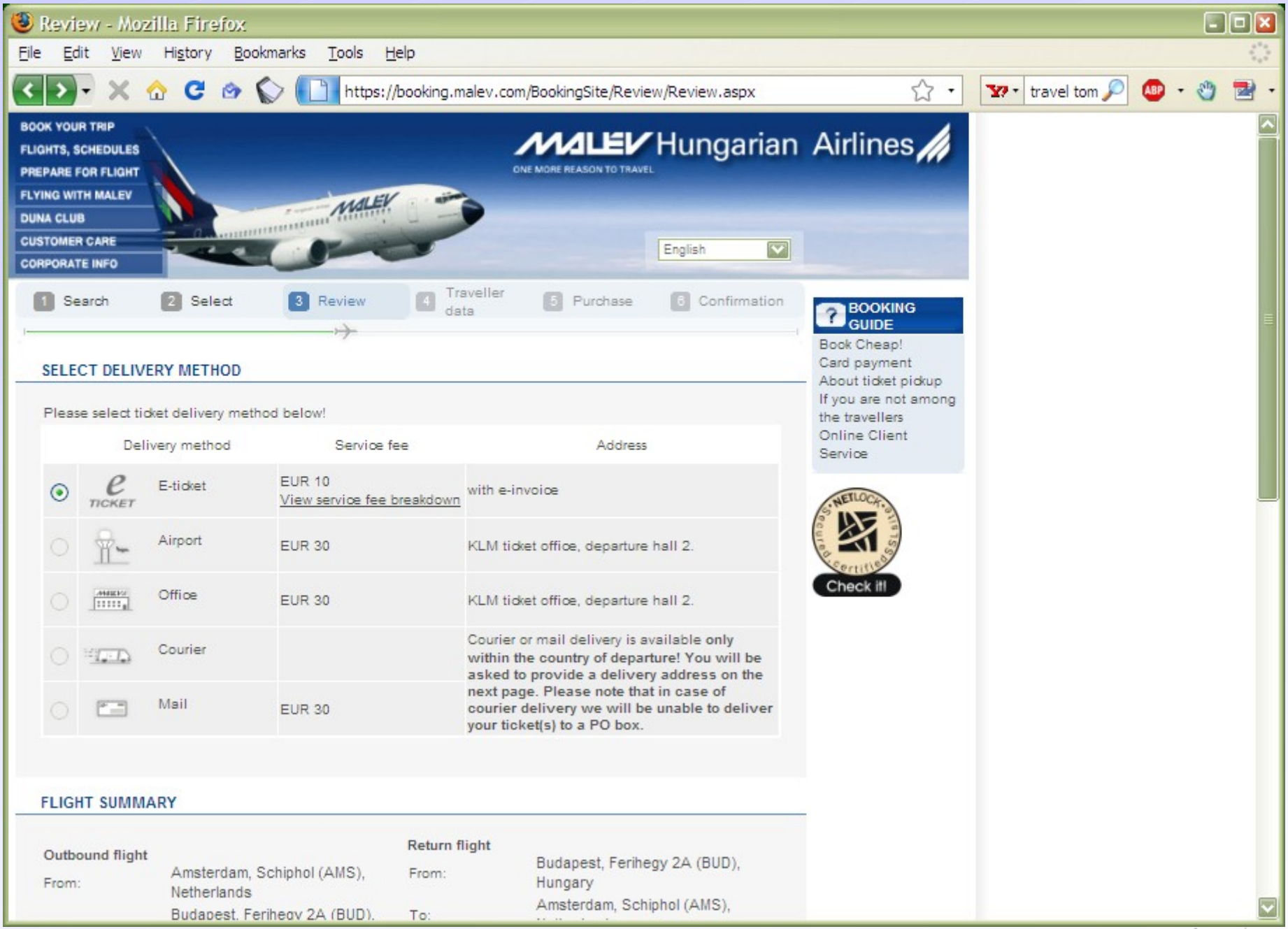
Let's organize a trip to Budapest using the Web!

You try to find a proper flight with …

… a big, reputable airline, or …

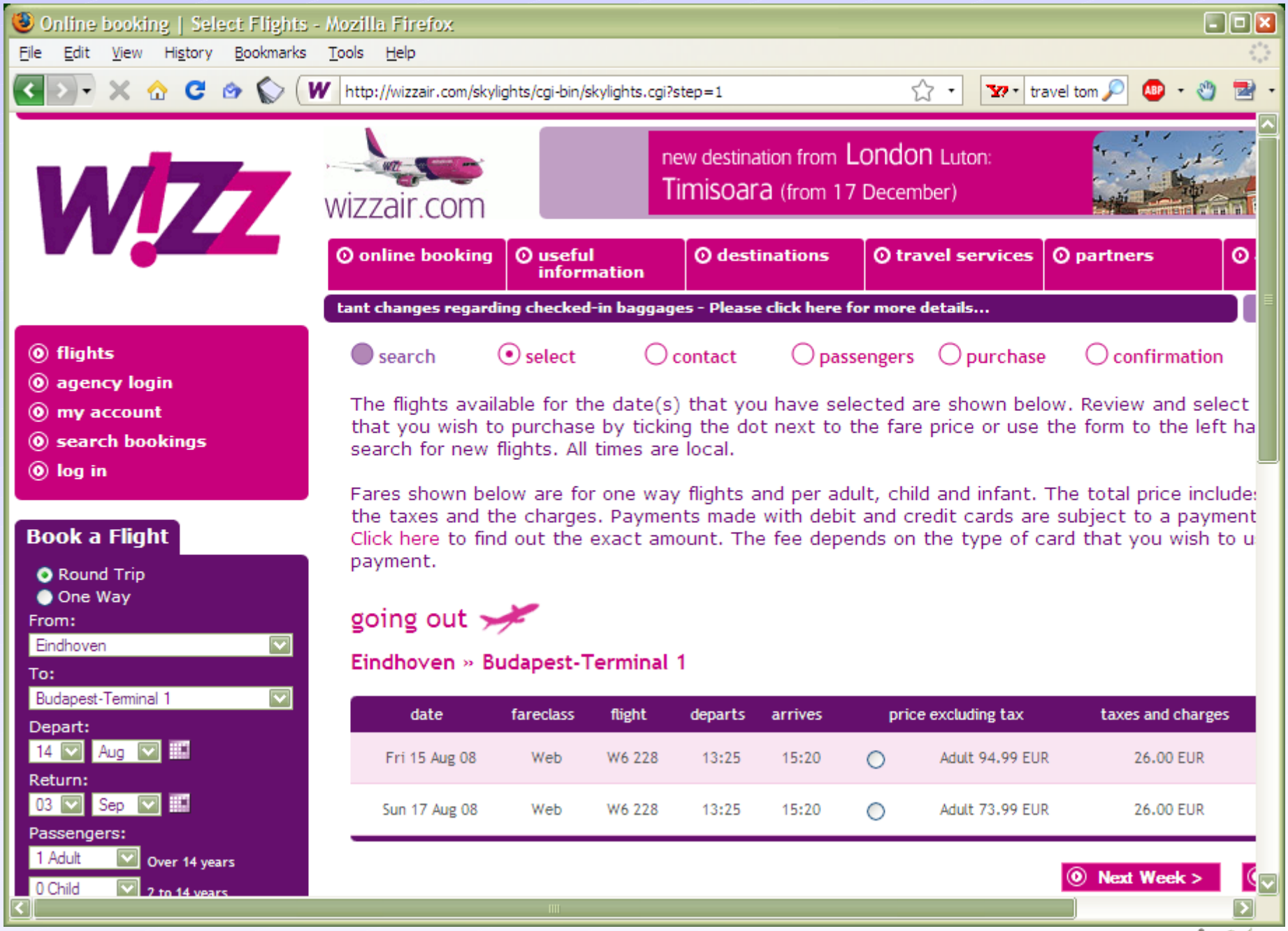


… the airline of the target country, or …



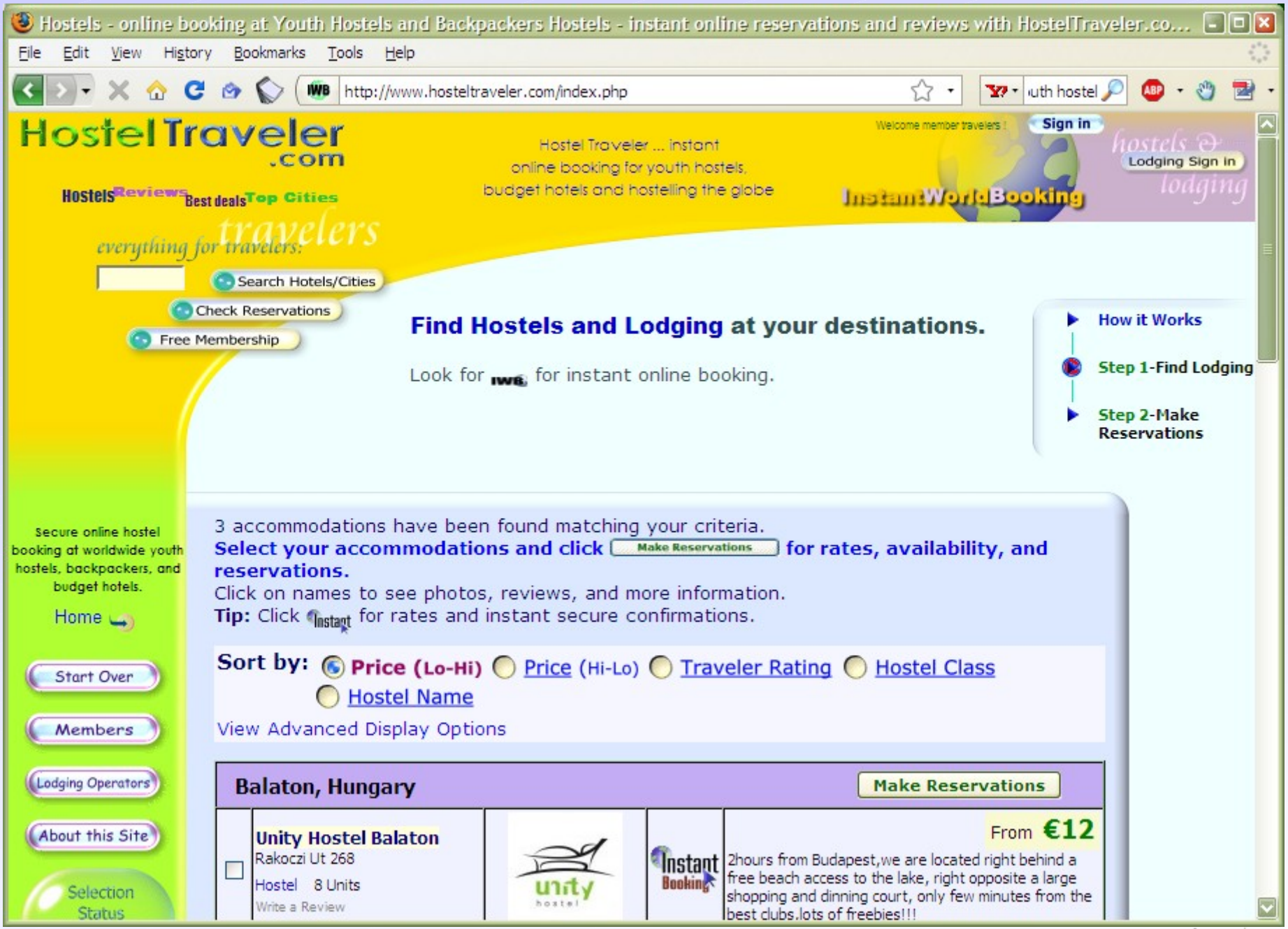
W3C Semantic Web

… or a low cost one



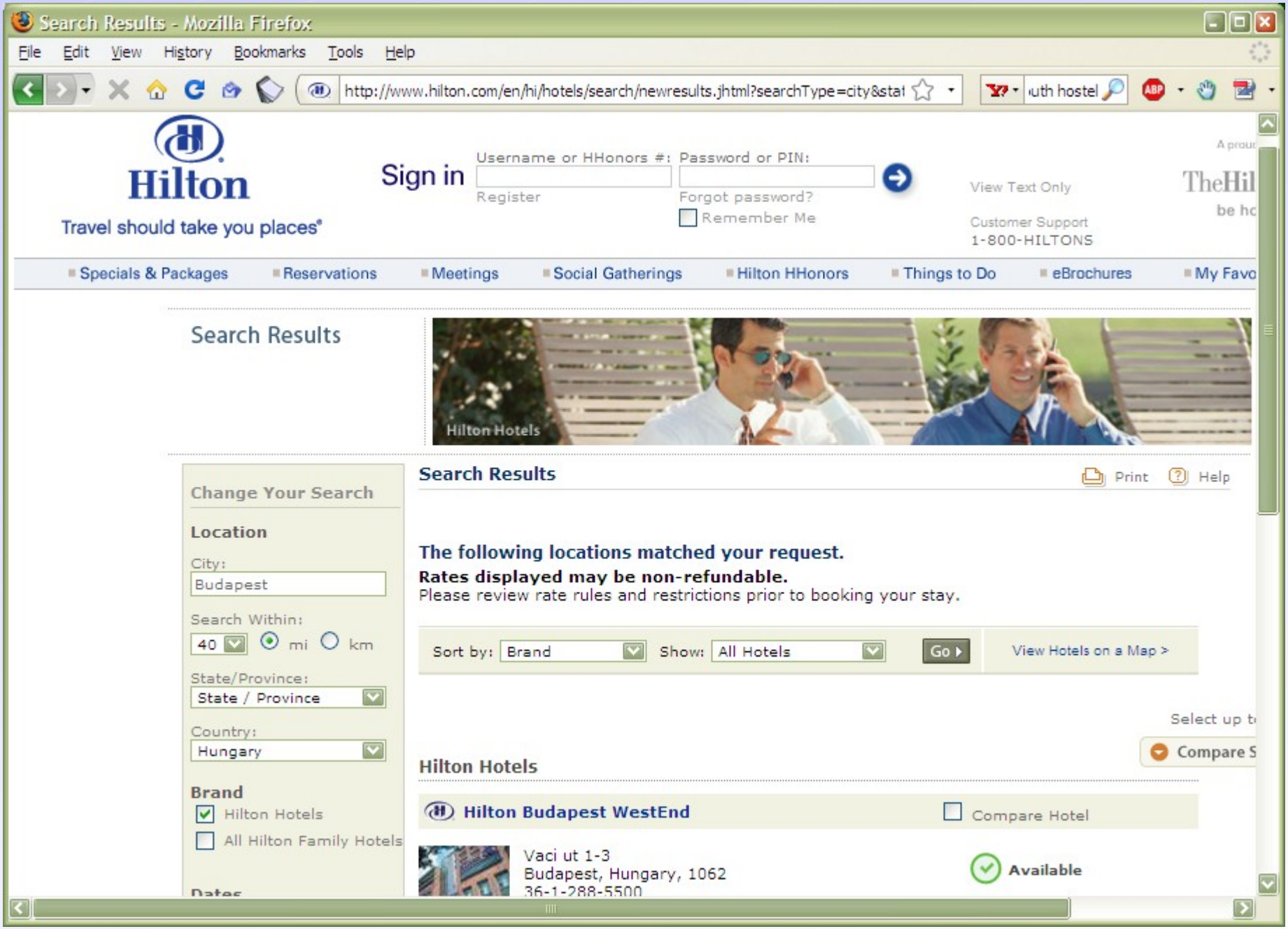
You have to find a hotel, so you look for…

… a really cheap accommodation, or …

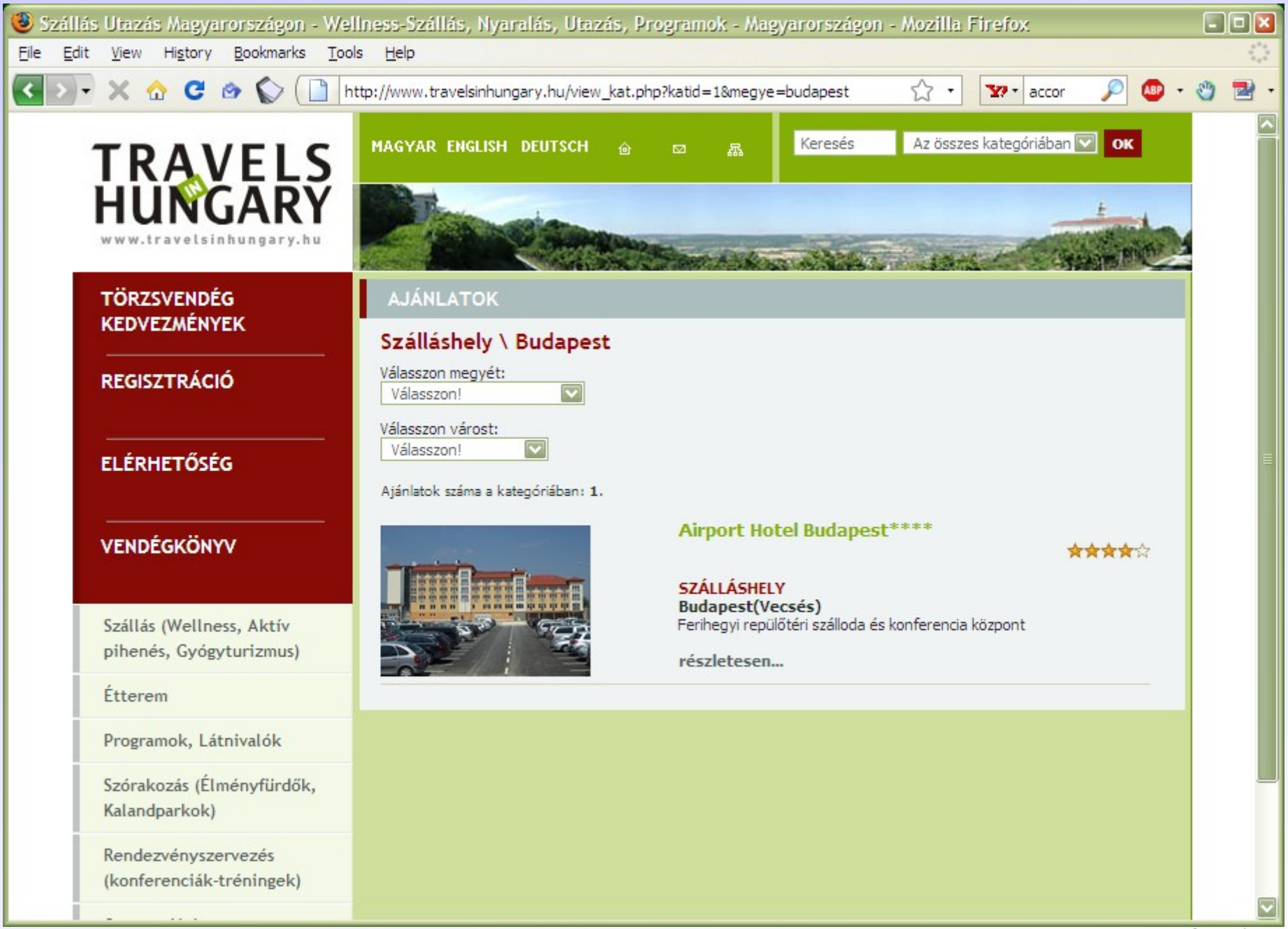


W3C Semantic Web

… or a really luxurious one, or …

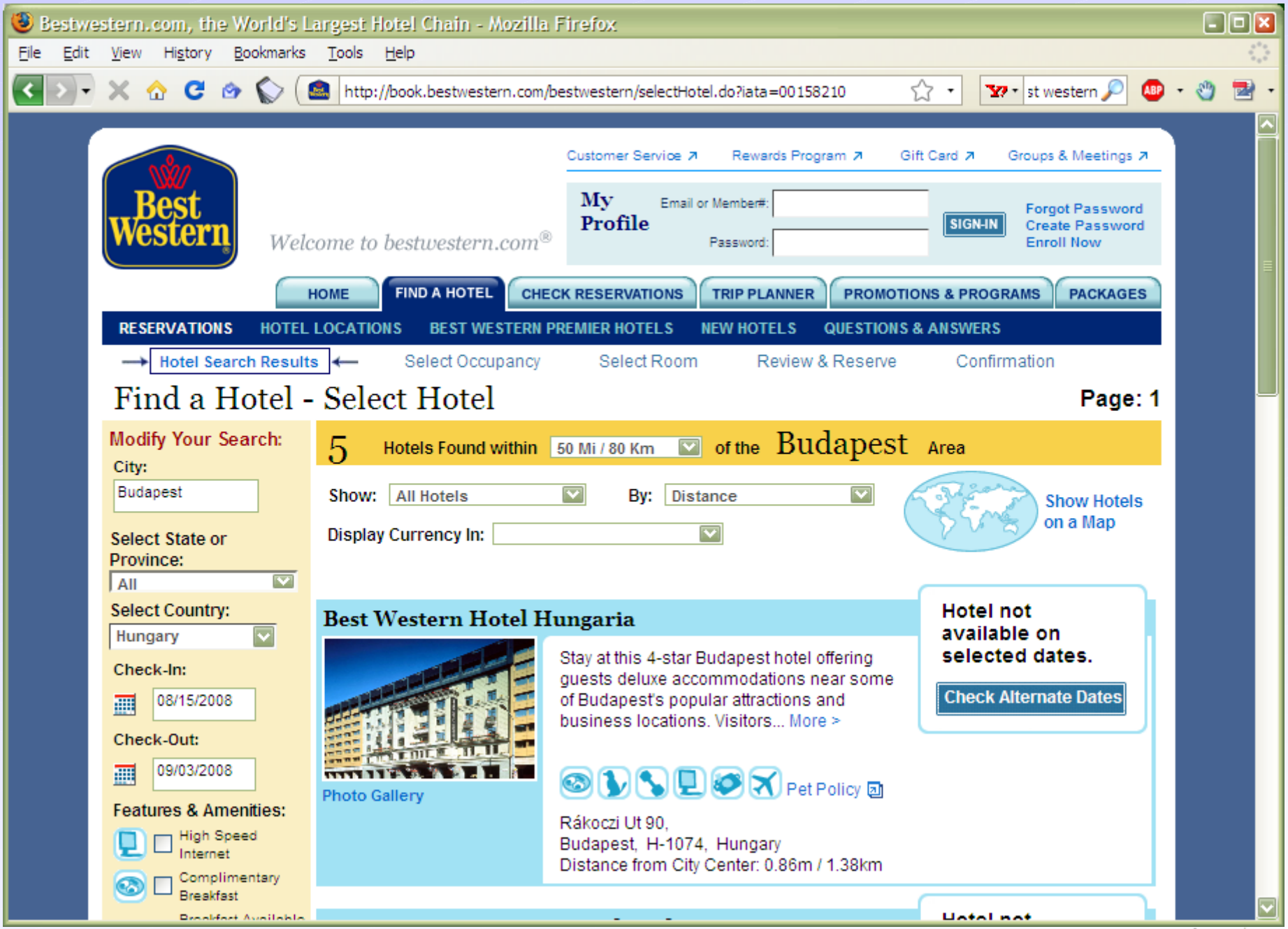


… an intermediate one …



oops, that is no good, the page is in Hungarian that almost nobody understands, but…

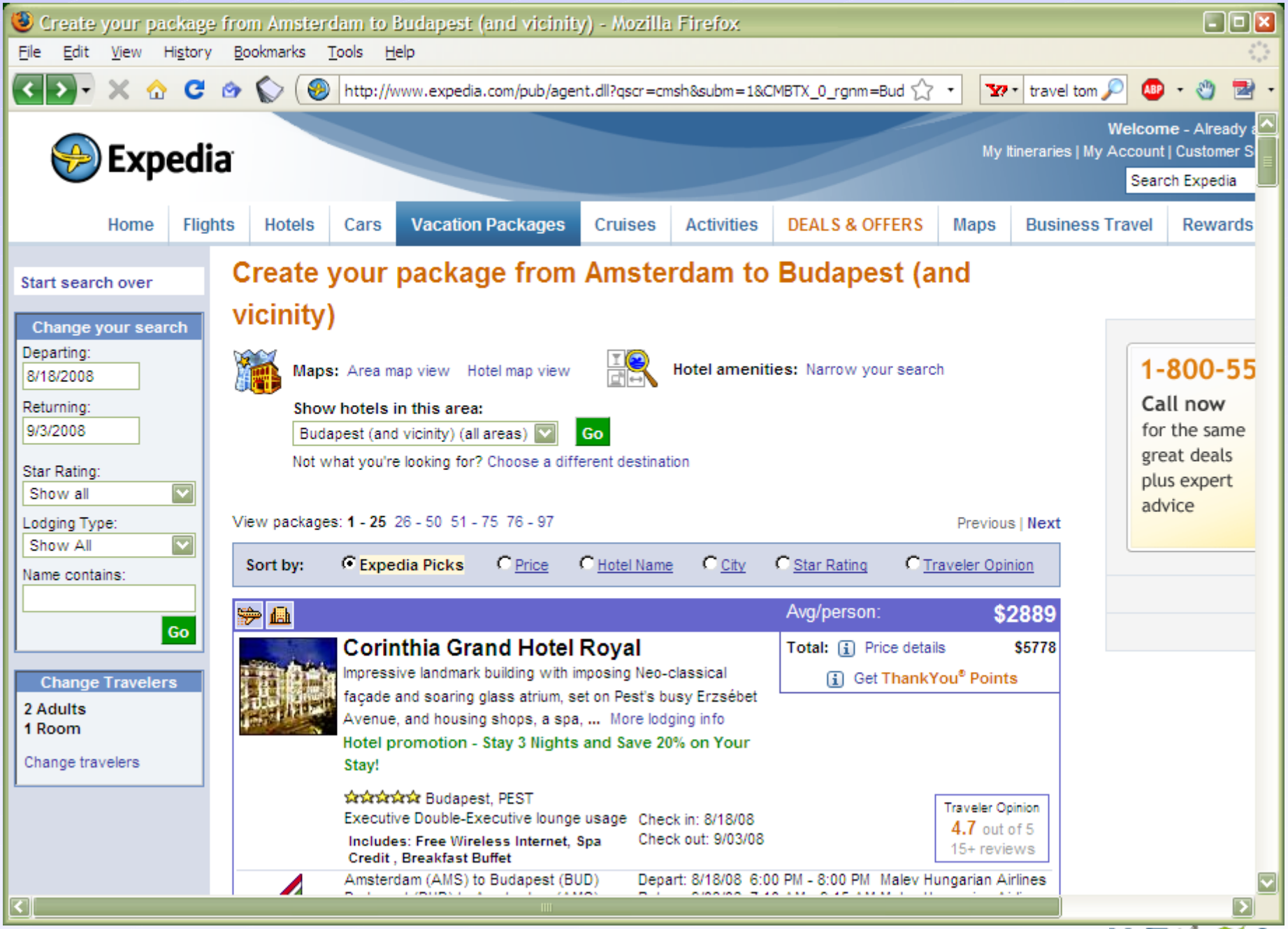
… this one could work



W3C Semantic Web

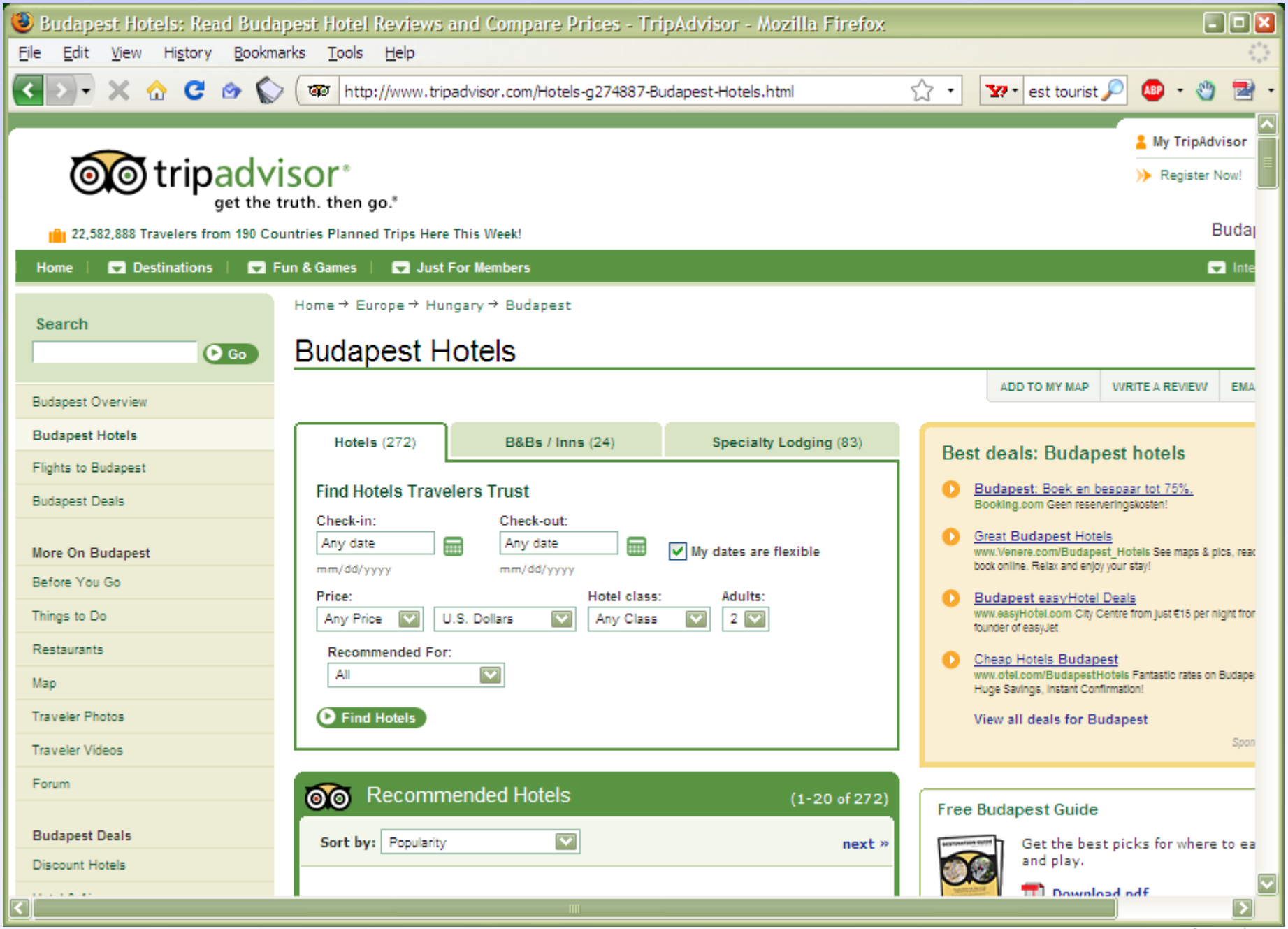
Of course, you could decide to trust a specialized site…

… like this one, or…



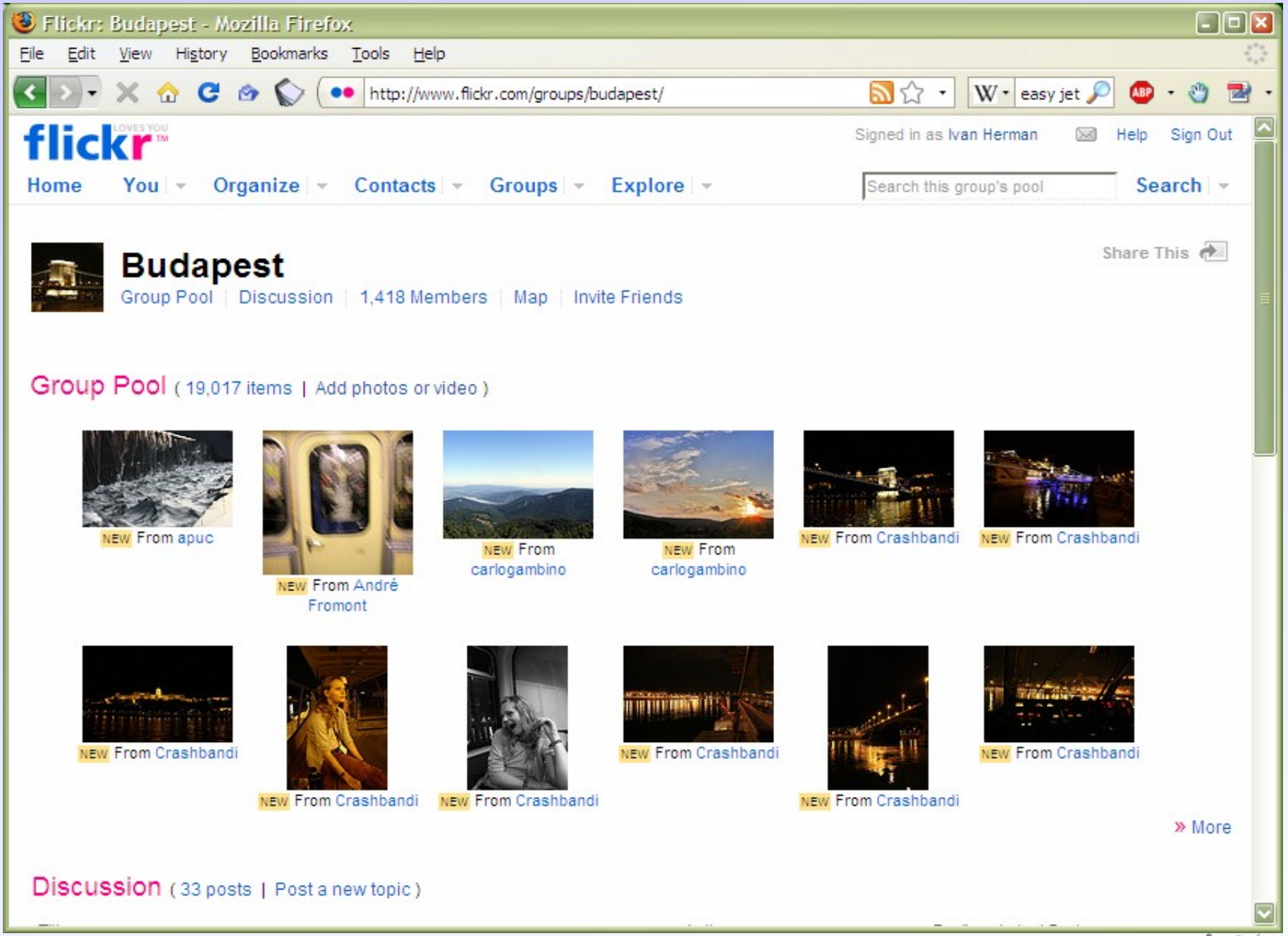
W3C Semantic Web

… or this one

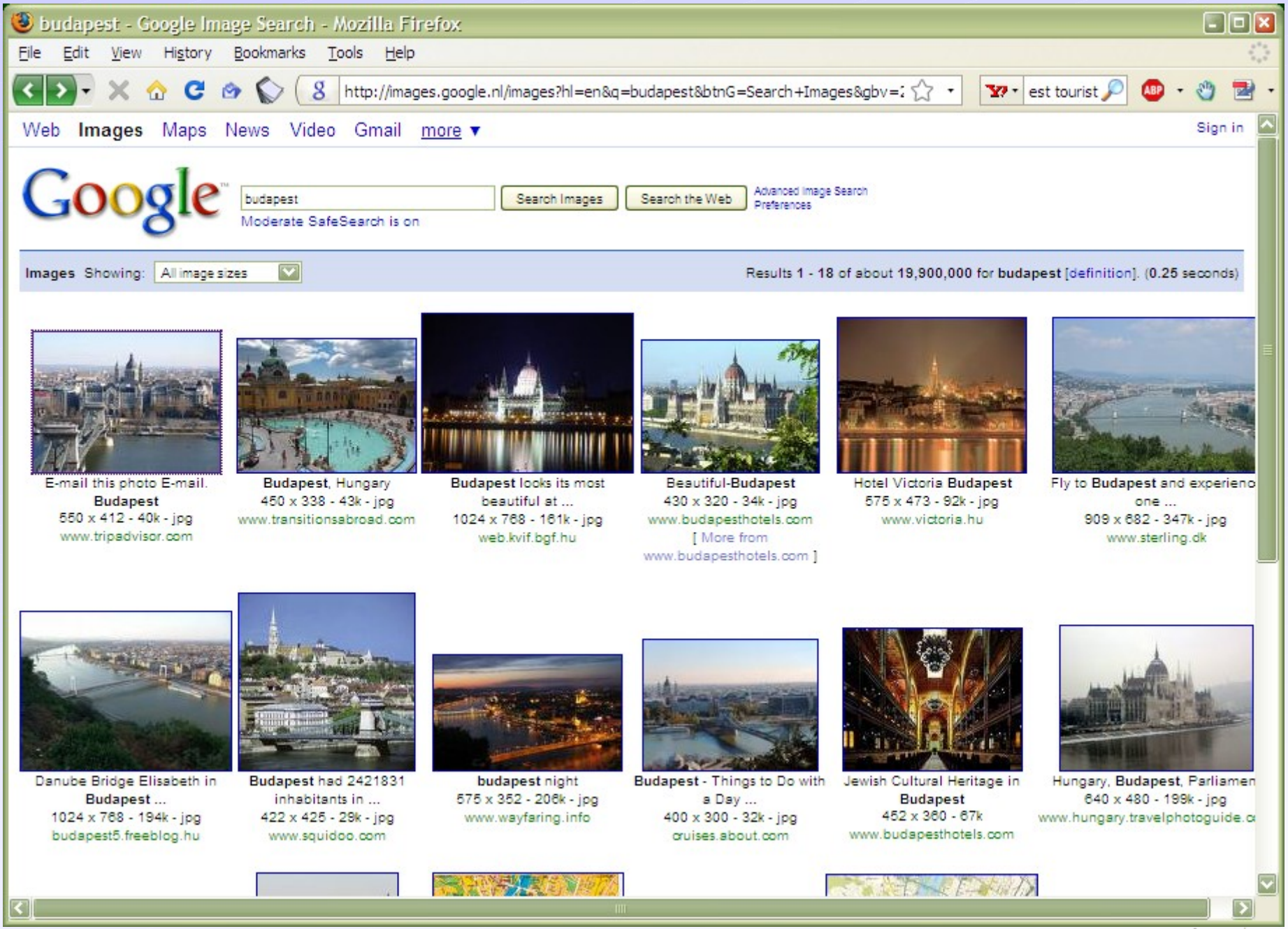

Budapest Hotels: Read Budapest Hotel Reviews and Compare Prices - TripAdvisor - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.tripadvisor.com/Hotels-g274887-Budapest-Hotels.html
tripadvisor
get the truth. then go.
My TripAdvisor
Register Now!
22,582,888 Travelers from 190 Countries Planned Trips Here This Week!
Home | Destinations | Fun & Games | Just For Members
Search
Go
Budapest Overview
Budapest Hotels
Flights to Budapest
Budapest Deals
More On Budapest
Before You Go
Things to Do
Restaurants
Map
Traveler Photos
Traveler Videos
Forum
Budapest Deals
Discount Hotels
Home → Europe → Hungary → Budapest
Budapest Hotels
ADD TO MY MAP
WRITE A REVIEW
Hotels (272)
B&Bs / Inns (24)
Specialty Lodging (83)
Find Hotels Travelers Trust
Check-in:
Any date
mm/dd/yyyy
Check-out:
Any date
mm/dd/yyyy
My dates are flexible
Price:
Any Price
U.S. Dollars
Hotel class:
Any Class
Adults:
2
Recommended For:
All
Find Hotels
Best deals: Budapest hotels
Budapest: Boek en bespaar tot 75%.
Booking.com Geen reserveringskosten!
Great Budapest Hotels
Budapest easyHotel Deals
www.easyHotel.com City Centre from just €15 per night fror
founder of easyJet
Cheap Hotels Budapest
www.otel.com/BudapestHotels Fantastic rates on Budape
Huge Savings. Instant Confirmation!
View all deals for Budapest
Recommended Hotels
(1-20 of 272)
Sort by: Popularity
next »
Free Budapest Guide
Get the best picks for where to ea
and play.
Download pdf

You may want to know something about Budapest; look for some photographs…

… on flickr …

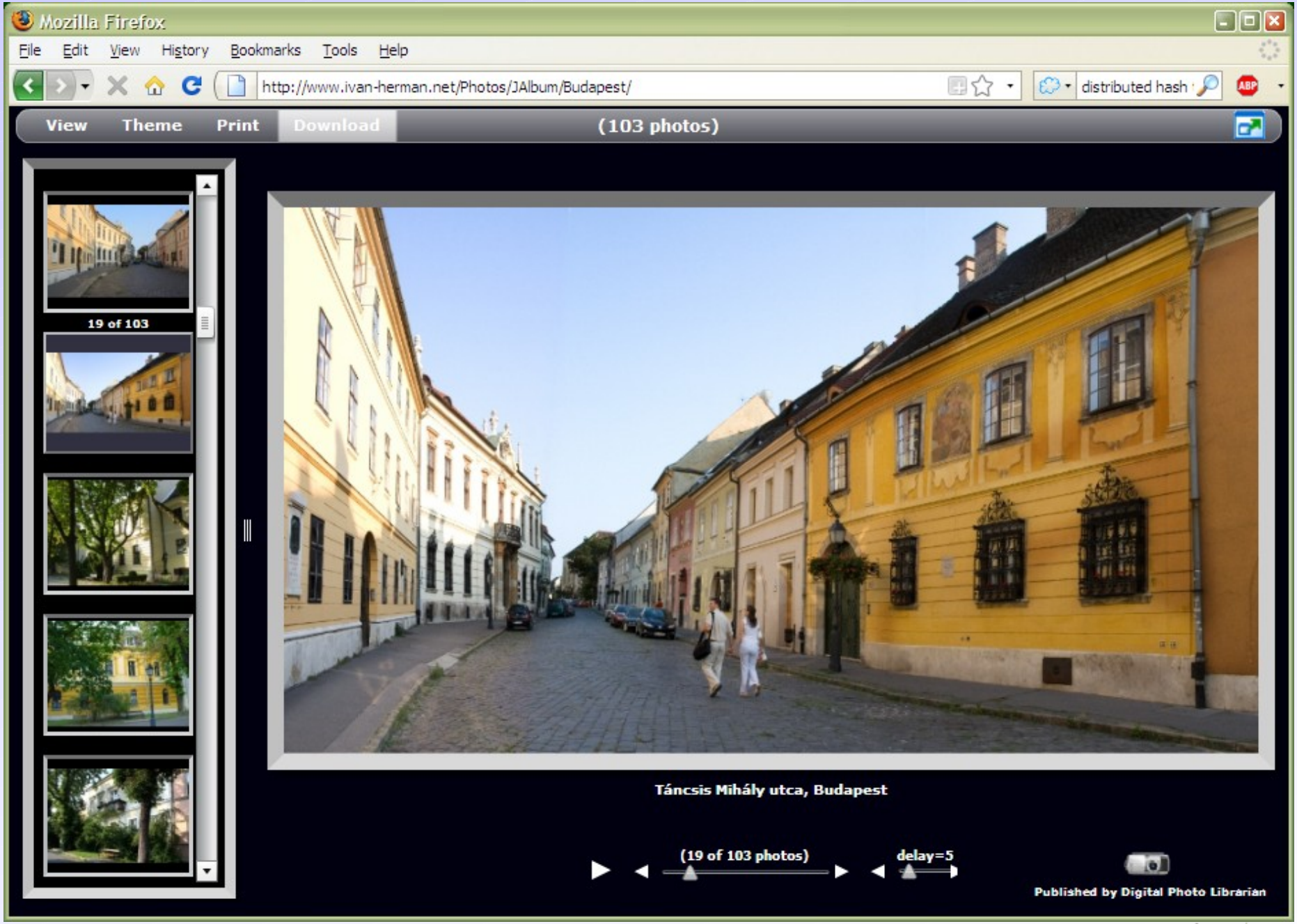

Flickr: Budapest - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.flickr.com/groups/budapest/
easy jet
flickr
Signed in as Ivan Herman
Help
Sign Out
Home
You
Organize
Contacts
Groups
Explore
Search this group's pool
Search
Budapest
Share This
Group Pool
Discussion
1,418 Members
Map
Invite Friends
Group Pool (19,017 items | Add photos or video)
NEW From apuc
NEW From André Fromont
NEW From carlogambino
NEW From carlogambino
NEW From Crashbandi
NEW From Crashbandi
NEW From Crashbandi
NEW From Crashbandi
NEW From Crashbandi
NEW From Crashbandi
NEW From Crashbandi
NEW From Crashbandi
» More
Discussion (33 posts | Post a new topic)

… on Google …



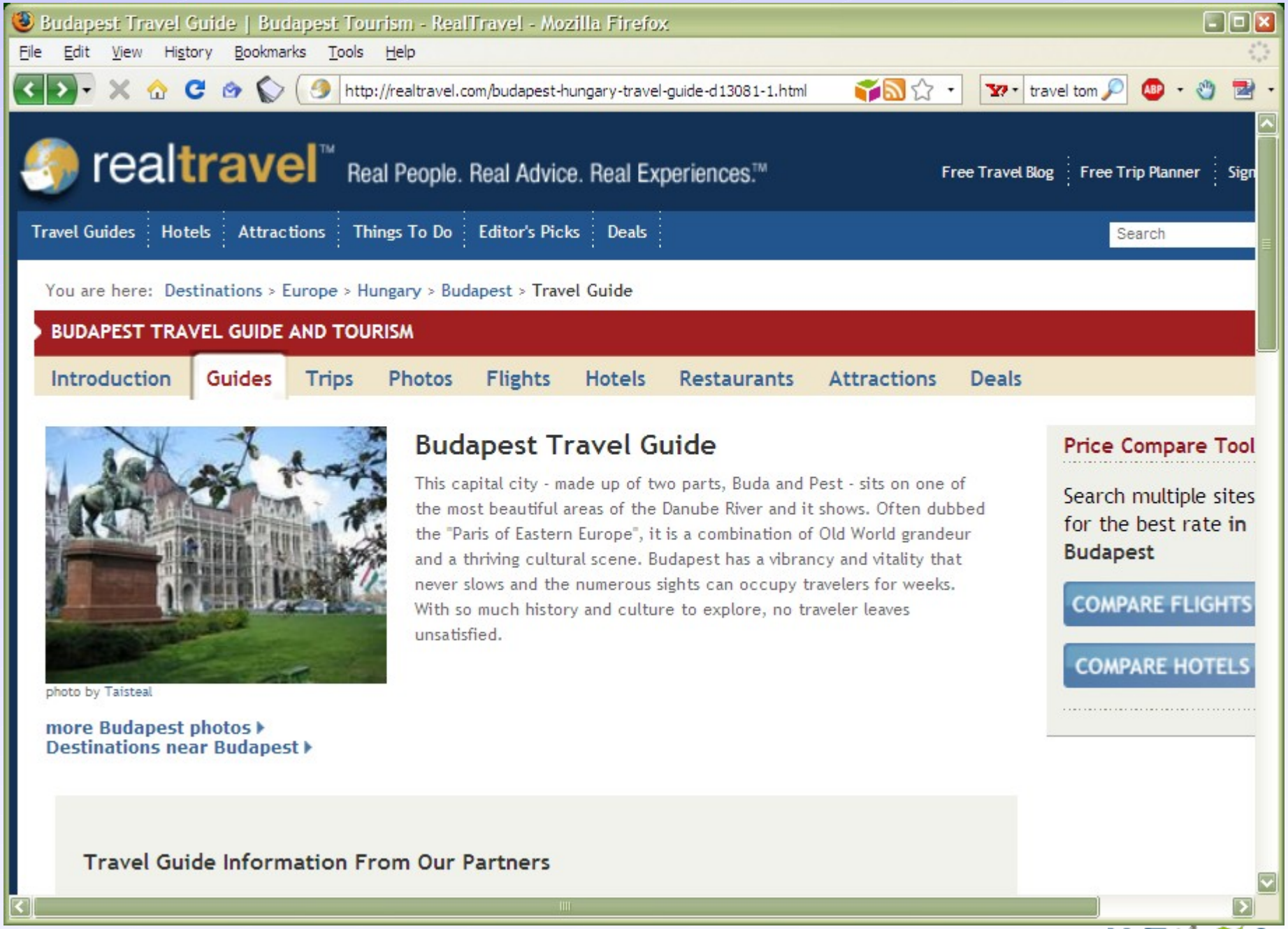
W3C Semantic Web

… or you can look at mine



W3C Semantic Web

… or a (social) travel site



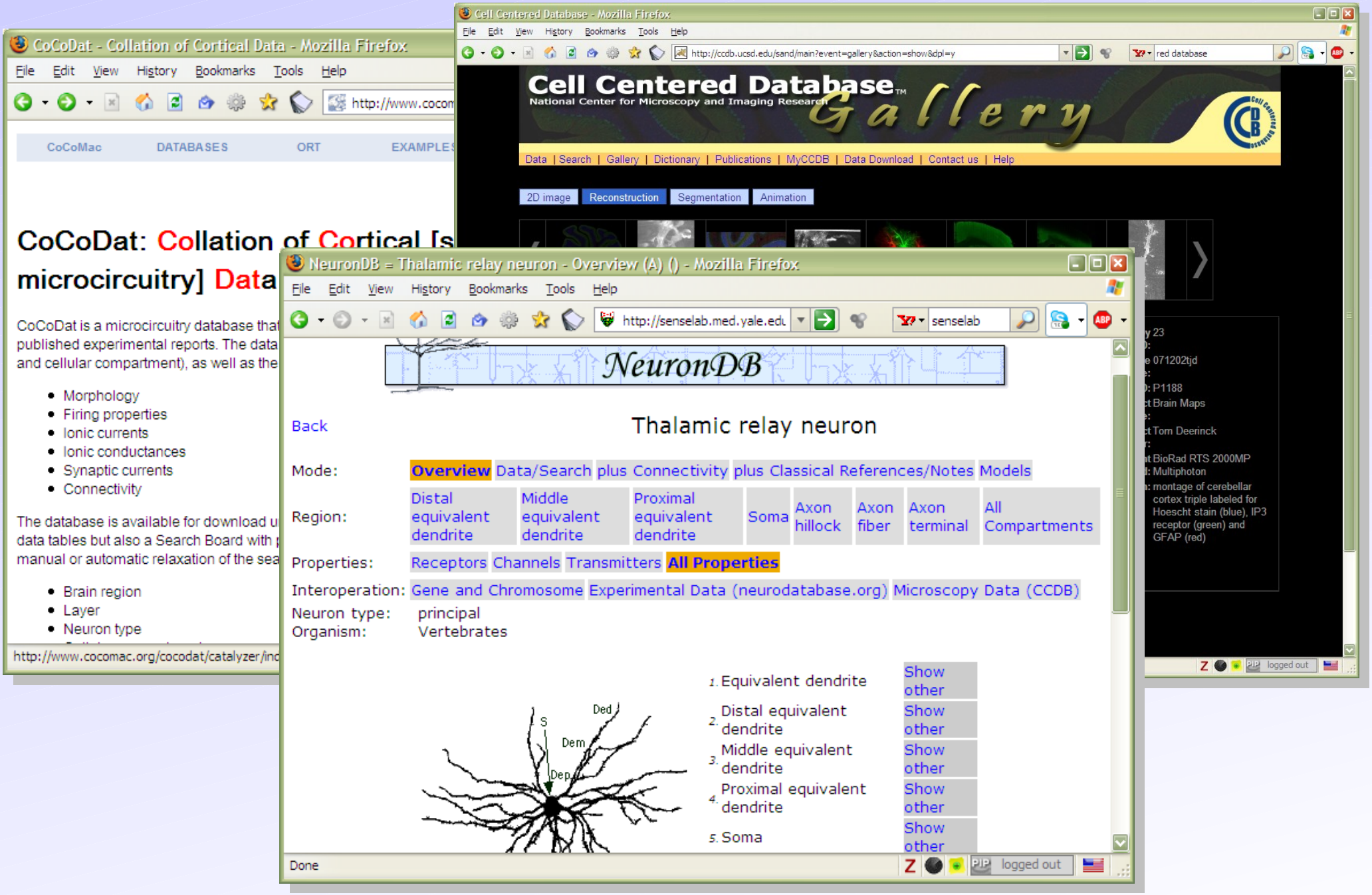
What happened here?

- You had to consult a large number of sites, all different in style, purpose, possibly language…
- You had to mentally *integrate* all those information to achieve your goals
- We all know that, sometimes, this is a long and tedious process!

- All those pages are only tips of respective icebergs:
  - the real *data* is hidden somewhere in databases, XML files, Excel sheets, …
  - you have only access to what the Web page designers allow you to see

- Specialized sites (Expedia, TripAdvisor) do a bit more:
  - they gather and combine data from other sources (usually with the approval of the data owners)
  - but they still control how you see those sources
- But sometimes you want to personalize: access the original data and combine it yourself!

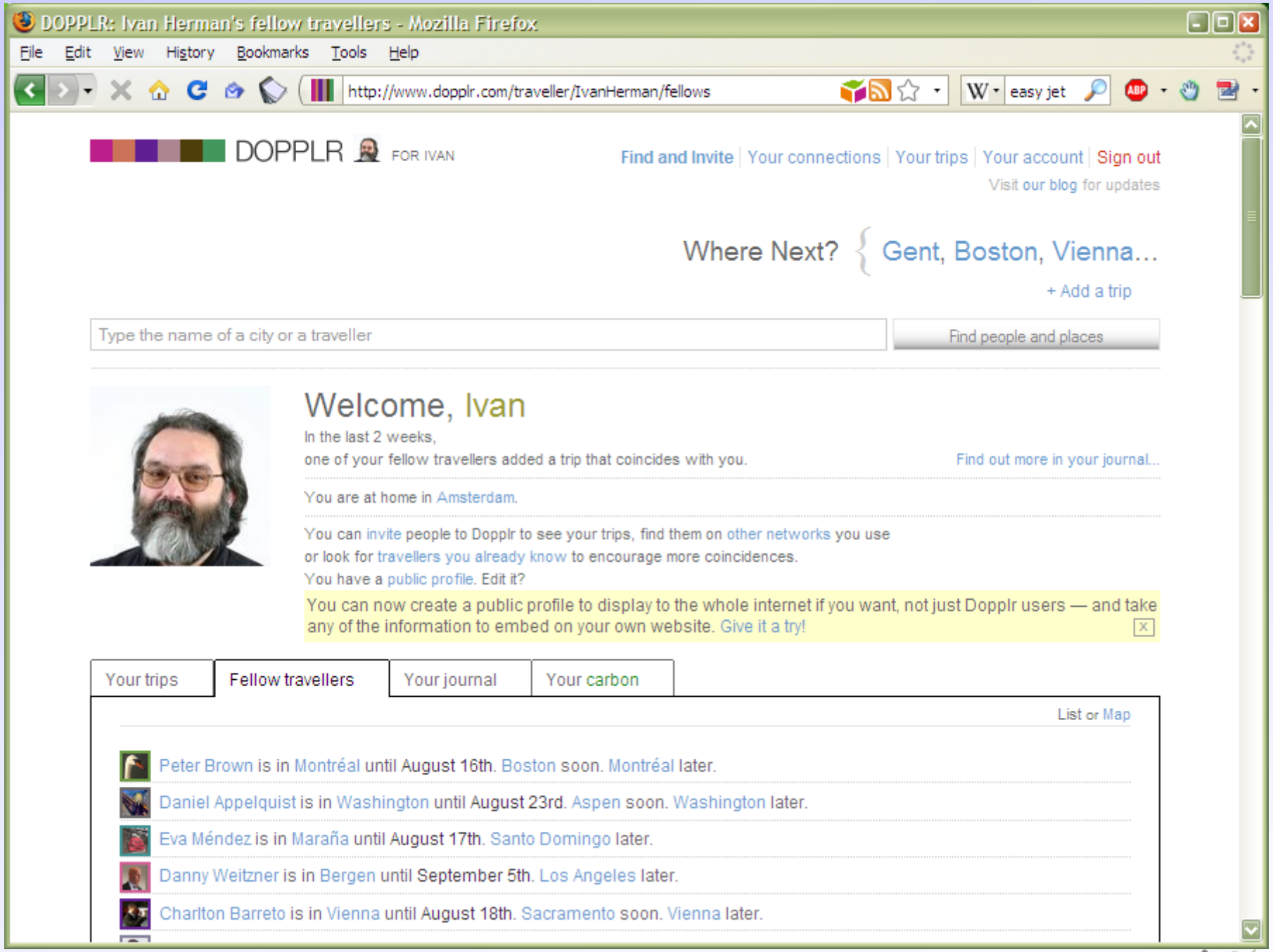
Here is another example…



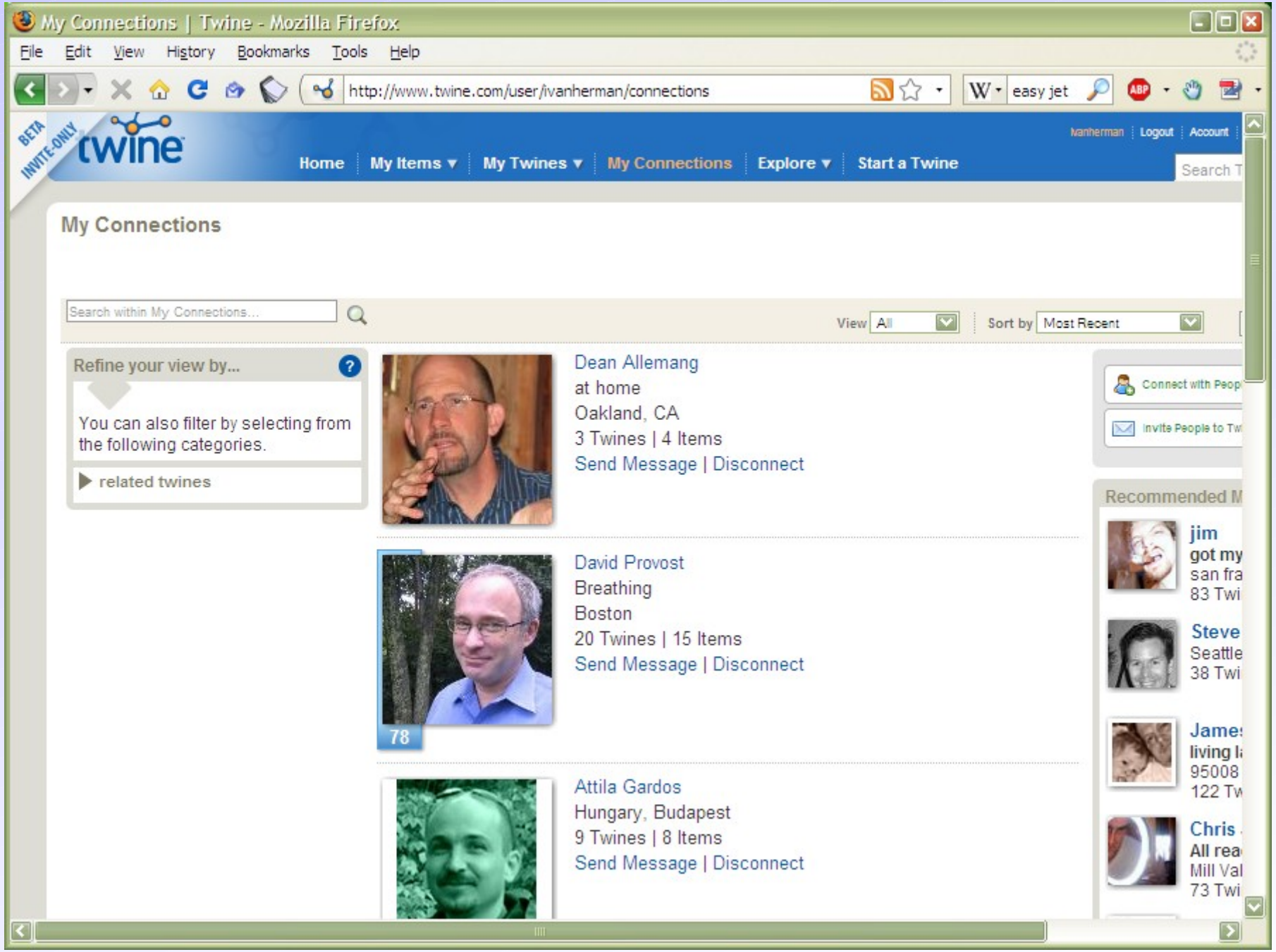
W3C Semantic Web

Another example: social sites. I have a list of "friends" by…

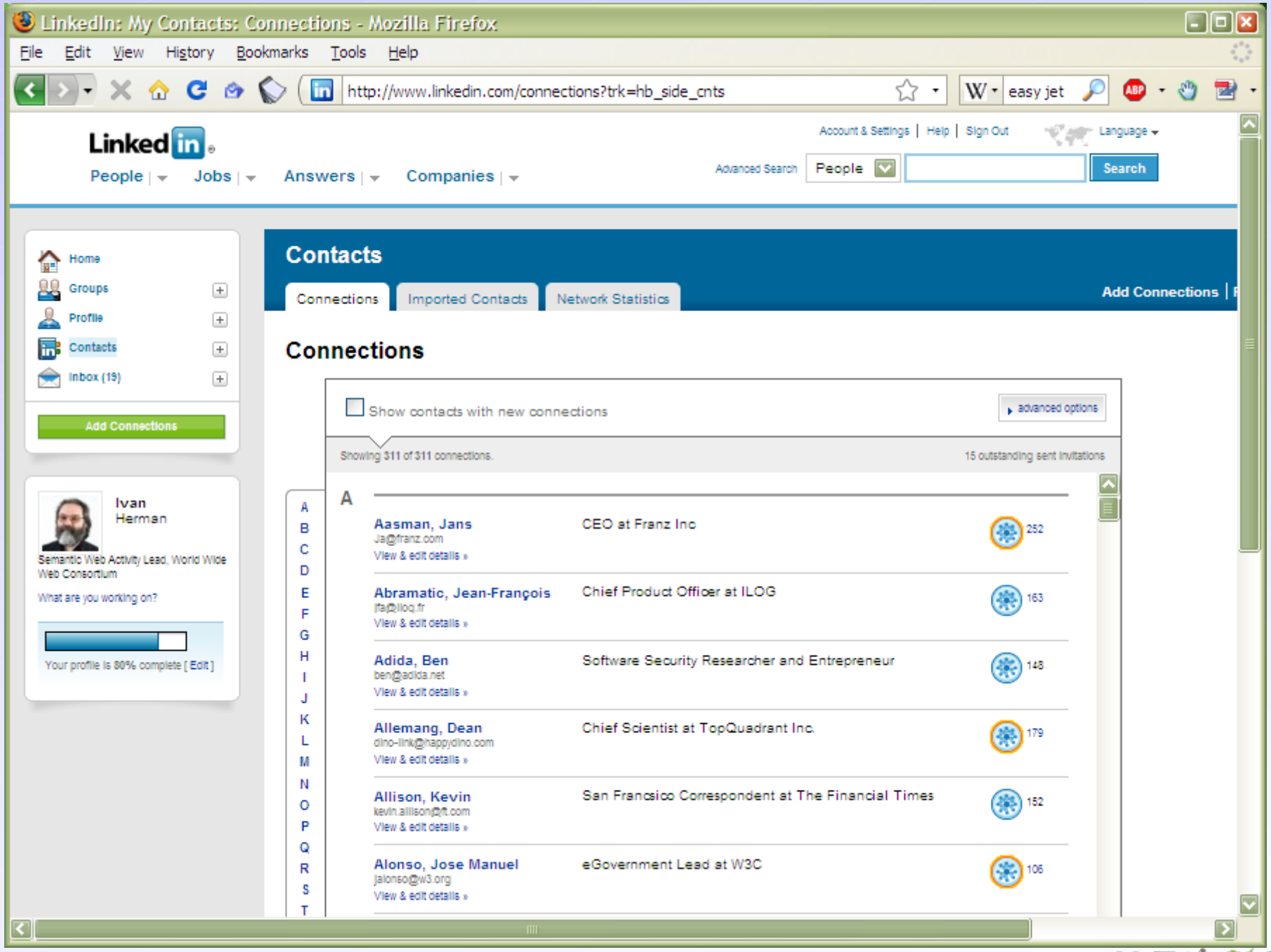
… Dopplr,


DOPPLR: Ivan Herman's fellow travellers - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.dopplr.com/traveller/IvanHerman/fellows
easy jet
DOPPLR FOR IVAN
Find and Invite | Your connections | Your trips | Your account | Sign out
Visit our blog for updates
Where Next? { Gent, Boston, Vienna…
+ Add a trip
Type the name of a city or a traveller
Find people and places
Welcome, Ivan
In the last 2 weeks,
one of your fellow travellers added a trip that coincides with you.
Find out more in your journal...
You are at home in Amsterdam.
You can invite people to Dopplr to see your trips, find them on other networks you use
or look for travellers you already know to encourage more coincidences.
You have a public profile. Edit it?
You can now create a public profile to display to the whole internet if you want, not just Dopplr users — and take any of the information to embed on your own website. Give it a try!
Your trips
Fellow travellers
Your journal
Your carbon
List or Map
Peter Brown is in Montréal until August 16th. Boston soon. Montréal later.
Daniel Appelquist is in Washington until August 23rd. Aspen soon. Washington later.
Eva Méndez is in Maraña until August 17th. Santo Domingo later.
Danny Weitzner is in Bergen until September 5th. Los Angeles later.
Charlton Barreto is in Vienna until August 18th. Sacramento soon. Vienna later.

… Twine,

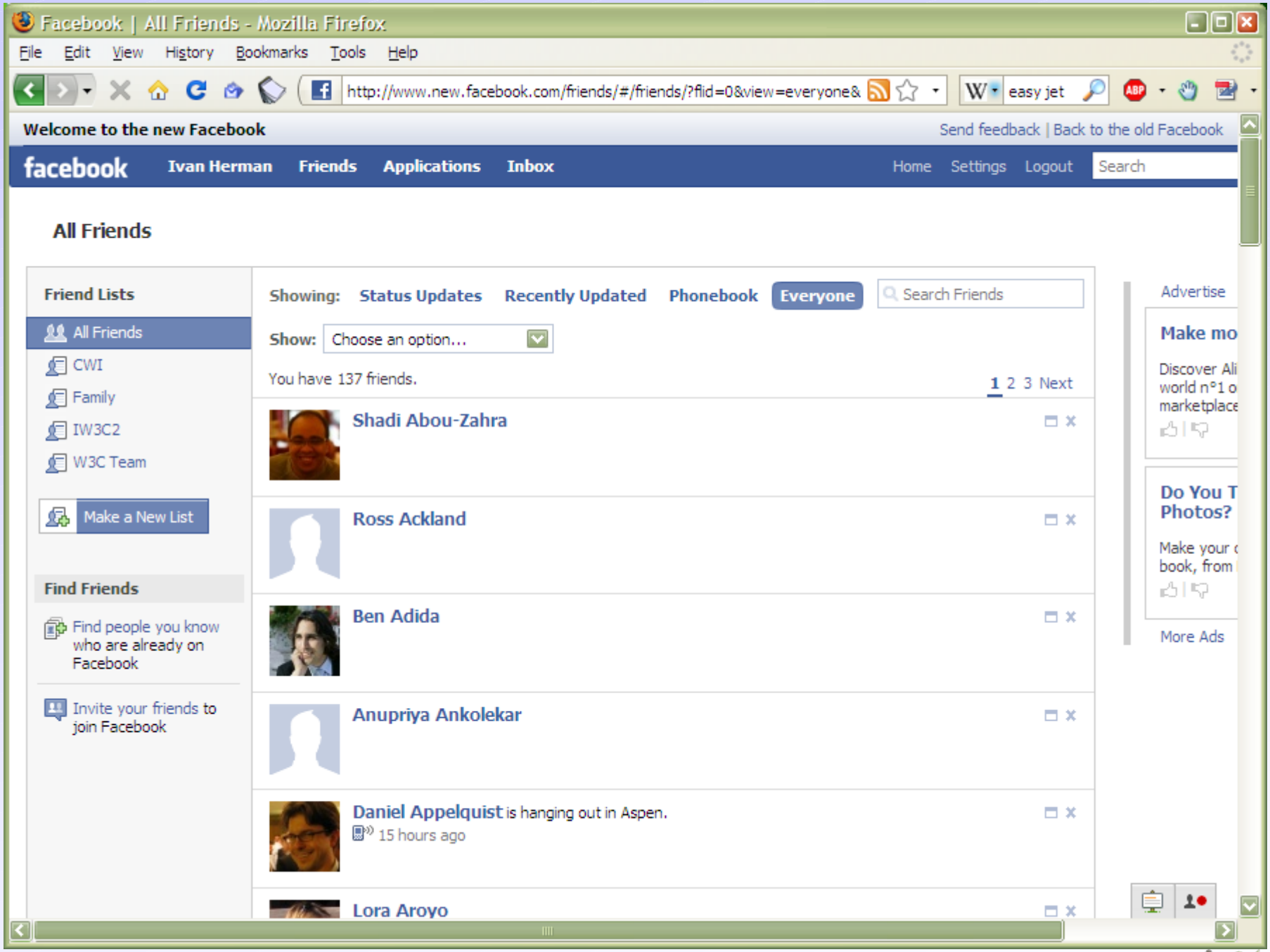

My Connections | Twine - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.twine.com/user/ivanherman/connections
easy jet
BETA INVITE-ONLY
twine
Home
My Items
My Twines
My Connections
Explore
Start a Twine
Logout
Account
My Connections
Search within My Connections...
View All
Sort by Most Recent
Refine your view by...
You can also filter by selecting from the following categories.
related twines
Dean Allemang
at home
Oakland, CA
3 Twines | 4 Items
Send Message | Disconnect
David Provost
Breathing
Boston
20 Twines | 15 Items
Send Message | Disconnect
78
Attila Gardos
Hungary, Budapest
9 Twines | 8 Items
Send Message | Disconnect
Connect with Peop
Invite People to Tw
Recommended M
jim
got my
san fra
83 Twi
Steve
Seattle
38 Twi
Jame
living l
95008
122 Tw
Chris
All rea
Mill Val
73 Twi

… LinkedIn,



W3C Semantic Web

… and, of course, Facebook



- I had to type in and connect with friends again and again for each site independently 
- This is even worse then before: *I* feed the icebergs, but I still do not have an easy access to data…

What would we like to have?

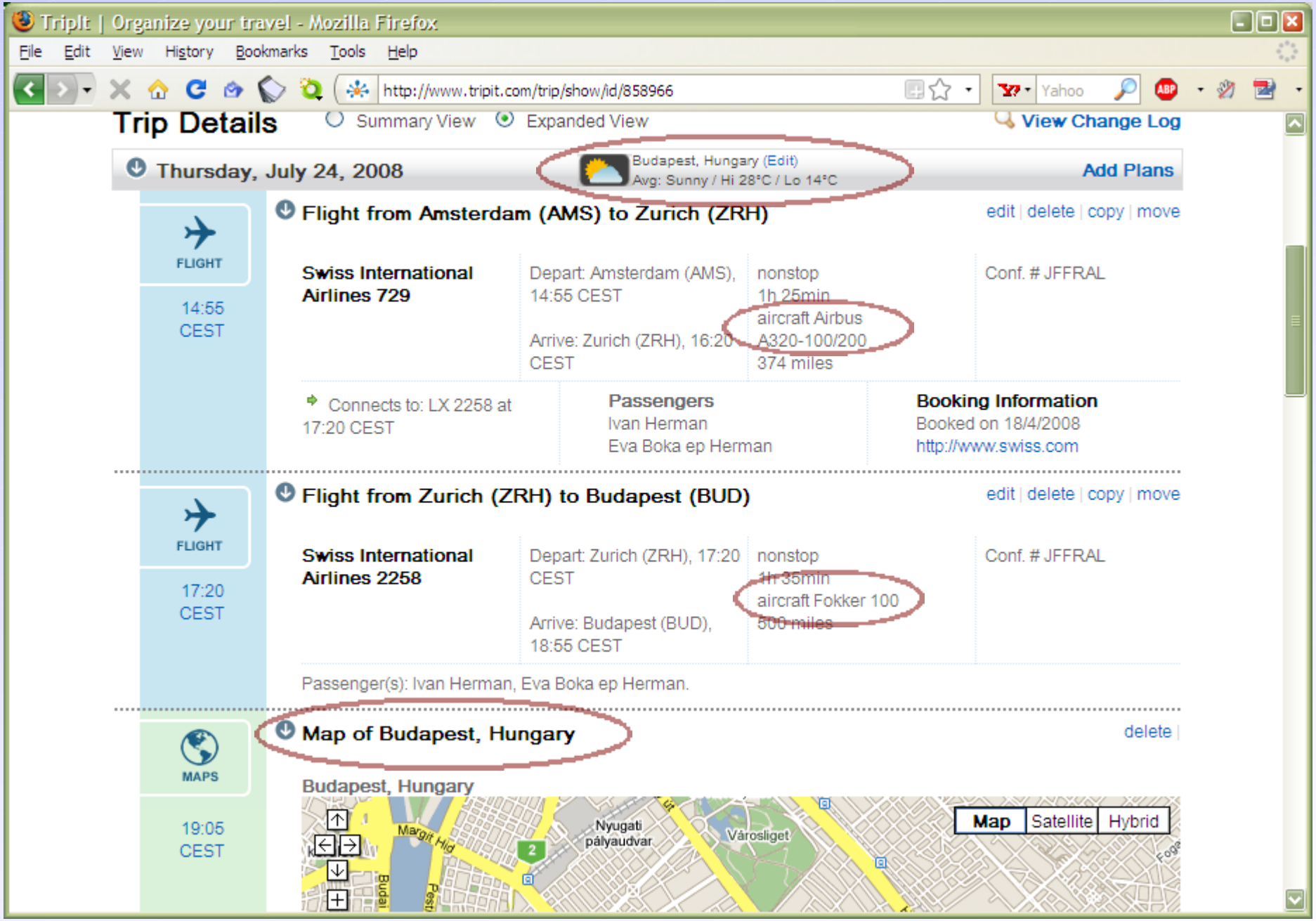
- Use the data on the Web the same way as we do with documents:
  - be able to link to data (independently of their presentation)
  - use that data the way I want (present it, mine it, etc)
  - agents, programs, scripts, etc, should be able to *interpret* part of that data

Put it another way…

- We would like to *extend* the current Web to a "*Web of data*":
  - allow for applications to exploit the data directly

But wait! Isn't what mashup sites are already doing?

A “mashup” example:



- In some ways, yes, and that shows the huge power of what such Web of data provides
- But mashup sites are forced to do very ad-hoc jobs
  - various data sources expose their data via Web Services
  - each with a different API, a different logic, different structure
  - these sites are forced to reinvent the wheel many times because there is no standard way of doing things 

Put it another way (again)…

- We would like to extend the current Web to a *standard* way for a “Web of data”

But what does this mean?

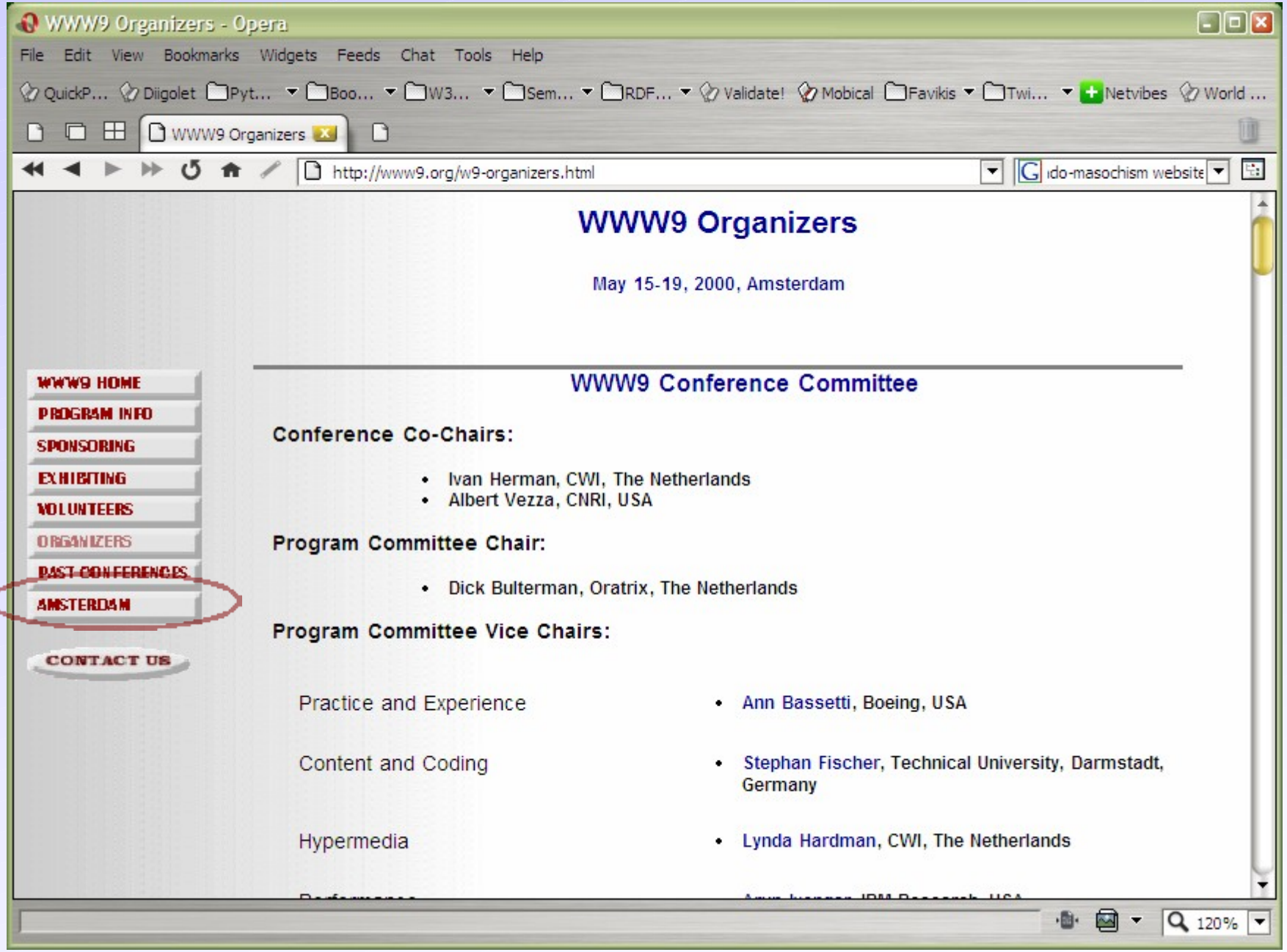
- What makes the current (document) Web work?
  - people create different documents
  - they give an address to it (ie, a URI) and make it accessible to others on the Web

Steven's site on Amsterdam (done for some visiting friends)

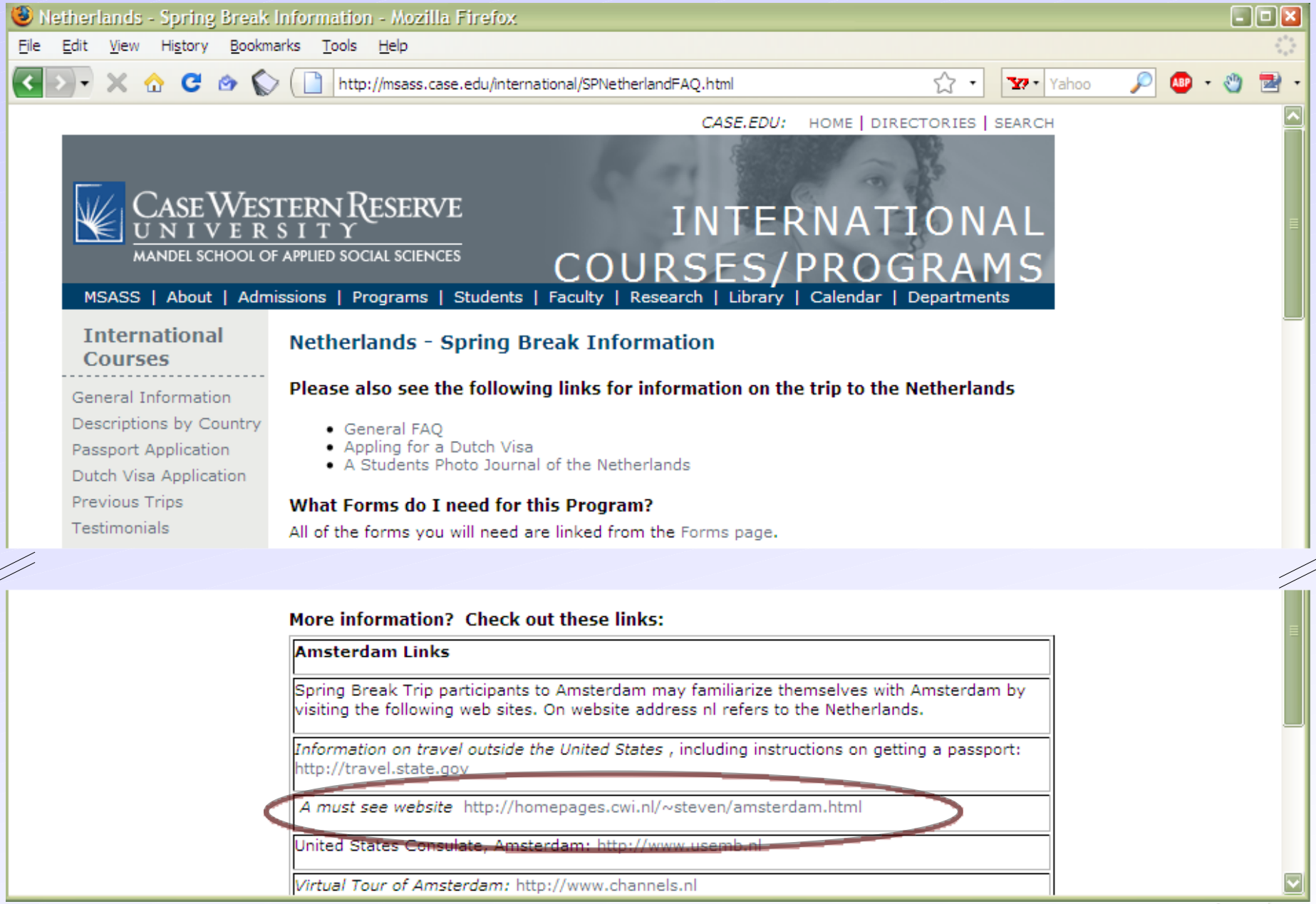
Then some magic happens…

- Others discover the site and they link to it
- The more they link to it, the more important and well known the page becomes
  - remember, this is what, eg, Google exploits!
- This is the “Network effect”: some pages become important, and others begin to rely on it *even if the author did not expect it…*

This could be expected…



but this one, from the other side of the Globe, was not…



What would that mean for a Web of Data?

- Lessons learned: we should be able to:
  - "publish" the data to make it known on the Web
    - standard ways should be used instead of ad-hoc approaches
    - the analogous approach to documents: *give URI-s to the data*
  - make it possible to "link" to that URI from *other* sources of data (not only Web pages)
    - ie, applications should not be forced to make targeted developments to access the data
    - generic, standard approaches should suffice
  - and let the network effect work its way…

But it is a little bit more complicated

- On the traditional Web, humans are implicitly taken into account
- A Web link has a “context” that a person may use

Eg: address field on my page:

Ivan Herman - Mozilla Firefox

File Edit View History Bookmarks Tools Help

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

Ivan Herman

My Work at W3C | Contact information | Short CV | Upcoming trips | Public presentations

My Work at W3C

I am Semantic Web Activity Lead; that is my main work at W3C. I am member of IW3C2 (International World Wide Web Conference Committee) (the committee coordinating the yearly WWW conference series), serving as a liaison for W3C, and of SWSA (Semantic Web Science Association), the committee responsible for the International Semantic Web Conferences series.

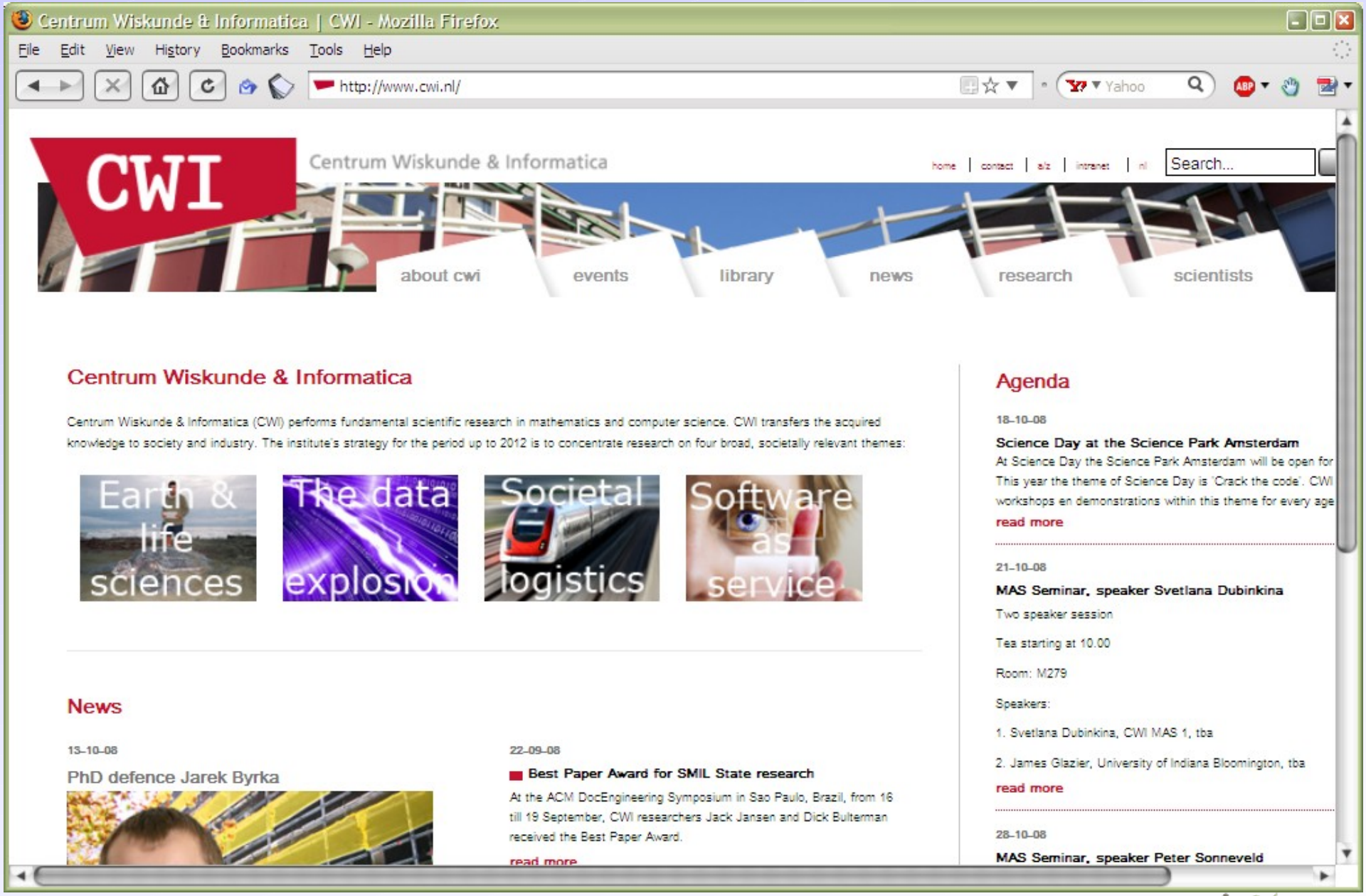
As part of my work, I also participate in lots of outreach activities, and I regularly make presentations, tutorials, etc. You can consult my list of presentations for further details.

Contact information

Email:
ivan@w3.org
(sha1sum: 5ac8032d5f6012aa1775ea2f63e1676bafd5e80b)
Postal address:
C/o Centre for Mathematics and Computer Sciences (CWI)
Kruislaan 413, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands.
Phone numbers:
phone: +31-20-5924163
mobile phone: +31-641044153
fax: +31-20-5924312
PGP/GPG:
My GnuPGP key and signature is available on-line.
FOAF:
You can either extract a short FOAF information from this page, of consult my more complete, public FOAF file.
Misc:

W3C Semantic Web

… leading to this page



- A human understands that this is my institution's home page
- He/she knows what it means (realizes that it is a research institute in Amsterdam)
- On a Web of Data, something is missing; machines can't make sense of the link alone

- New lesson learned:
  - extra information (“label”) must be added to a link: “this links to my institution, which is a research institute”
  - this information should be machine readable
  - this is a *characterization* (or “classification”) of *both* the link *and* its target
  - in some cases, the classification should allow for some limited “reasoning”

Let us put it together

- What we need for a Web of Data:
  - use URI-s to publish data, not only full documents
  - allow the data to link to other data
  - characterize/classify the data and the links (the “terms”) to convey some extra meaning
  - and use standards for all these!

So What is the Semantic Web?

It is a collection of standard technologies to realize a Web of Data

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

In what follows…

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

The rough structure of data integration

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

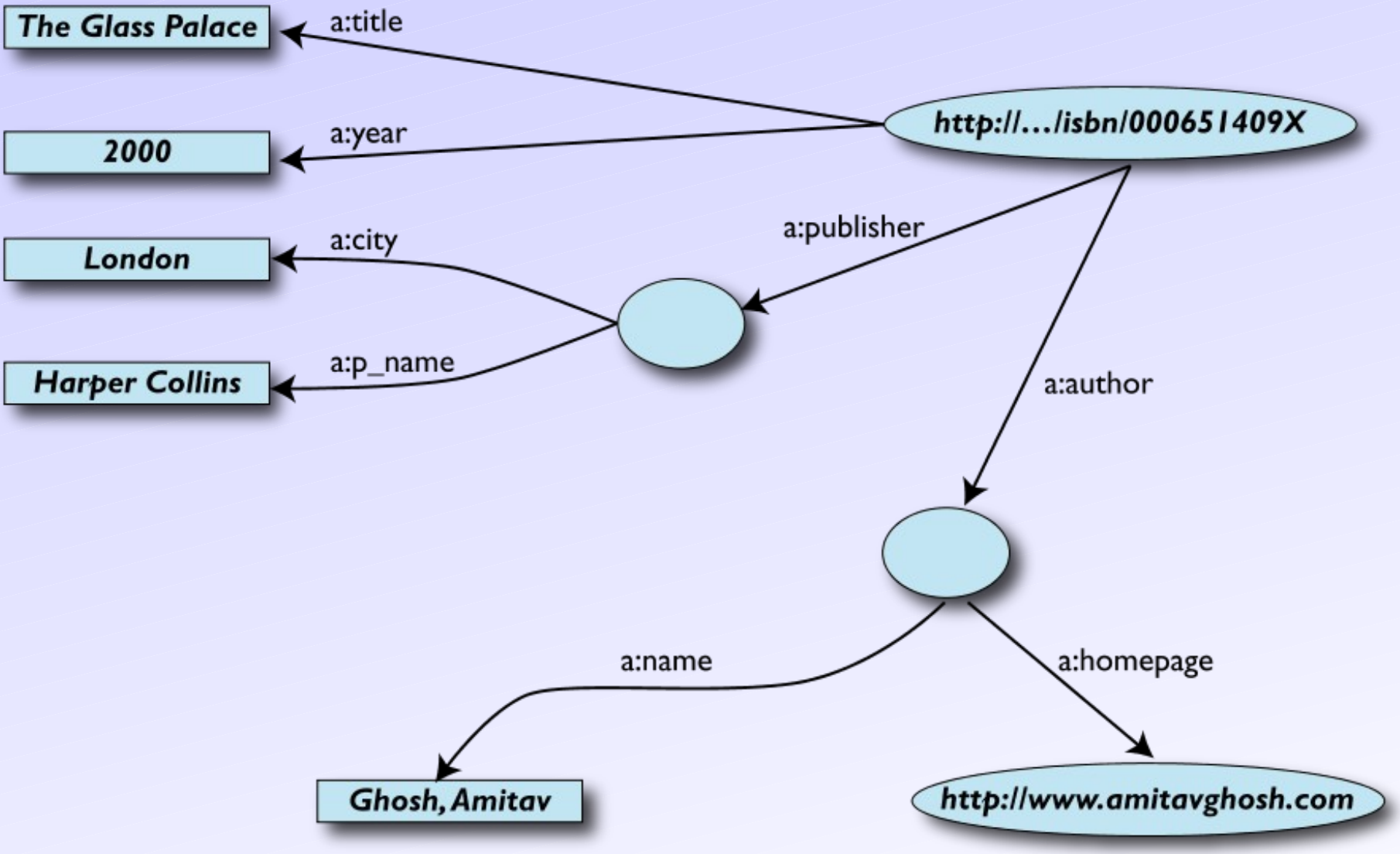
A *simplified* bookstore data (dataset "A")

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

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

ID	Publ. Name	City
id_qpr	Harper Collins	London

1st: export your data as a set of relations



Some notes on the exporting the data

- Relations form a graph
  - the nodes refer to the “real” data or contain some literal
  - how the graph is represented in machine is immaterial for now
- Data export does *not* necessarily mean physical conversion of the data
  - relations can be generated on-the-fly at query time
    - via SQL “bridges”
    - scraping HTML pages
    - extracting data from Excel sheets
    - etc.
- One can export *part* of the data

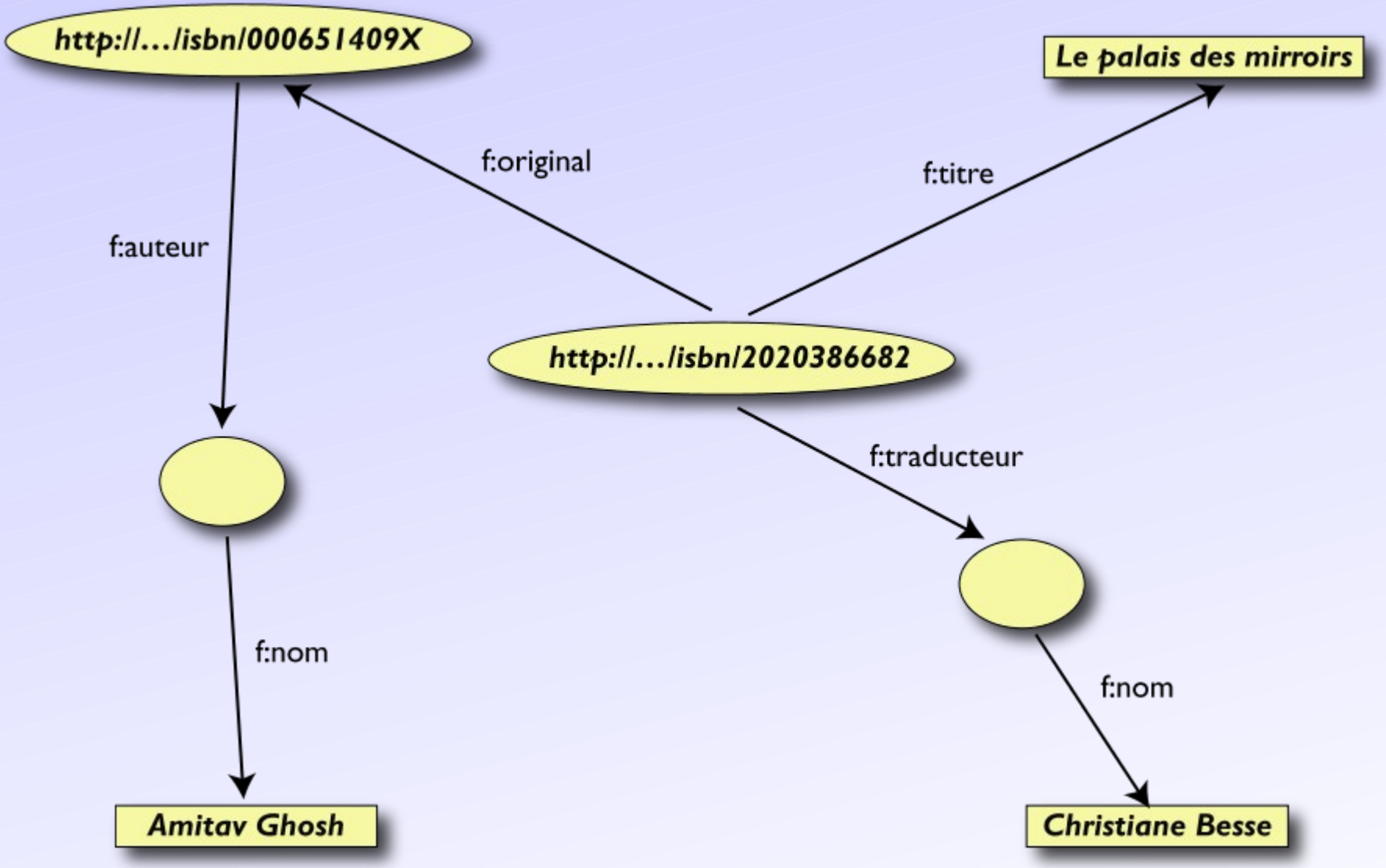
Another *bookstore data (dataset "F")*

	A	B	D	E
1	**ID**	**Titre**	**Traducteur**	**Original**
2	ISBN0 2020386682	Le Palais des miroirs	A13	ISBN-0-00-651409-X
3				

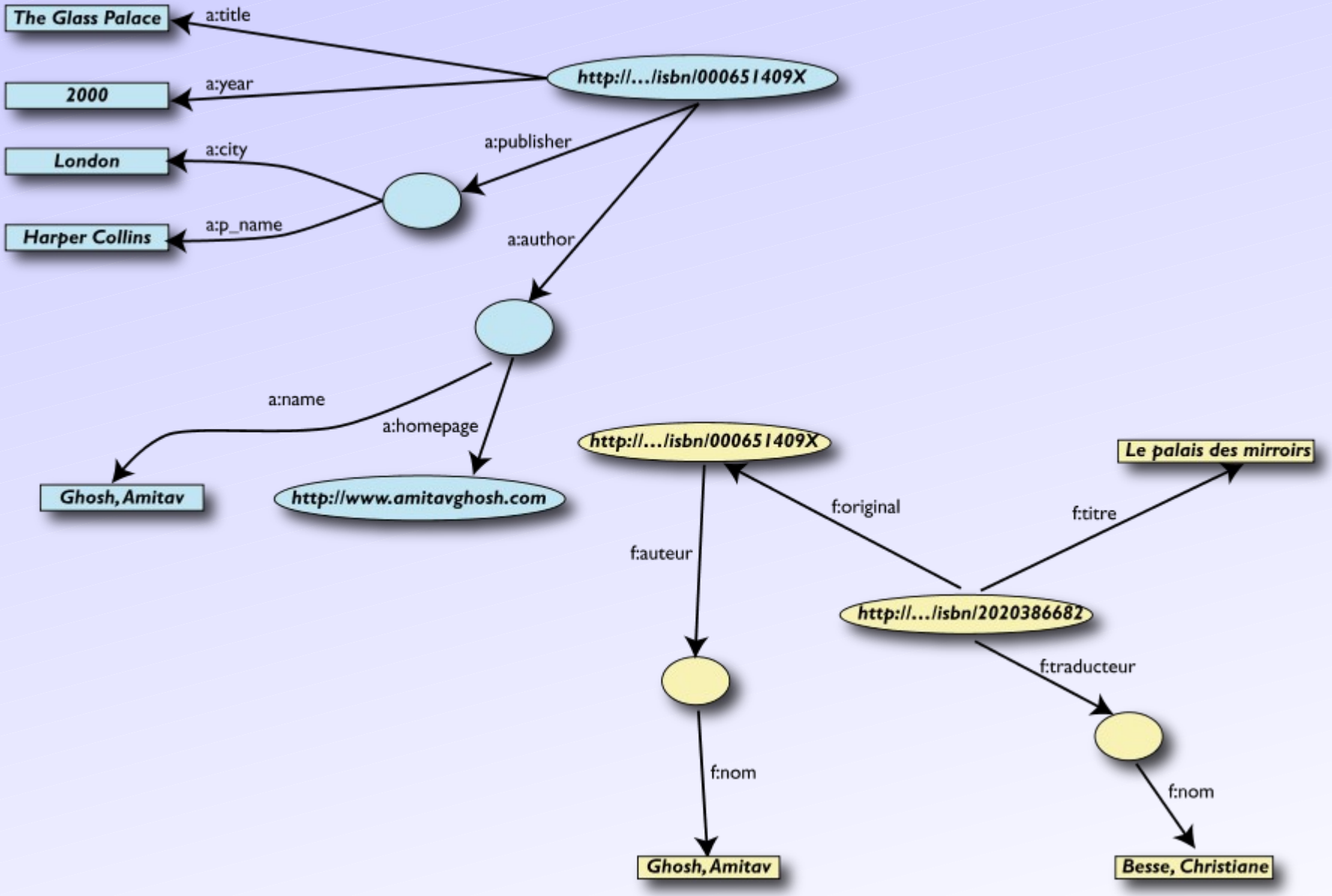
6	**ID**	**Auteur**
7	ISBN-0-00-651409-X	A12

11	**Nom**
12	Ghosh, Amitav
13	Besse, Christianne

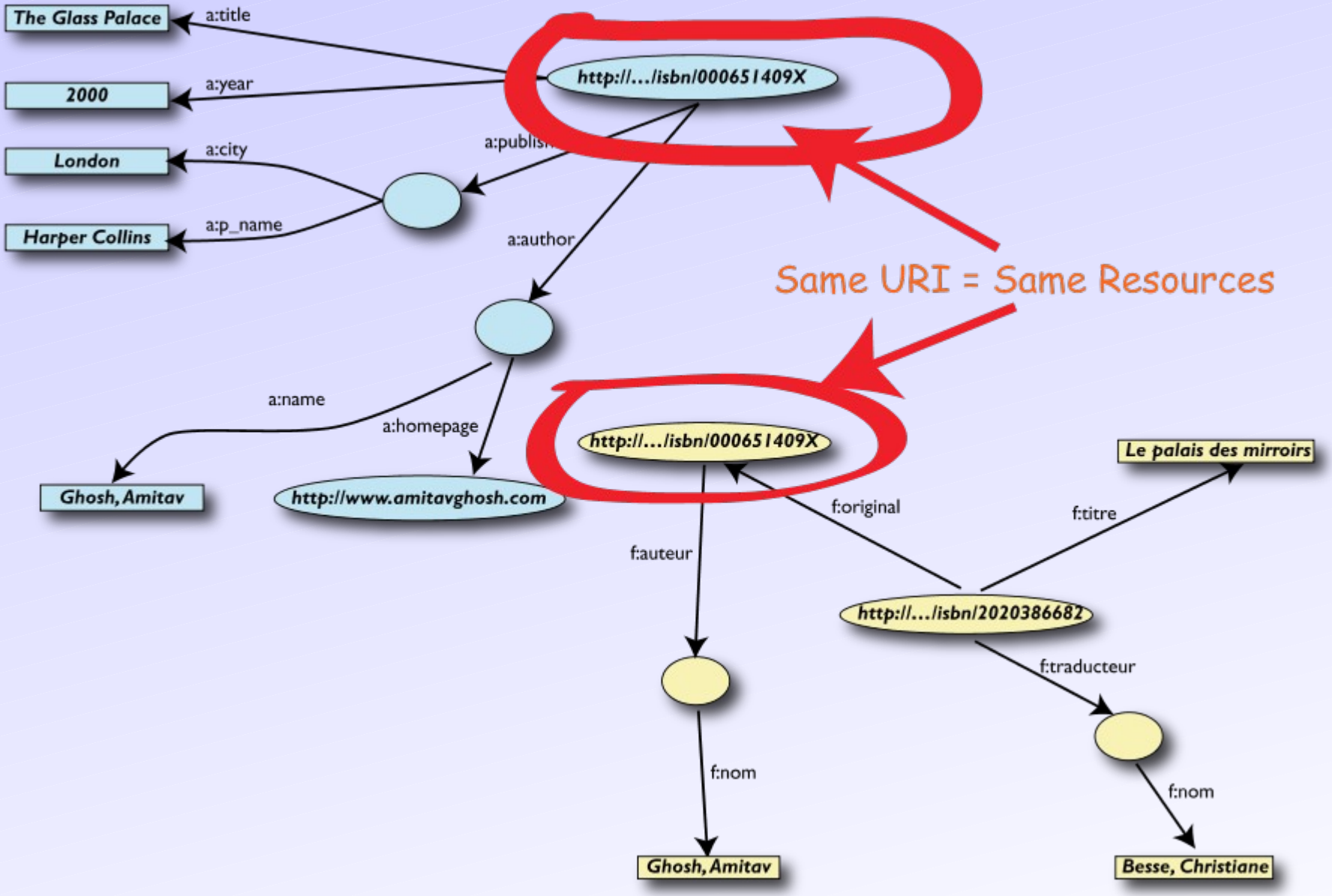
2nd: export your second set of data



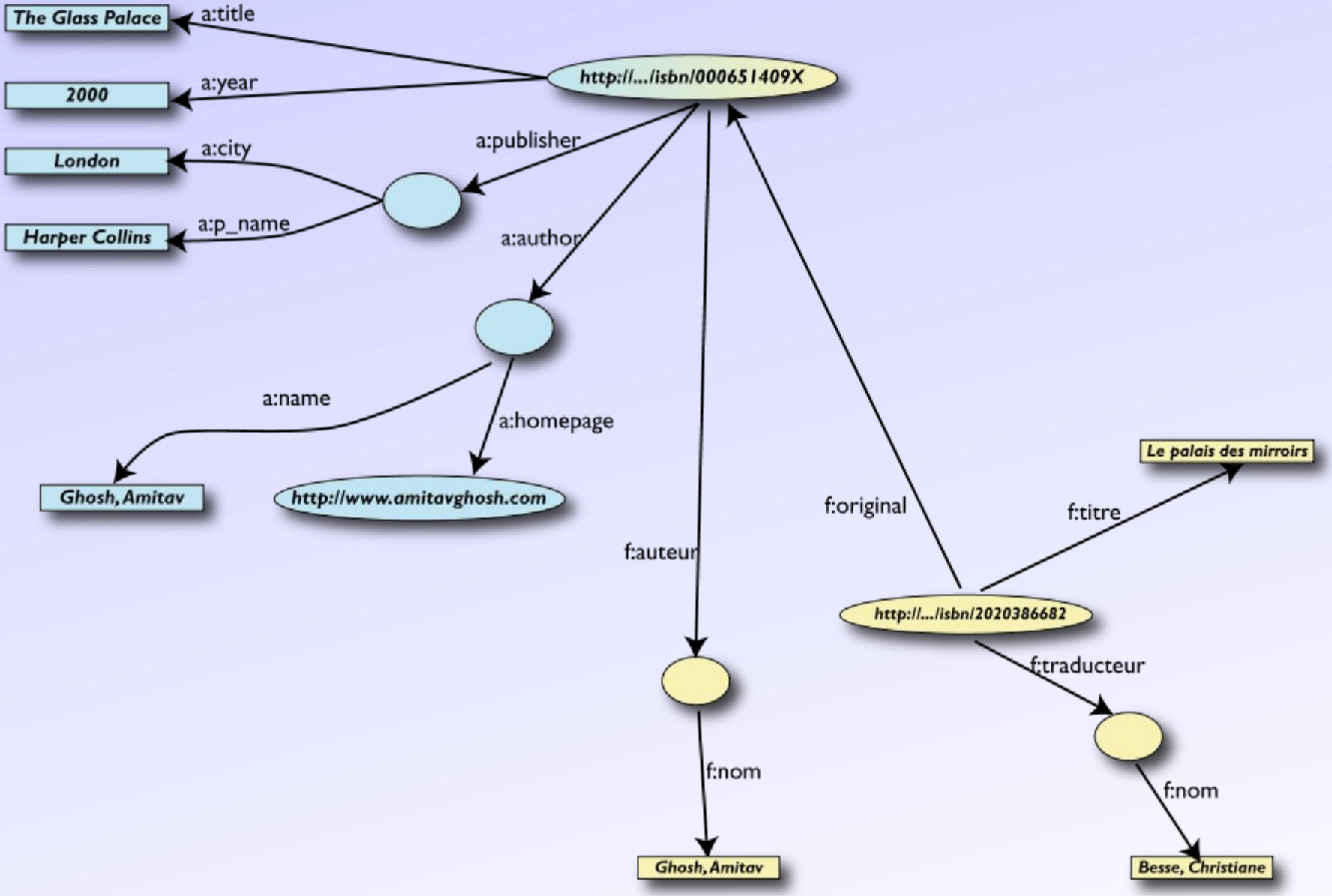
3rd: start merging your data



3rd: start merging your data (cont.)

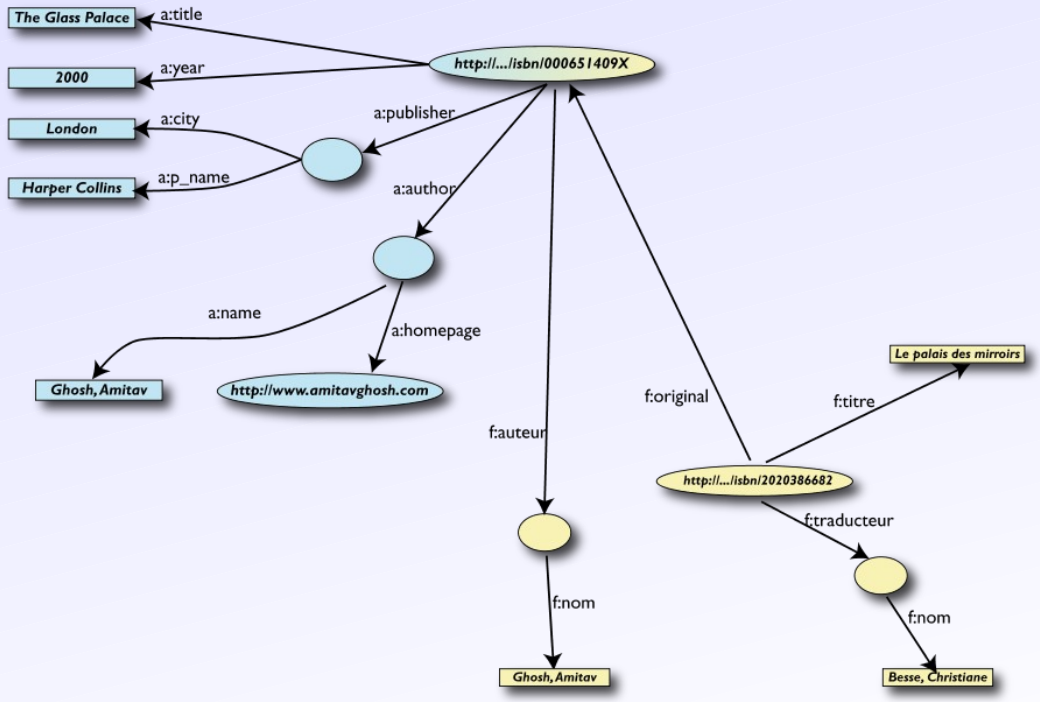


3rd: merge identical resources



Start making queries…

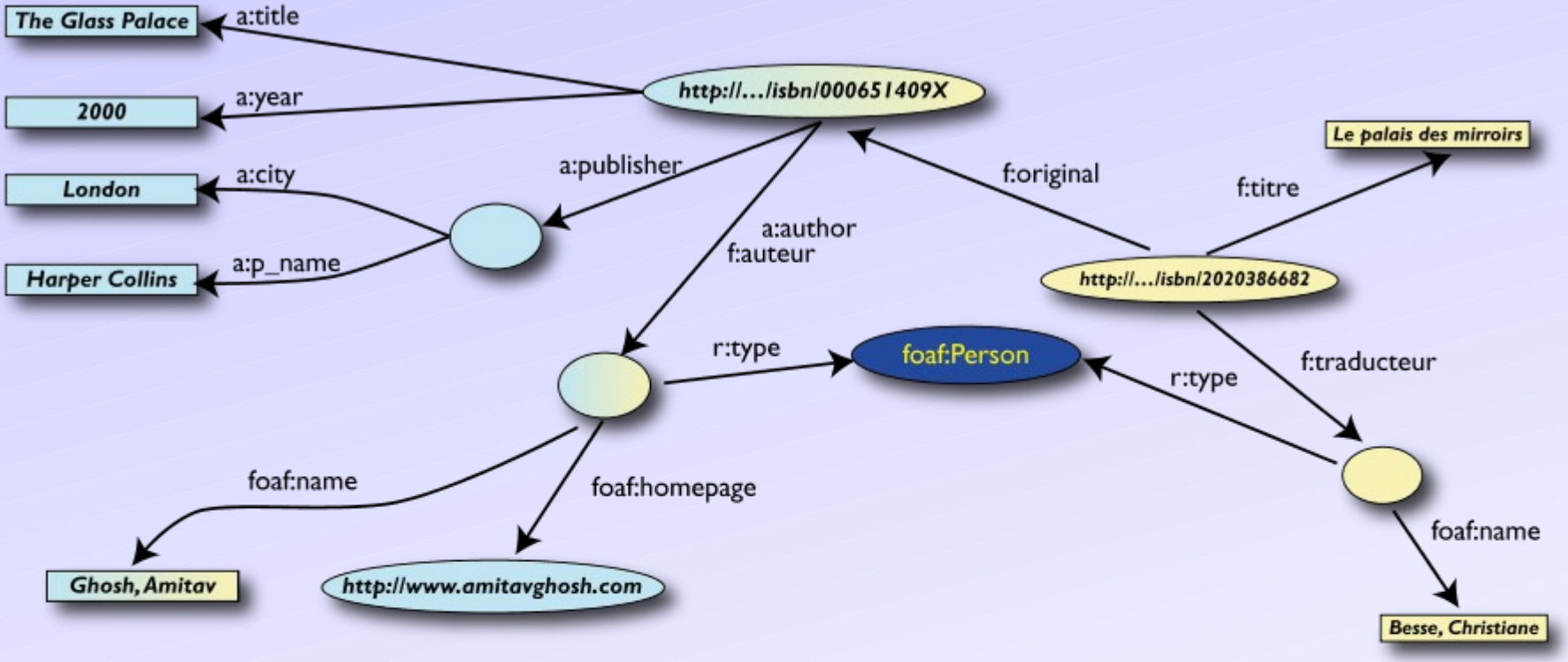
- User of data “F” can now ask queries like:
  - “give me the title of the original”
    - well, … « donnes-moi le titre de l’original »
- This information is not in the dataset “F”…
- …but can be retrieved by merging with dataset “A”!

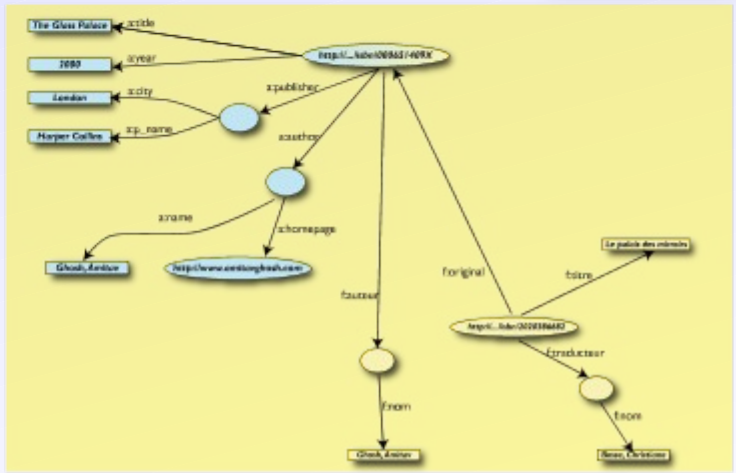


However, more can be achieved…

- 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

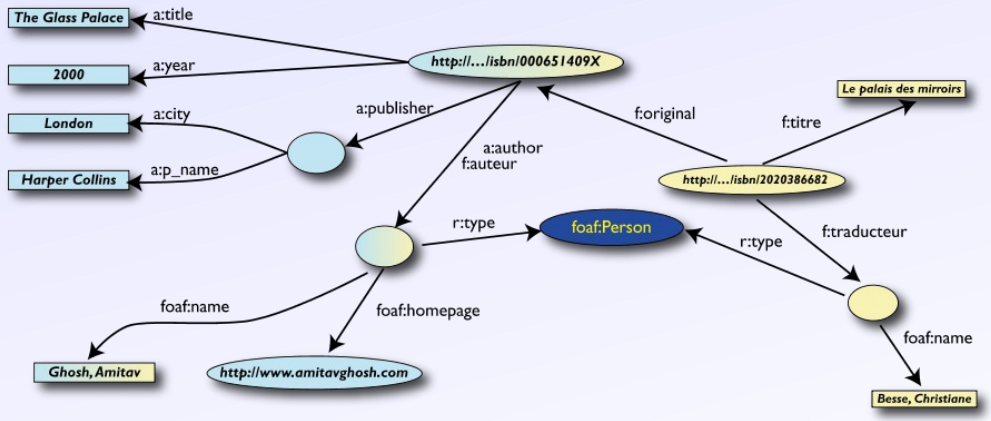
3rd revisited: use the extra knowledge





Start making richer queries!

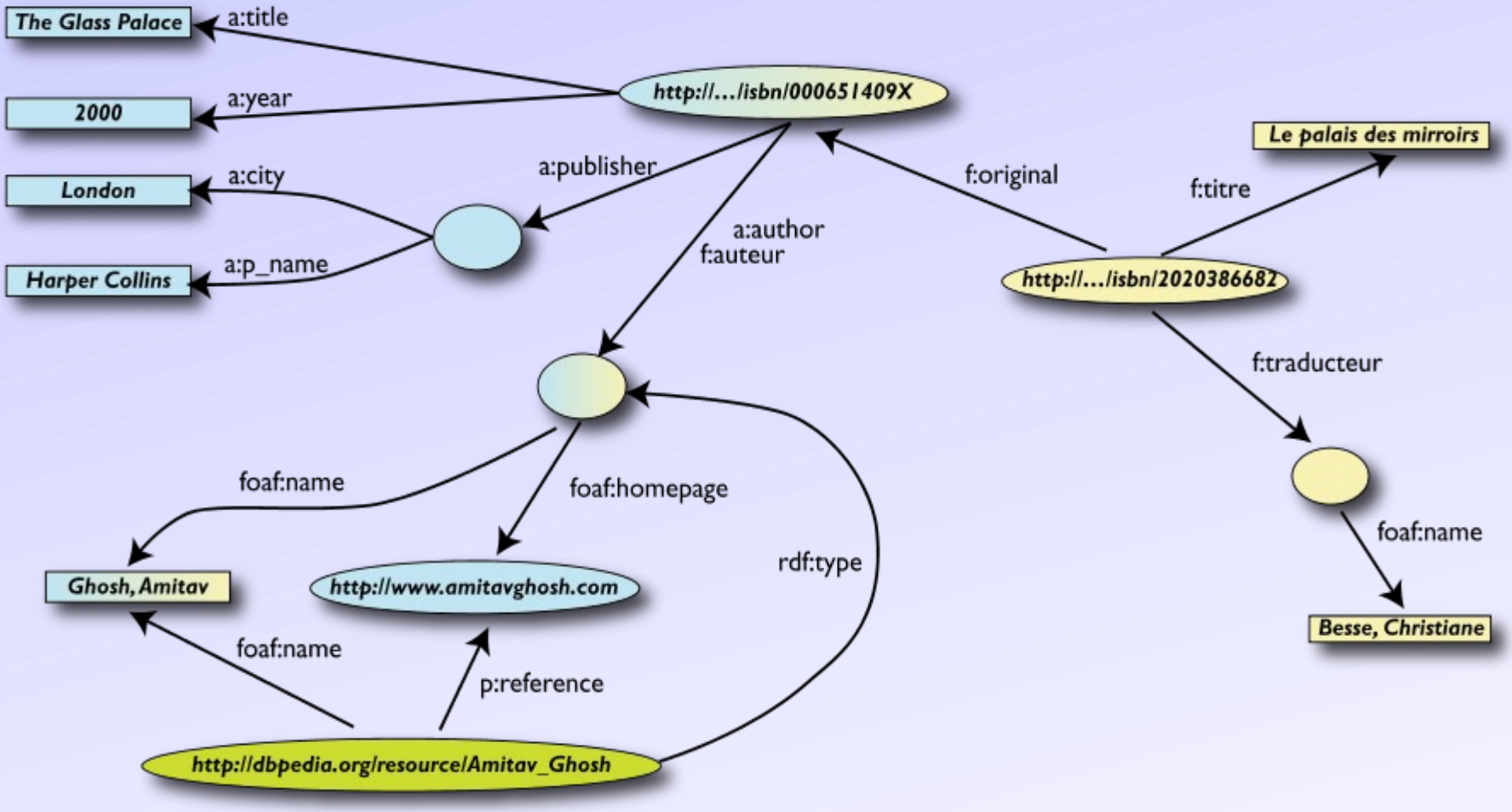
- User of dataset “F” can now query:
  - “donnes-moi la page d’accueil de l’auteur de l’originale”
    - well… “give me the home page of the original’s ‘auteur’”
- The information is not in datasets “F” or “A”…
- …but was made available by:
  - merging datasets “A” and datasets “F”
  - adding three simple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



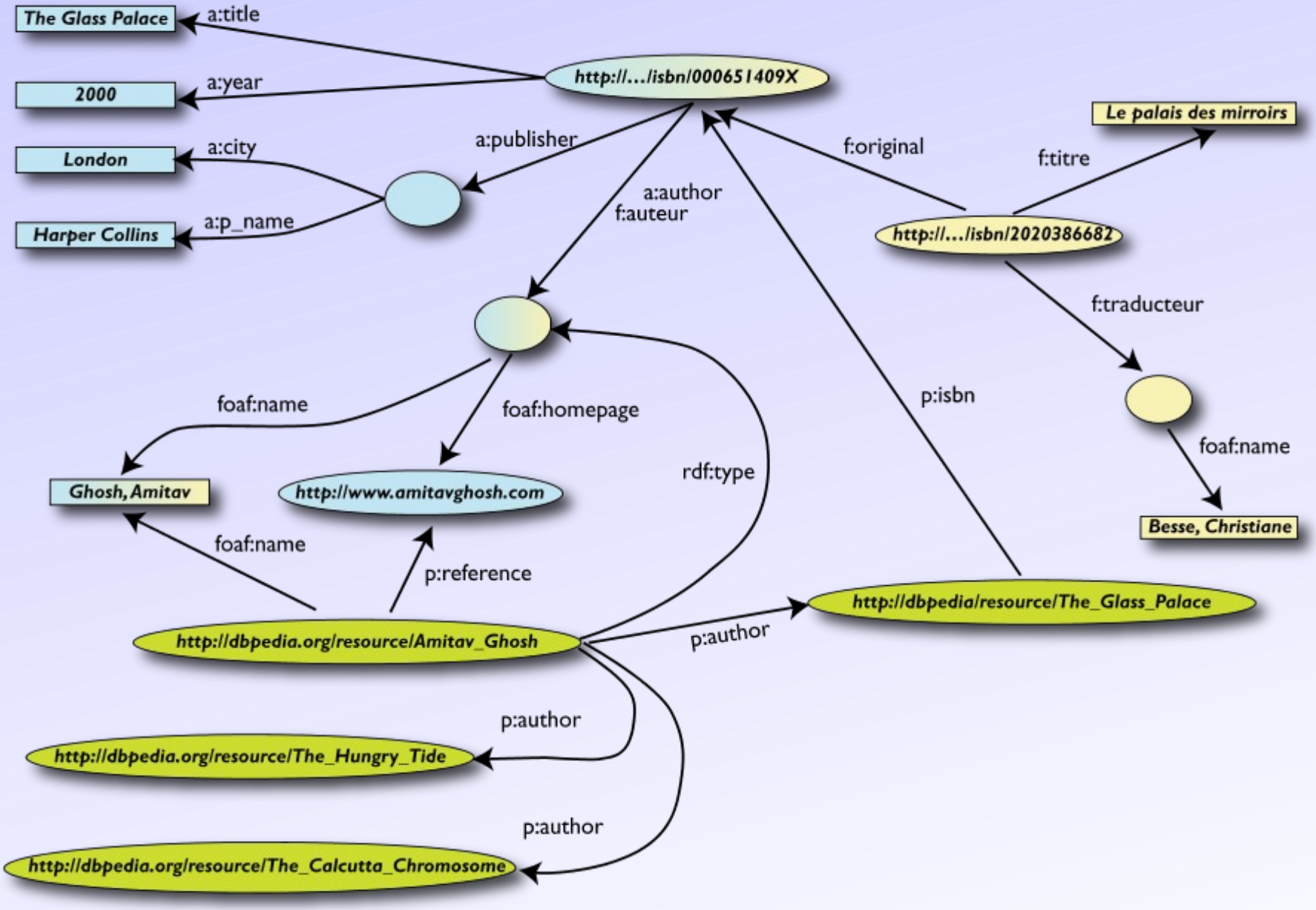
Combine with different datasets

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

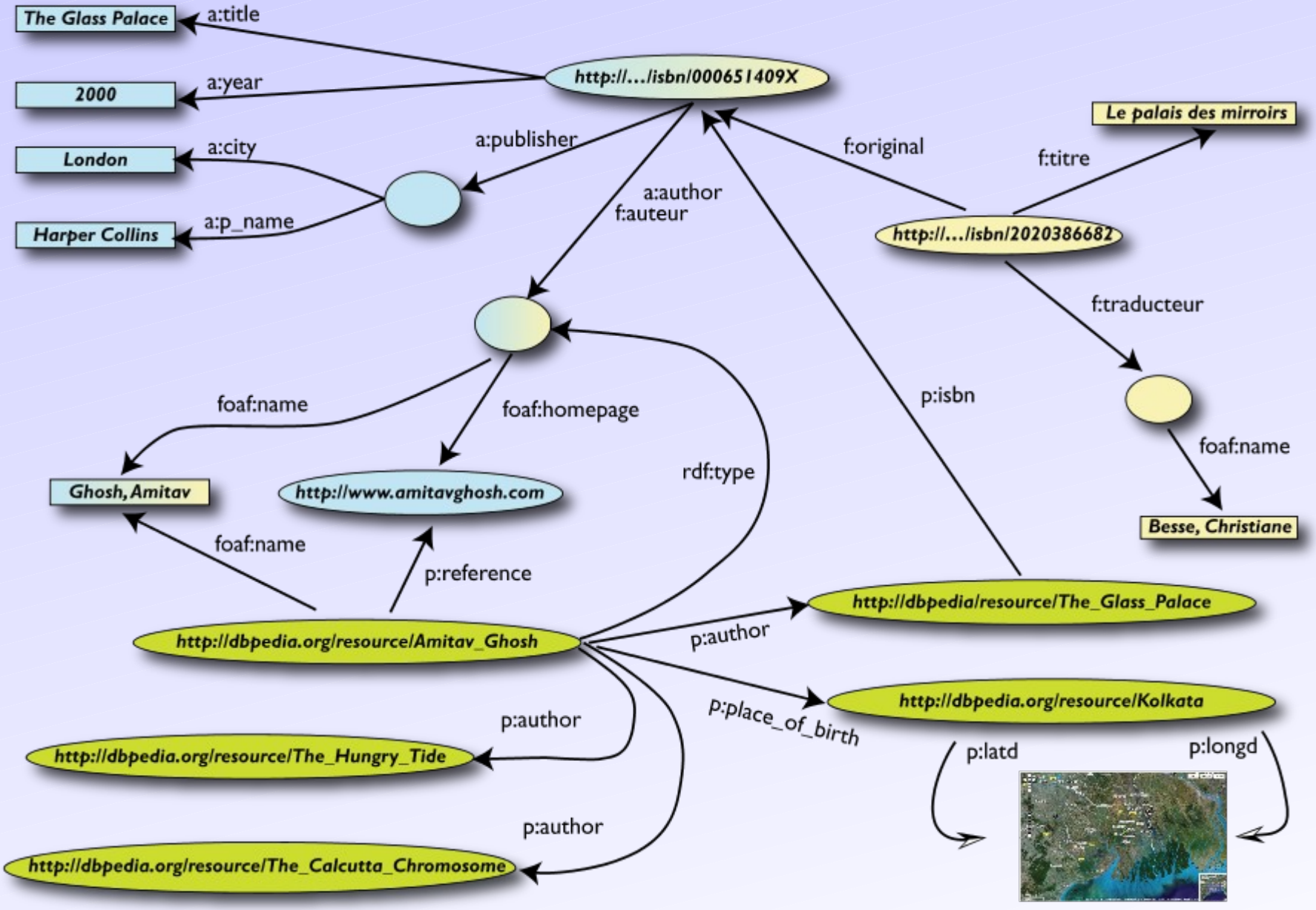
Merge with Wikipedia data



Merge with Wikipedia data



Merge with Wikipedia data



Is that surprising?

- It may look like it but, in fact, it should not be…
- What happened via automatic means is done every day by Web users!
- The difference: a bit of extra rigour so that machines could do this, too

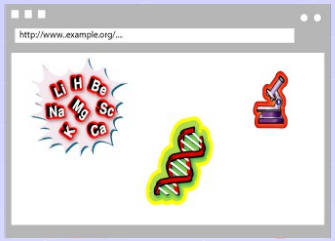
What did we do?

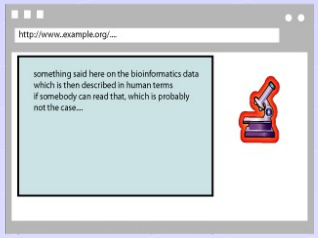
- We combined different datasets that
  - are somewhere on the web
  - are of different formats (mysql, excel sheet, XHTML, etc)
  - have different names for relations
- We could combine the data because some URI-s were identical (the ISBN-s in this case)
- We could add some simple additional information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

It could become even more powerful

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

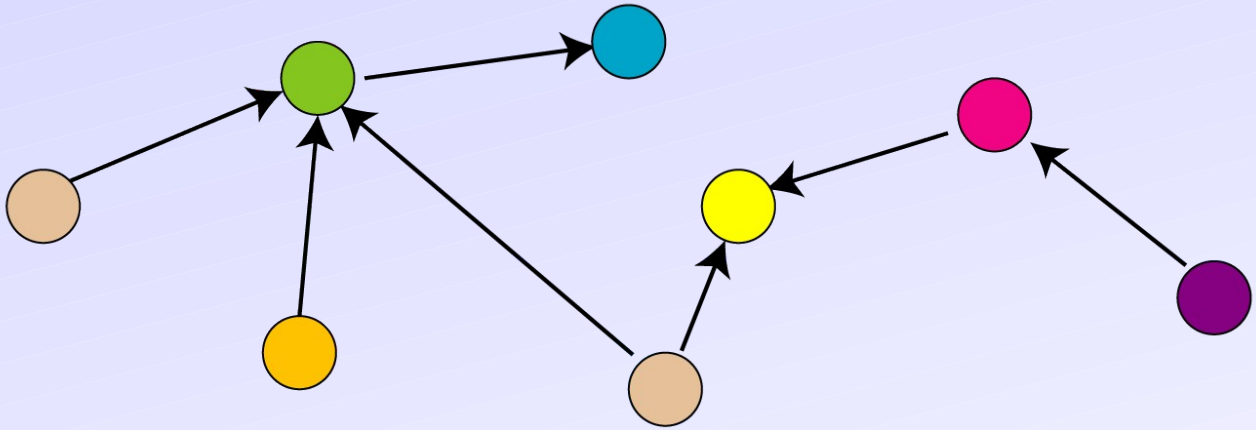
What did we do? (cont)





Applications

Query, Manipulate, etc.



Data represented in abstract format

Map, Expose, etc.

Data in various formats

The Basis: RDF

RDF triples

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

RDF triples (cont.)

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

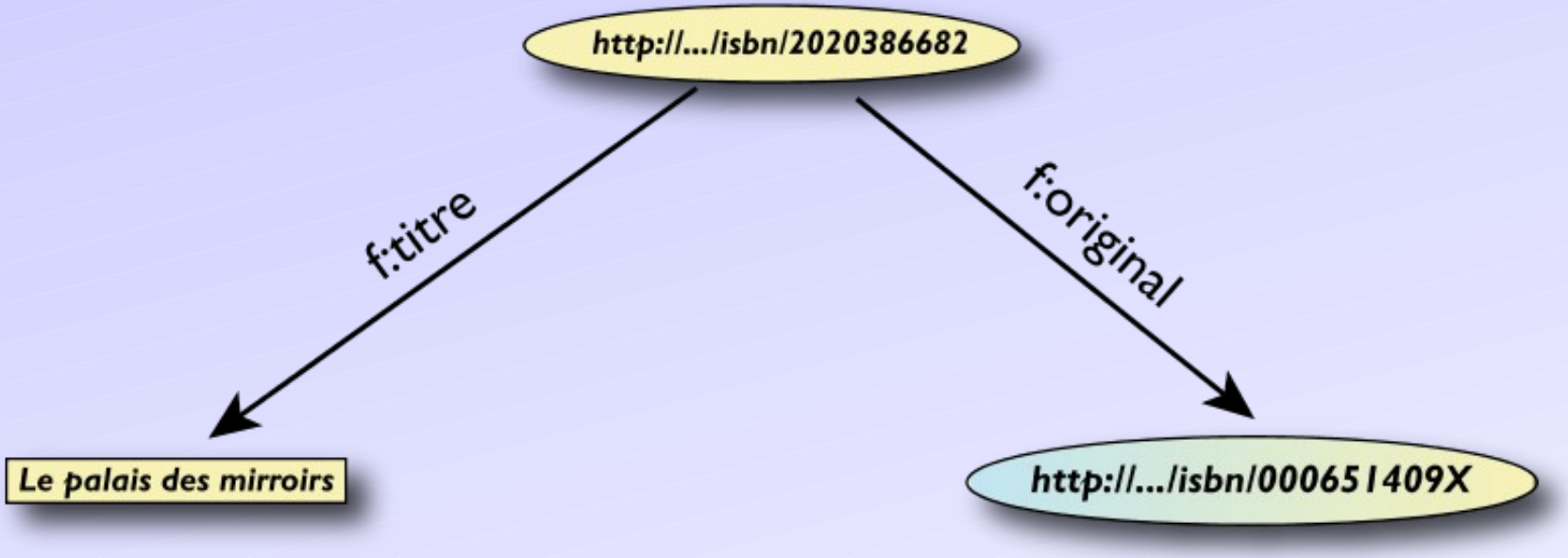
```
(<http://…isbn…6682>, <http://…/original>, <http://…isbn…409X>)
```

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

RDF triples (cont.)

- Resources can use *any* URI, e.g.:
  - `http://www.example.org/file.xml#element(home)`
  - `http://www.example.org/file.html#home`
  - `http://www.example.org/file2.xml#xpath1(//q[@a=b])`
- URI-s can also denote non Web entities:
  - `http://www.ivan-herman.net/me` is me
  - not my home page, not my publication list, but *me*
- RDF triples form a directed, labelled graph

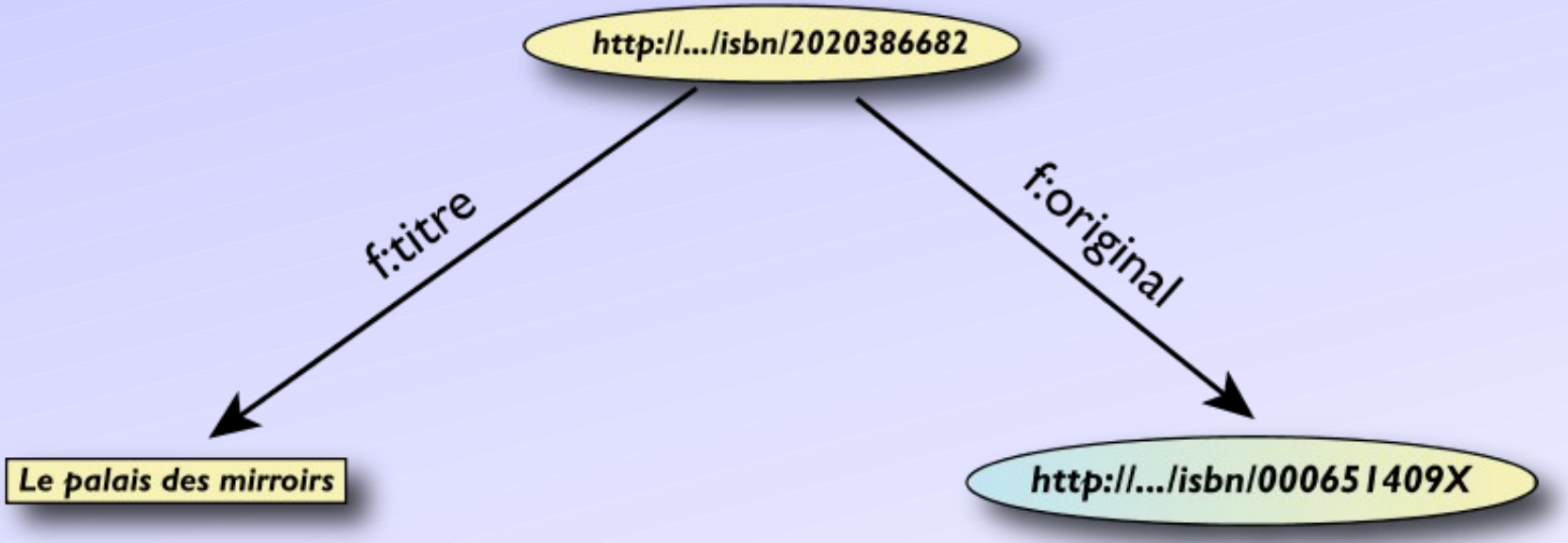
A simple RDF example (in RDF/XML)




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

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

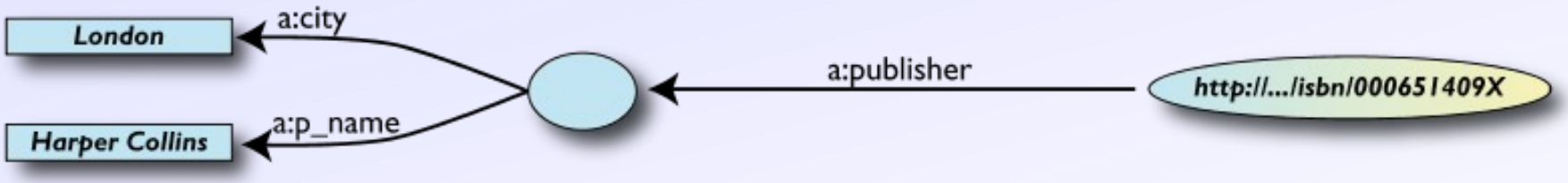
A simple RDF example (in Turtle)




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

"Internal" nodes

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



Internal identifier ("blank nodes")

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

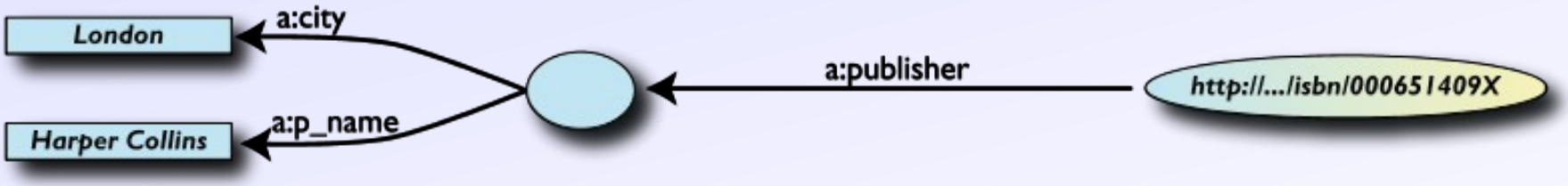
```
<http://…/isbn/2020386682> a:publisher _:A234.
_:A234 a:p_name "HarpersCollins".
```

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

Blank nodes: the system can also do it

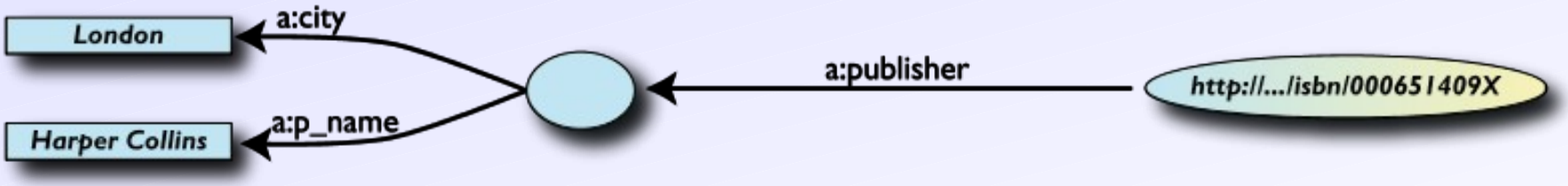
- Let the system create a “nodeID” internally (you do not really care about the name…)

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



Blank nodes: some more remarks

- Blank nodes require attention when merging
  - blanks nodes with identical nodeID-s in *different* graphs are *different*
  - implementations must be careful…
- Many applications prefer not to use blank nodes and define new URI-s “on-the-fly”

RDF in programming practice

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

Jena example

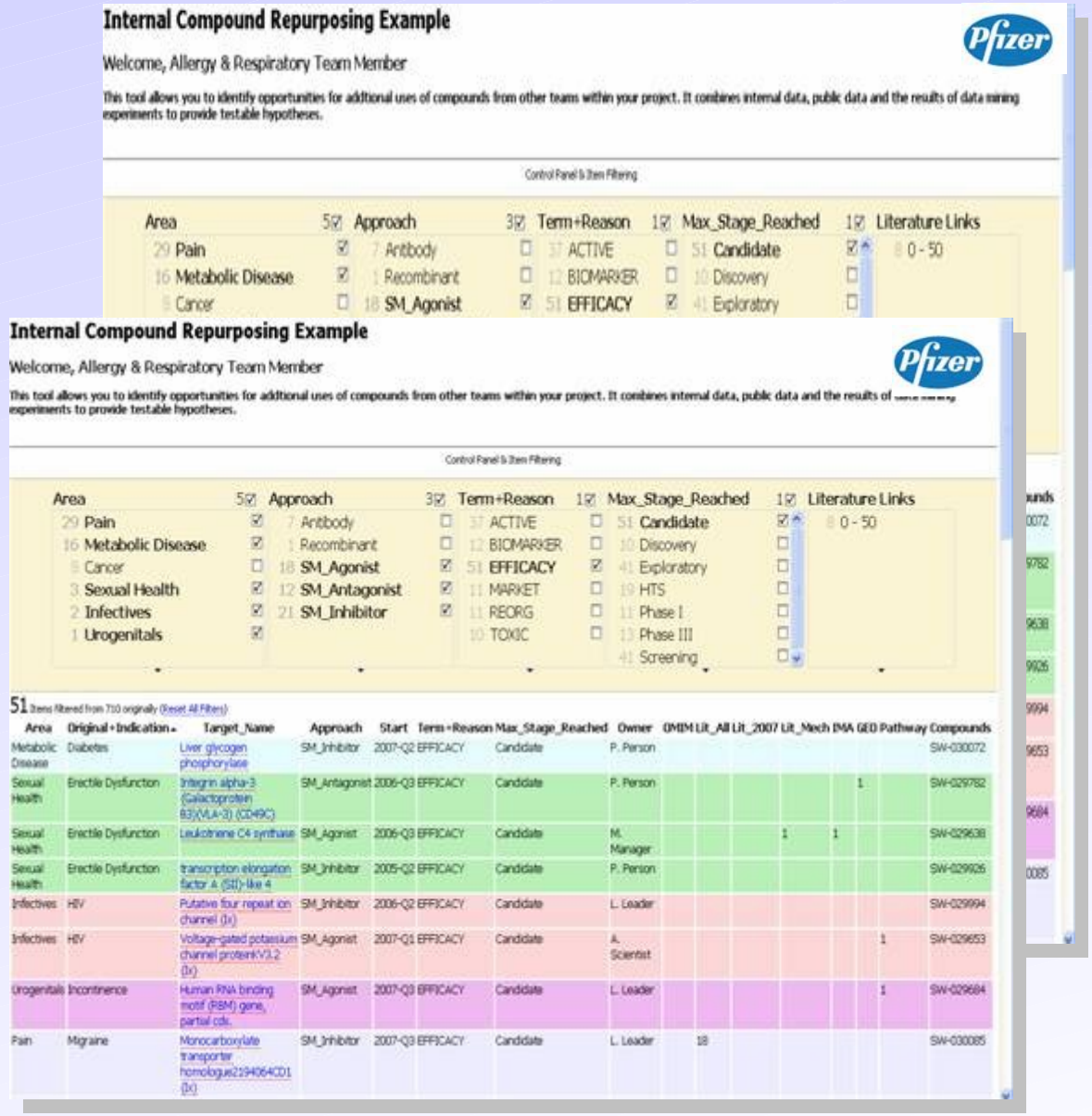
```
// create a model
Model model=new ModelMem();
Resource subject=model.createResource("URI_of_Subject")
// 'in' refers to the input file
model.read(new InputStreamReader(in));
StmtIterator iter=model.listStatements(subject,null,null);
while(iter.hasNext()) {
   st = iter.next();
   p = st.getProperty();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

Merge in practice

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

Example: integrate experimental data

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



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

One level higher up
(RDFS, Datatypes)

Need for RDF schemas

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

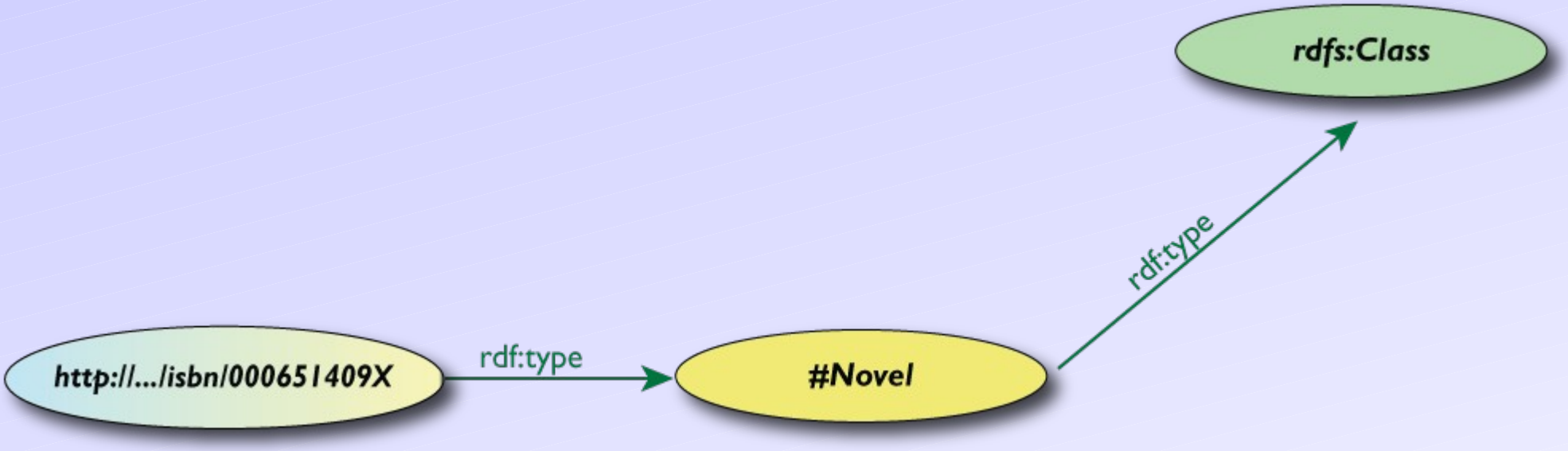
Classes, resources, …

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

Classes, resources, … (cont.)

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

Classes, resources in RDF(S)



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

Schema example in RDF/XML

- The schema part:

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

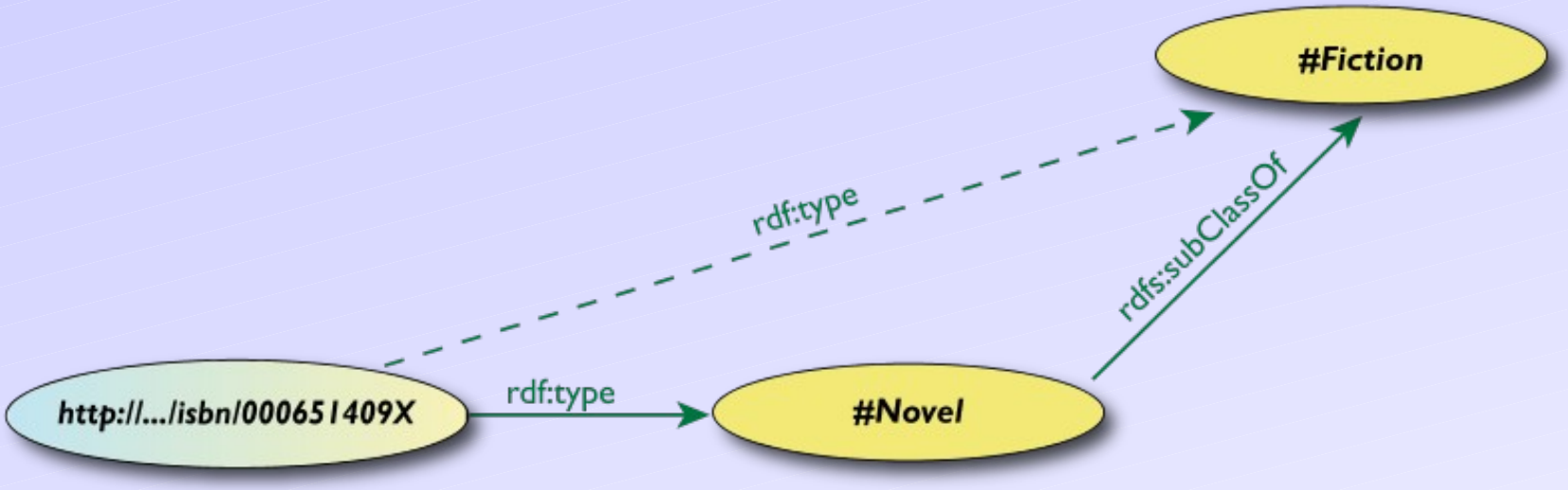
- The RDF data on a specific novel:

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

Further remarks on types

- A resource may belong to several classes
  - rdf:type is just a property…
    - "«The Glass Palace» is a novel, but «The Glass Palace» is also an «inventory item»…"
  - i.e., it is *not* like a datatype!
- The type information may be very important for applications
  - e.g., it may be used for a categorization of possible nodes
  - probably the most frequently used RDF property…
    - (remember the "Person" in our example?)

Inferred properties




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

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

Inference: let us be formal…

- The RDF Semantics document has a list of (33) *entailment rules*:
  - "if such and such triples are in the graph, add this and this"
  - do that recursively until the graph does not change
- The relevant rule for our example:

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

Properties

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

Property specification serialized

- In RDF/XML:

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

- In Turtle:

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

What does this mean?

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

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

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

- then the system can *infer* that:

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

Literals

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

Examples for datatypes

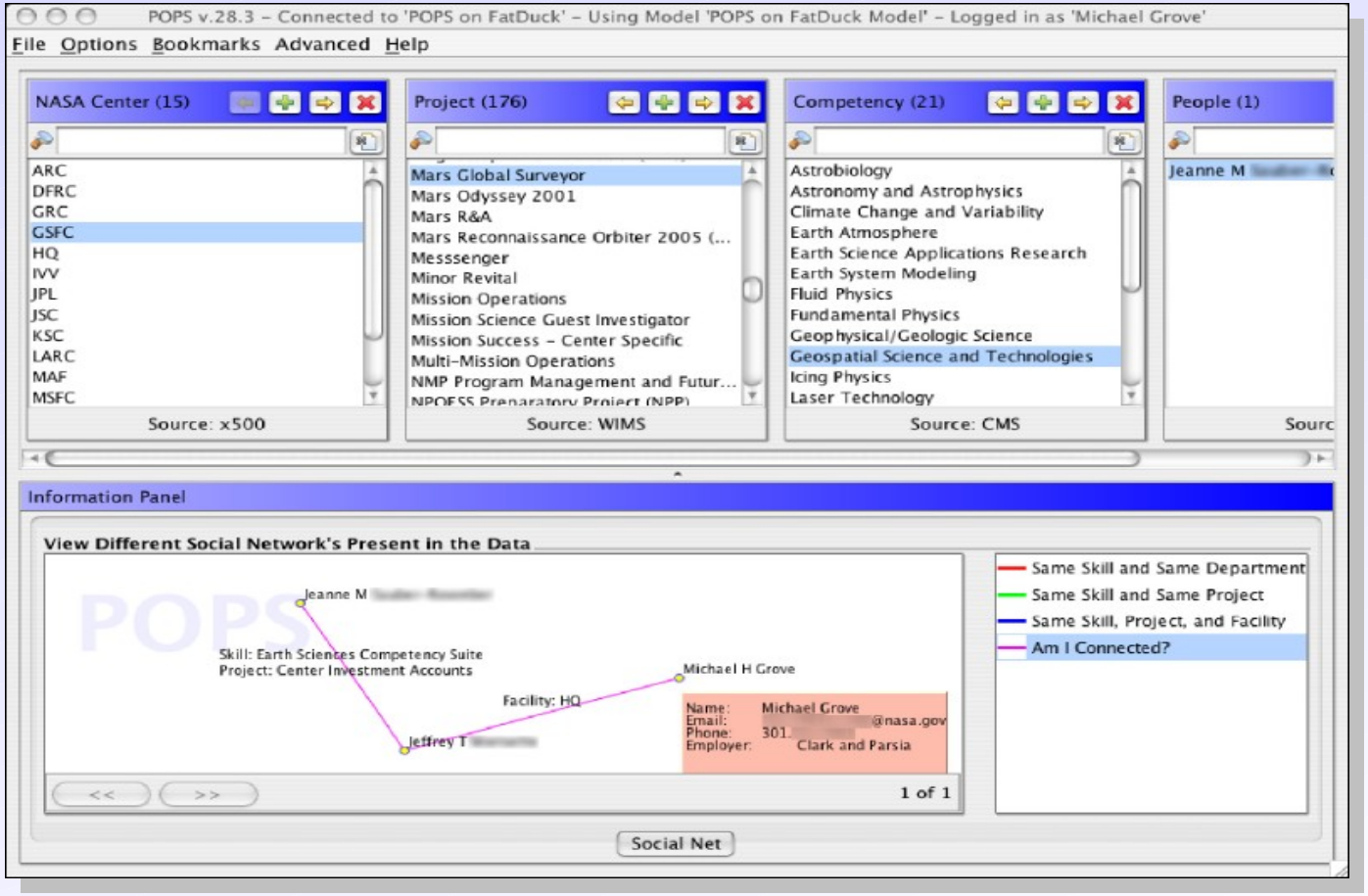
```
<http://…/isbn/000651409X>
      :page_number "543"^^xsd:integer ;
      :publ_date   "2000"^^xsd:gYear ;
      :price       "6.99"^^xsd:float .
```

A bit of RDFS can take you far…

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

Example: find the right experts at NASA

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



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

How to get RDF Data?
(Microformats, GRDDL, RDFa)

Simple approach

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

RDF with XHTML

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

Extract RDF

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

Formalizing the scraper approach: GRDDL

- GRDDL formalizes the scraper approach. For example:

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

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

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

GRDDL

- The transformation itself has to be provided for each set of conventions
- A more general syntax is defined for XML formats in general (e.g., via the namespace document)
  - a method to get data in other formats to RDF (e.g., XBRL)

Example for "structure": microformats

- *Not* a Semantic Web specification, originally
  - there is a separate microformat community
- Approach: re-use (X)HTML attributes and elements to add "meta" information
  - typically **`@abbr`**, **`@class`**, **`@title`**, …
  - different community agreements for different applications

RDFa

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

RDFa example

- For example:

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

- yields, through an RDFa processor:

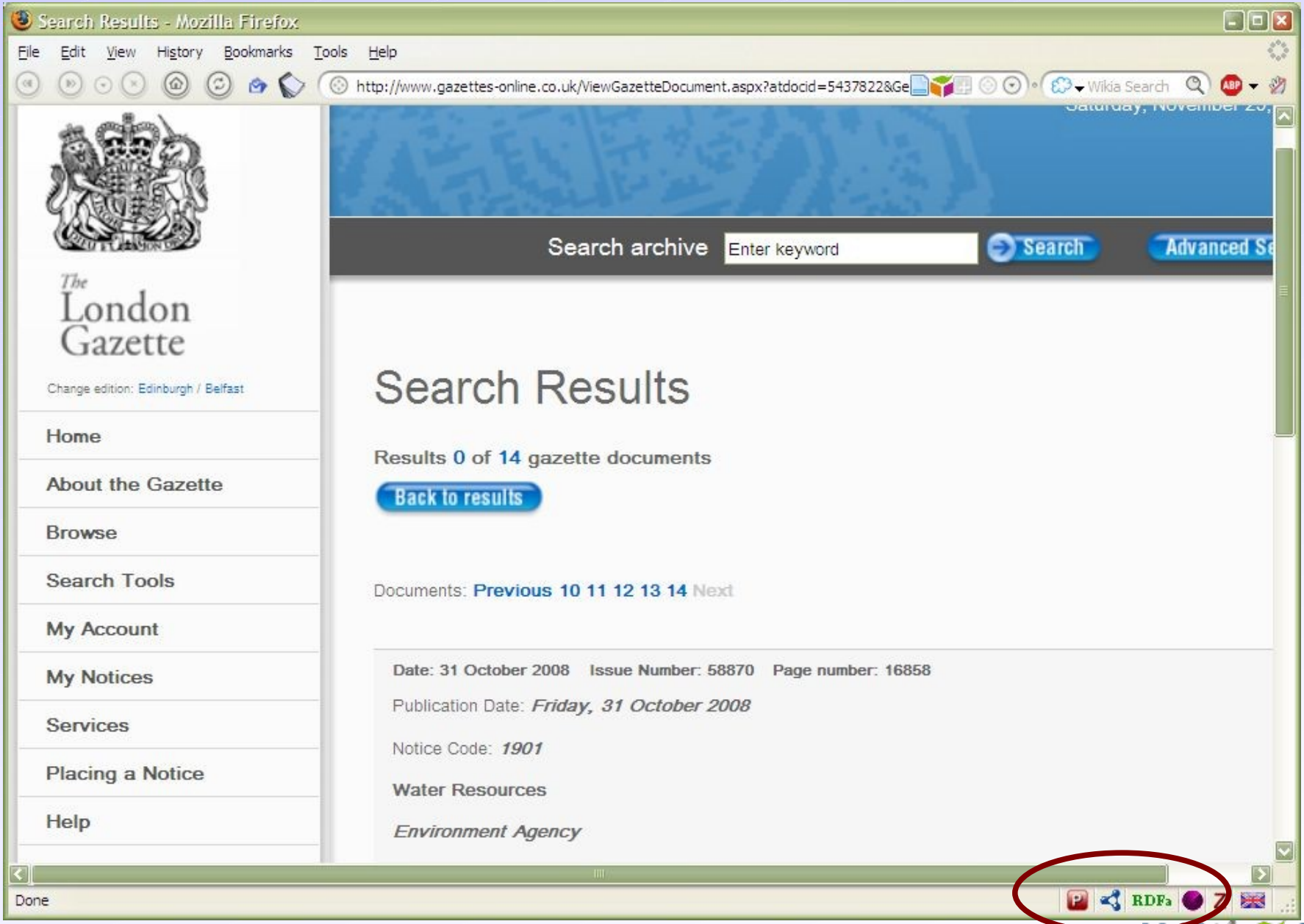
```
<http://uri.to.newsitem>
  dc:date                "March 23, 2004";
  dc:title               "Rollers hit casino for £1.3m;
  dc:creator             "Steve Bird";
  dcmtype:MovingImage <http://www.a.b.c/d.avi>.
```

Example: Yahoo's SearchMonkey

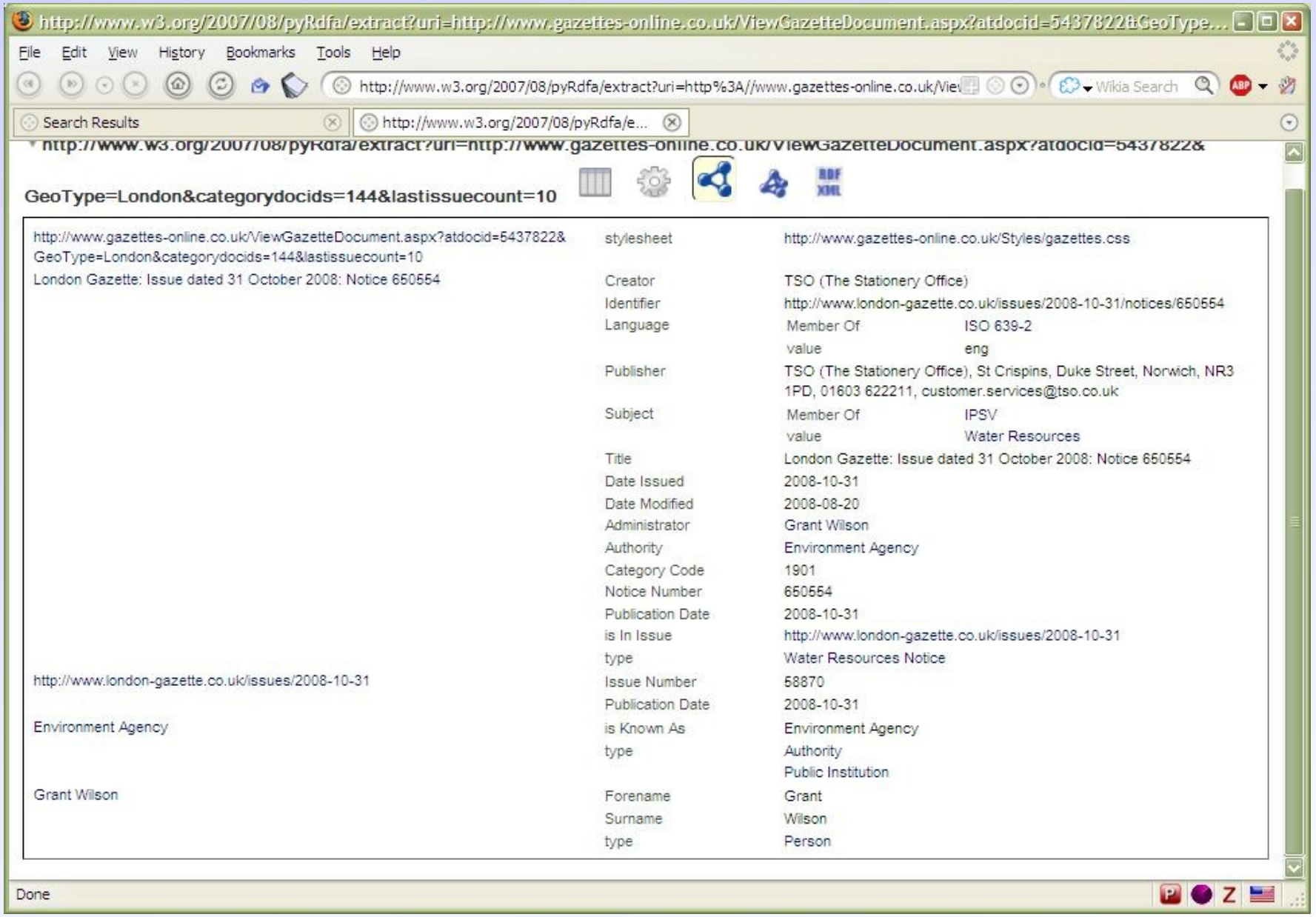
- Search based results may be customized via small applications
- Metadata in pages (in RDFa, microformats etc) are reused

Courtesy of Peter Mika, Yahoo! Research, (SWEO Case Study)

Example: RDFa data by the London Gazette



Example: RDFa data by the London Gazette



Bridge to relational databases

- Data on the Web are mostly stored in databases
- "Bridges" are being defined:
  - a layer between RDF and the relational data
    - RDB tables are "mapped" to RDF graphs, possibly on the fly
    - different mapping approaches are being used
  - a number RDB systems offer this facility already (eg, Oracle, OpenLink, …)
- A survey on mapping techniques has been published at W3C
- W3C plans to engage in a standardization work in this area

Linking Data

Linking Open Data Project

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

Example data source: DBpedia

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

Freie Universität Berlin

UNIVERSITÄT LEIPZIG

OPENLINK SOFTWARE

Extracting Wikipedia structured data

Amsterdam

The Keizersgracht at dusk

Location of Amsterdam

Coordinates: 52°22'23"N 4°53'32"E

Country	Netherlands
Province	North Holland
Government	
- Type	Municipality
- Mayor	Job Cohen[1] (PvdA)
- Aldermen	Lodewijk Asscher Carolien Gehrels Tjeerd Herrema Maarten van Poelgeest Marijke Vos
- Secretary	Erik Gerritsen
Area [2][3]	
- 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 [4]	2 m (7 ft)
Population (1 October 2008)[5][6]	
- City	755,269
- Density	4,459/km² (11,548.8/sq mi)
- Urban	1,364,422
- Metro	2,158,372
- Demonym	Amsterdammer
Time zone	CET (UTC+1)
- Summer (DST)	CEST (UTC+2)
Postcodes	1011 – 1109
Area code(s)	020

Website: www.amsterdam.nl

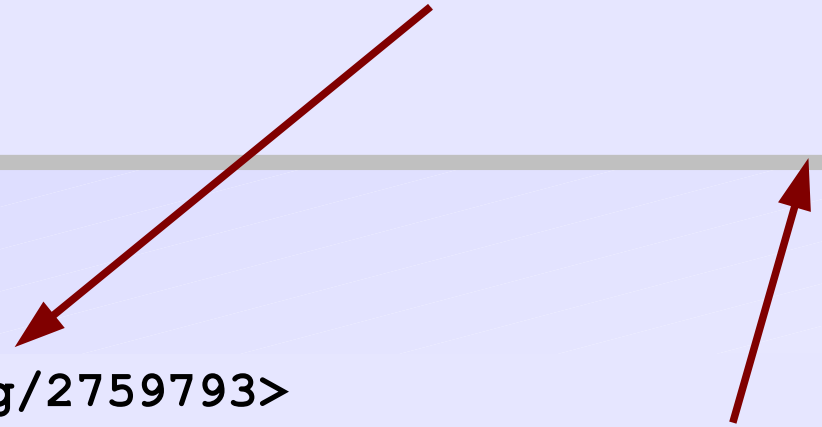
```
@prefix dbpedia <http://dbpedia.org/resource/>.
@prefix dbterm  <http://dbpedia.org/property/>.

dbpedia:Amsterdam
   dbterm:officialName "Amsterdam" ;
   dbterm:longd "4" ;
   dbterm:longm "53" ;
   dbterm:longs "32" ;
   ...
   dbterm:leaderTitle "Mayor" ;
   dbterm:leaderName dbpedia:Job_Cohen ;
   ...
   dbterm:areaTotalKm "219" ;
   ...
dbpedia:ABN_AMRO
   dbterm:location dbpedia:Amsterdam ;
   ...
```

W3C Semantic Web

Automatic links among open datasets

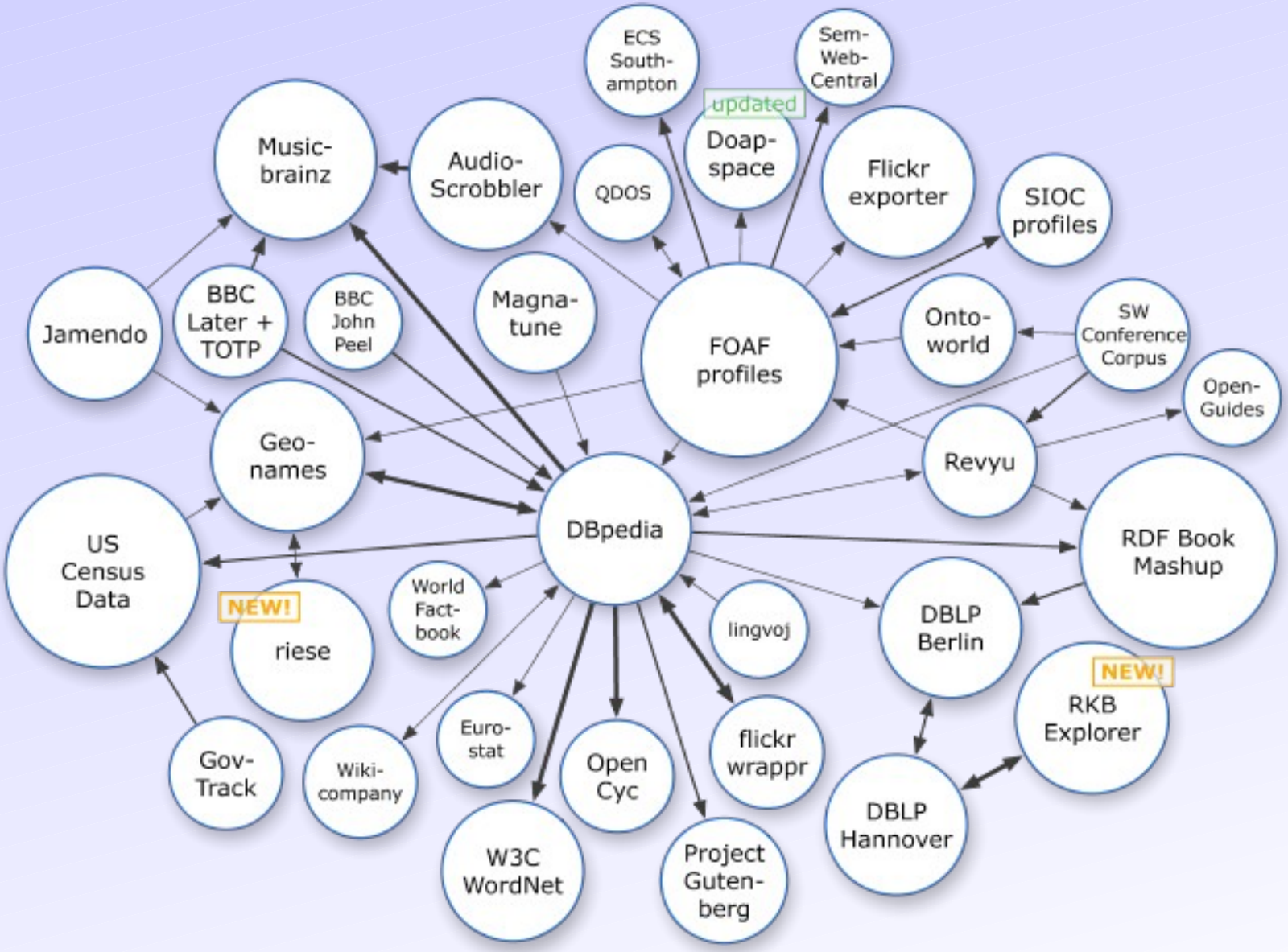
```
<http://dbpedia.org/resource/Amsterdam>
  owl:sameAs <http://rdf.freebase.com/ns/...> ;
  owl:sameAs <http://sws.geonames.org/2759793> ;
  ...
```



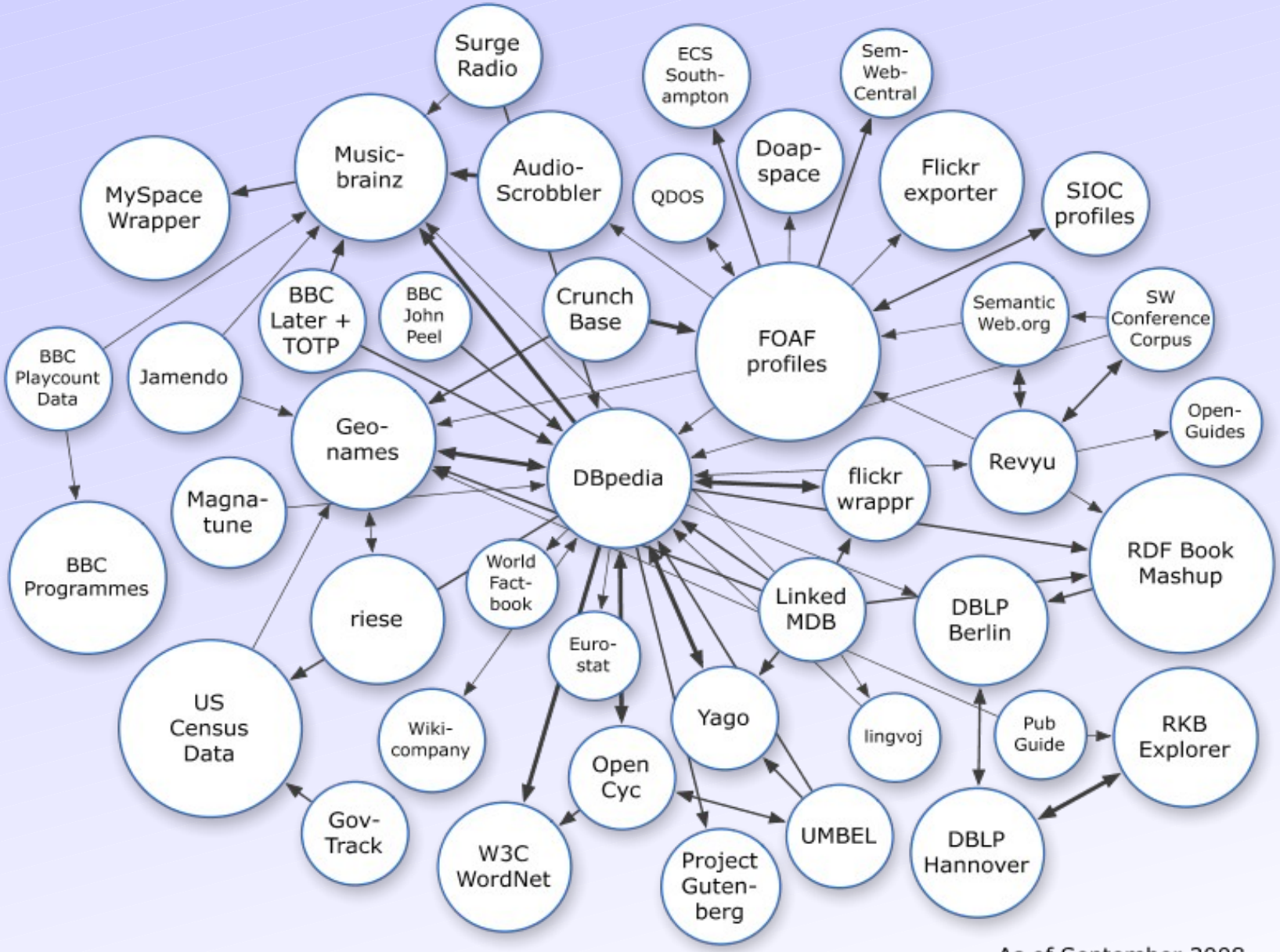
```
<http://sws.geonames.org/2759793>
  owl:sameAs <http://dbpedia.org/resource/Amsterdam>
  wgs84_pos:lat "52.3666667" ;
  wgs84_pos:long "4.8833333" ;
  geo:inCountry <http://www.geonames.org/countries/#NL> ;
 ...
```

Processors can switch automatically from one to the other…

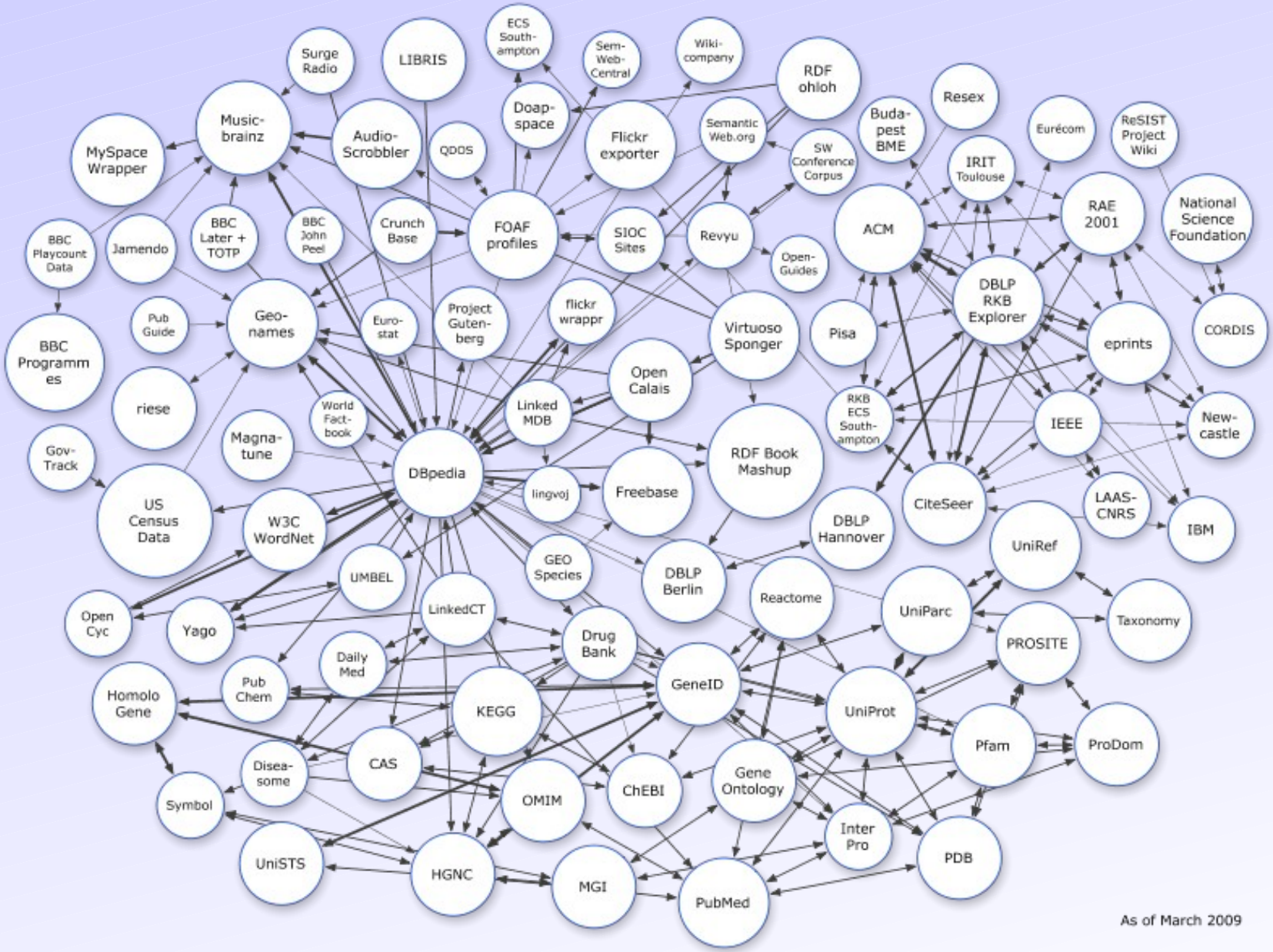
The LOD "cloud", March 2008



The LOD "cloud", September 2008



The LOD "cloud", March 2009



W3C Semantic Web

Example: mapping application on an iPhone

Courtesy of Chris Bizer and Christian Becker, Freie Universität, Berlin

W3C Semantic Web

Example: mapping application on an iPhone

Courtesy of Chris Bizer and Christian Becker, Freie Universität, Berlin

Query RDF Data
(SPARQL)

RDF data access

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

Querying RDF graphs

- Remember the Jena idiom:

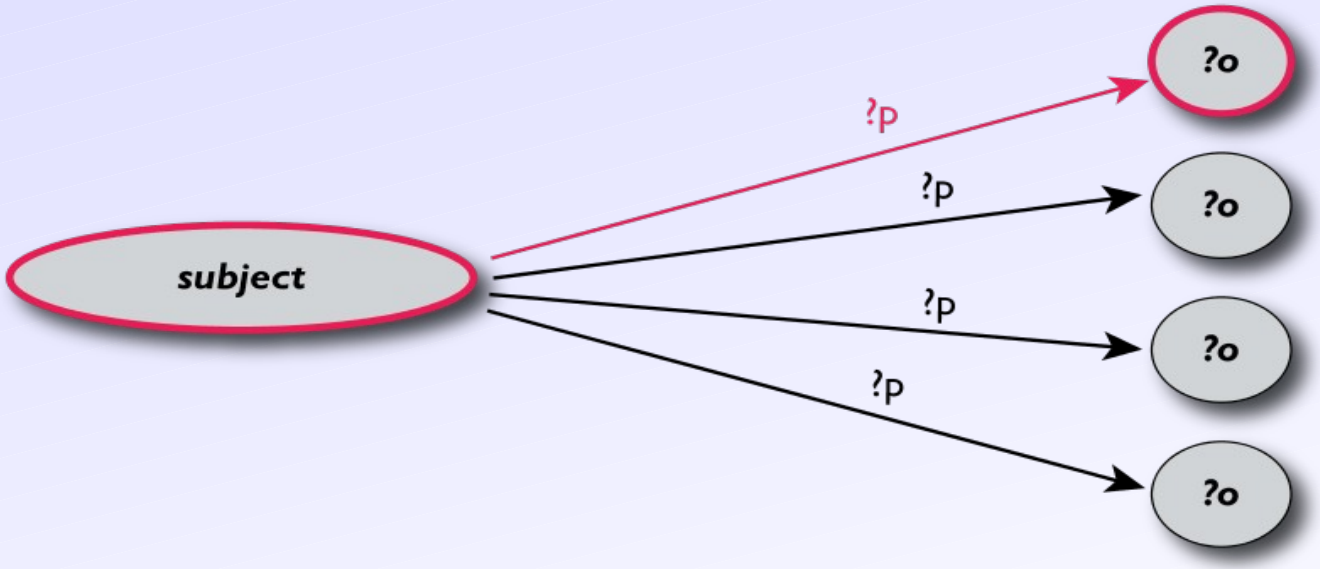
```
StmtIterator iter=model.listStatements(subject,null,null);
while(iter.hasNext()) {
    st = iter.next();
    p = st.getProperty(); o = st.getObject();
    do_something(p,o);
```

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

Analyse the Jena example

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

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



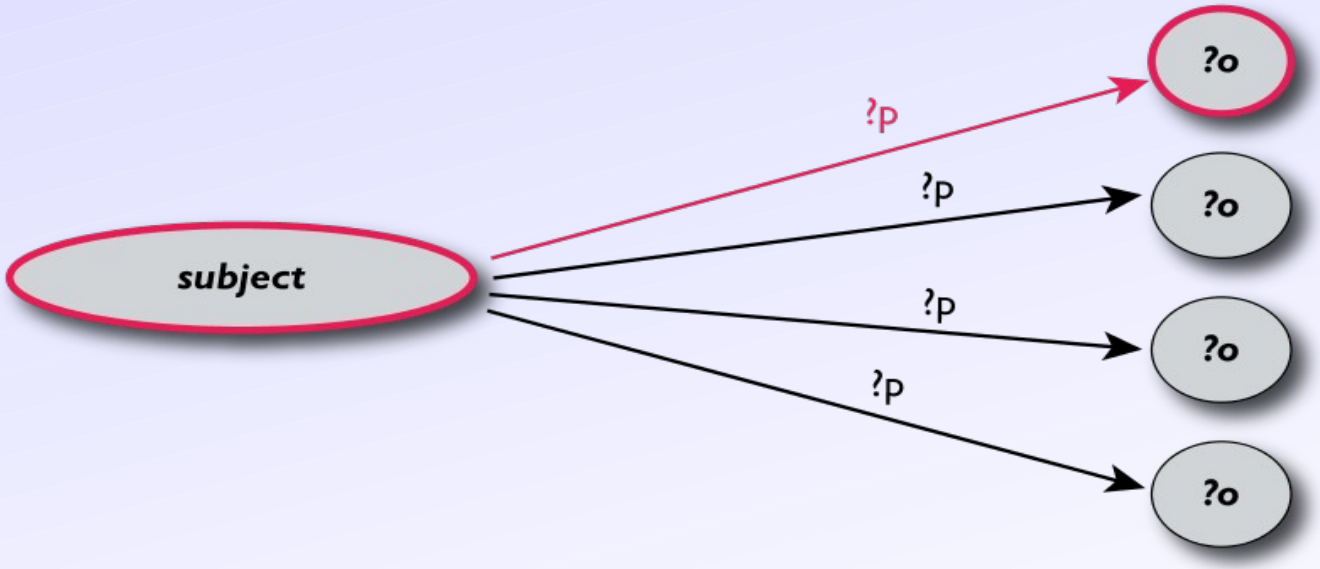
General: graph patterns

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

Our Jena example in SPARQL

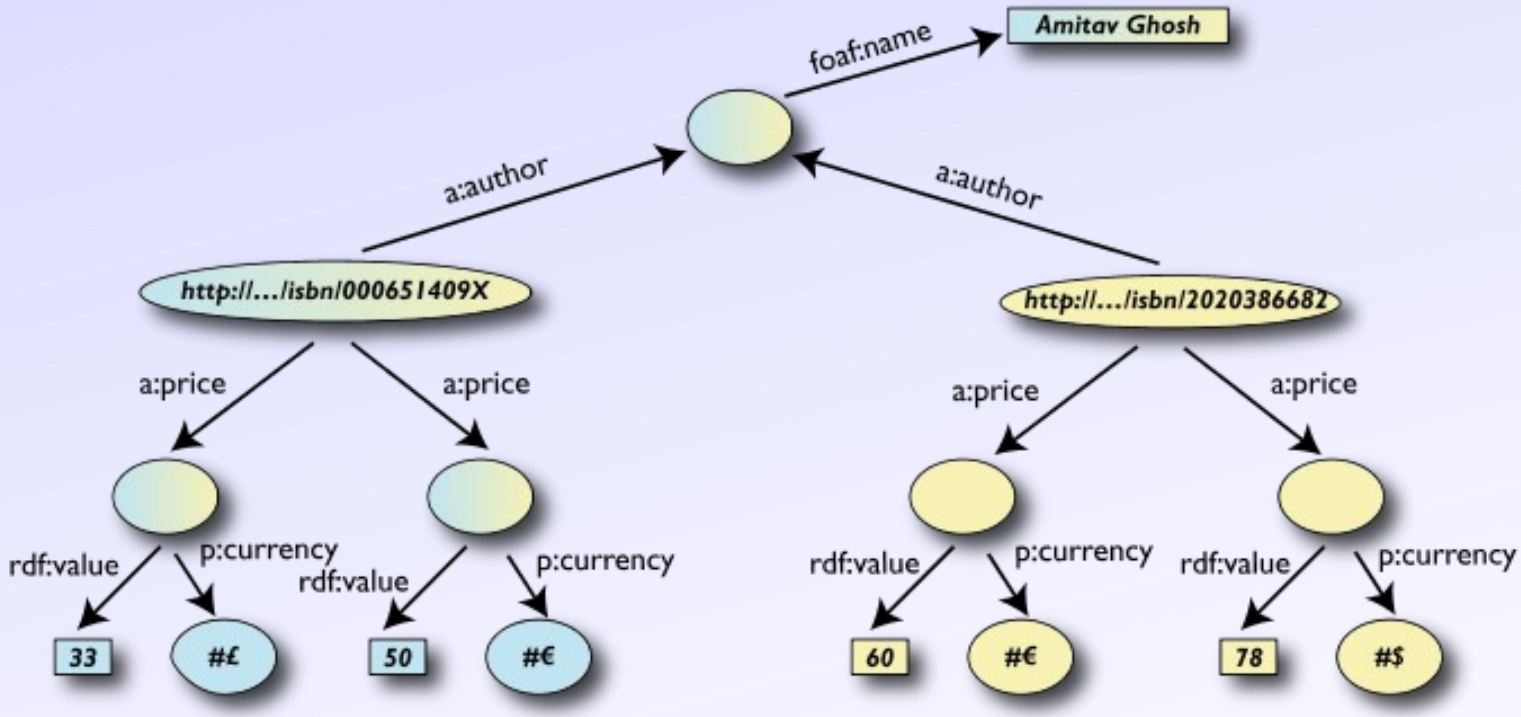
```
SELECT ?p ?o
WHERE {subject ?p ?o}
```

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



Simple SPARQL example

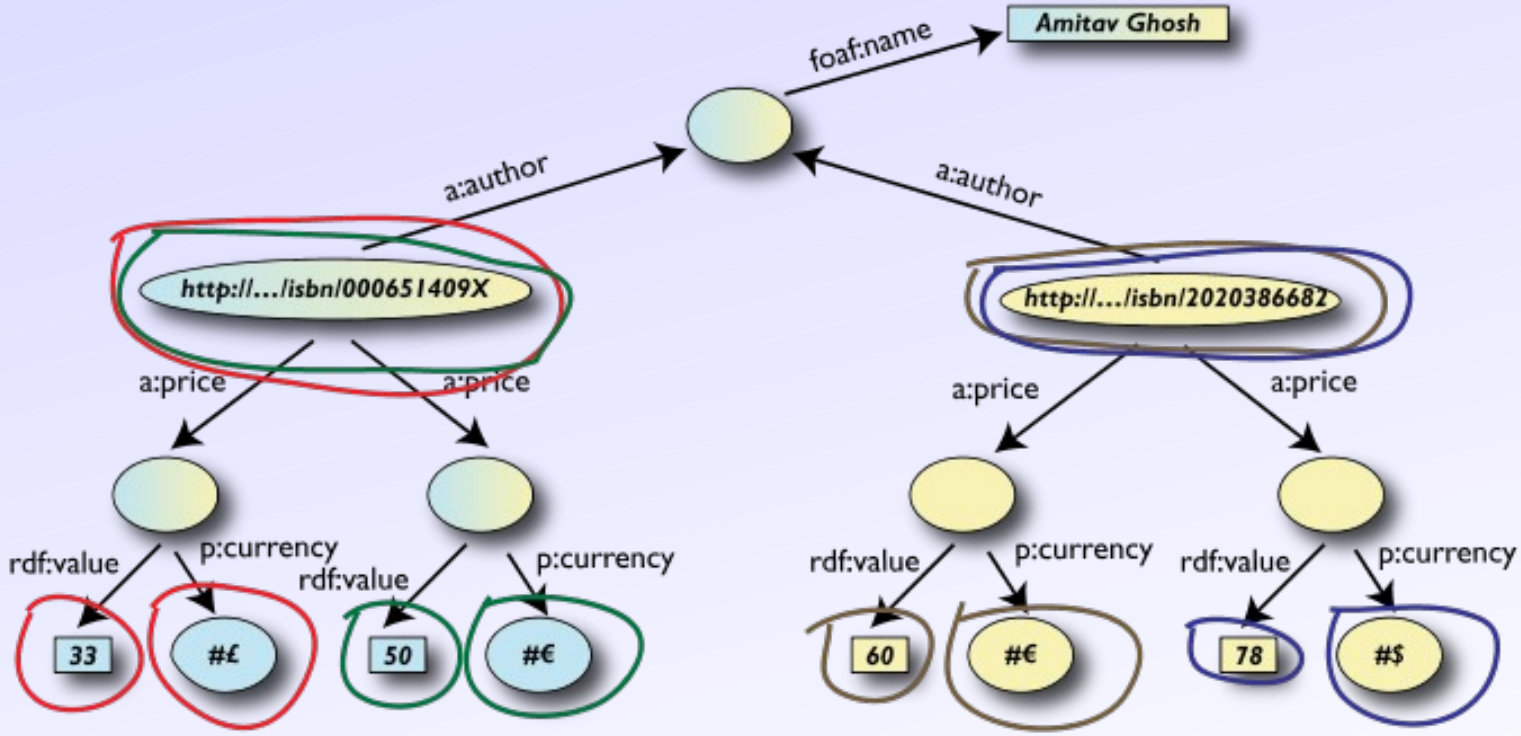
```
SELECT ?isbn ?price ?currency # note: not ?x!
WHERE {?isbn a:price ?x. ?x rdf:value ?price. ?x p:currency ?currency.}
```



Simple SPARQL example

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

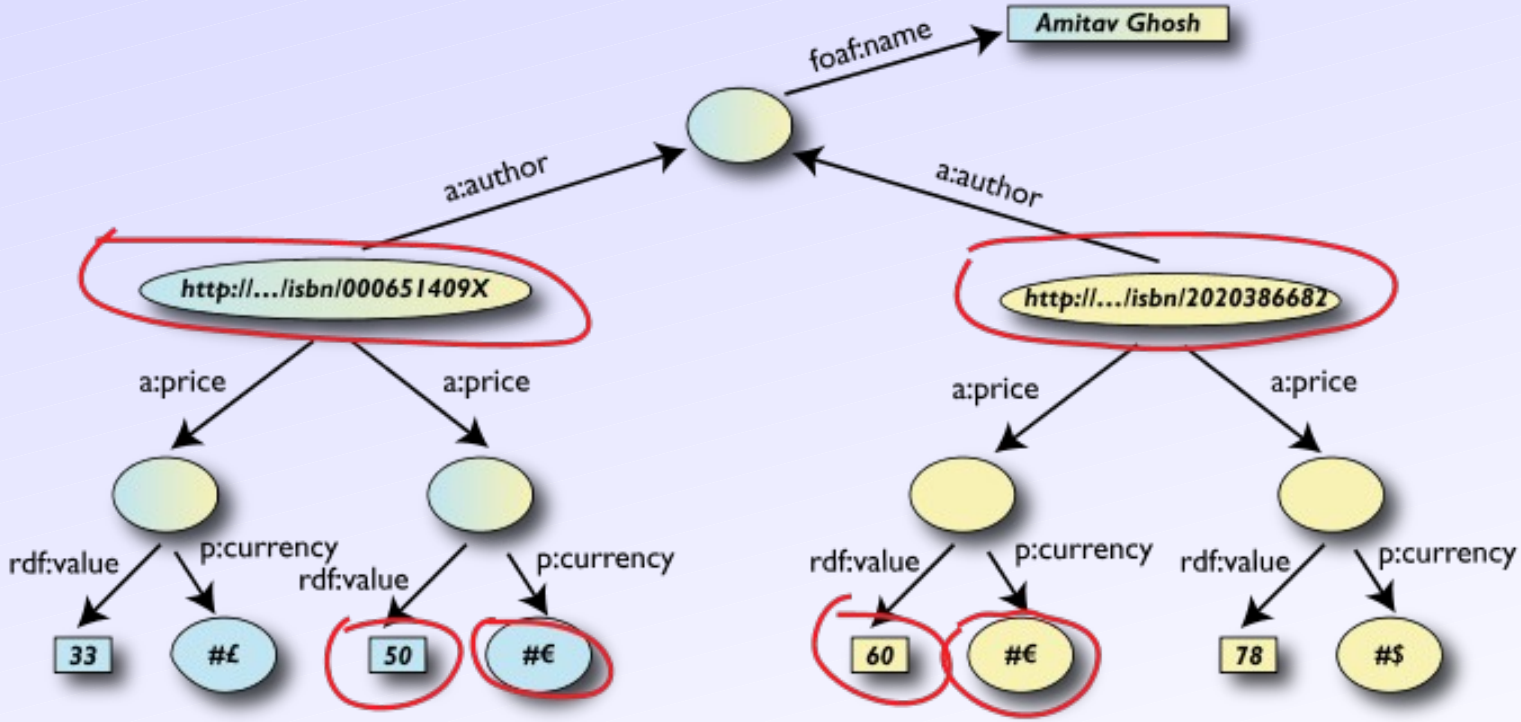
- Returns:
  [[<..49X>,33,£], [<..49X>,50,€], [<..6682>,60,€], [<..6682>,78,$]]



Pattern constraints

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

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



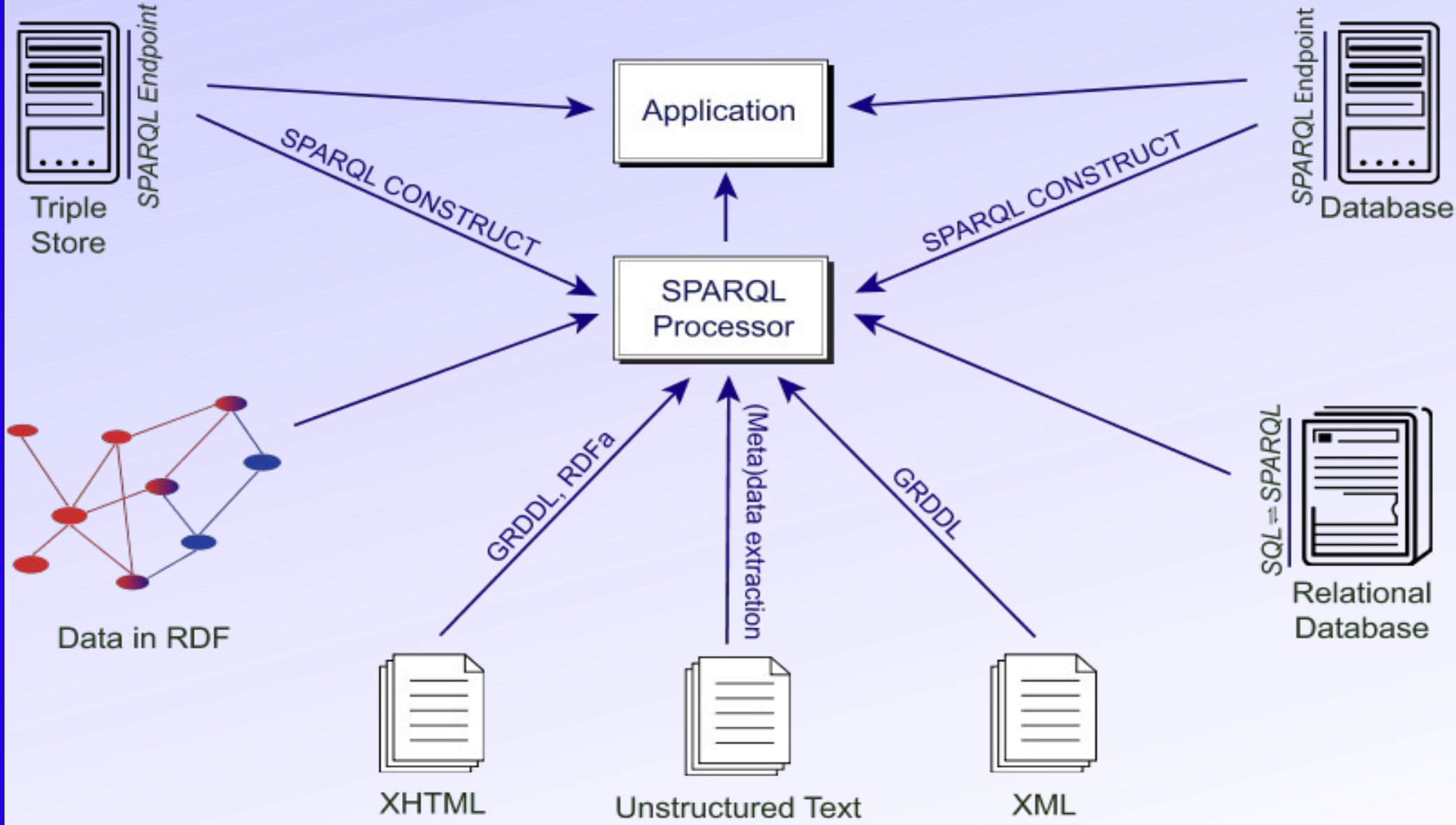
Other SPARQL features

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

SPARQL usage in practice

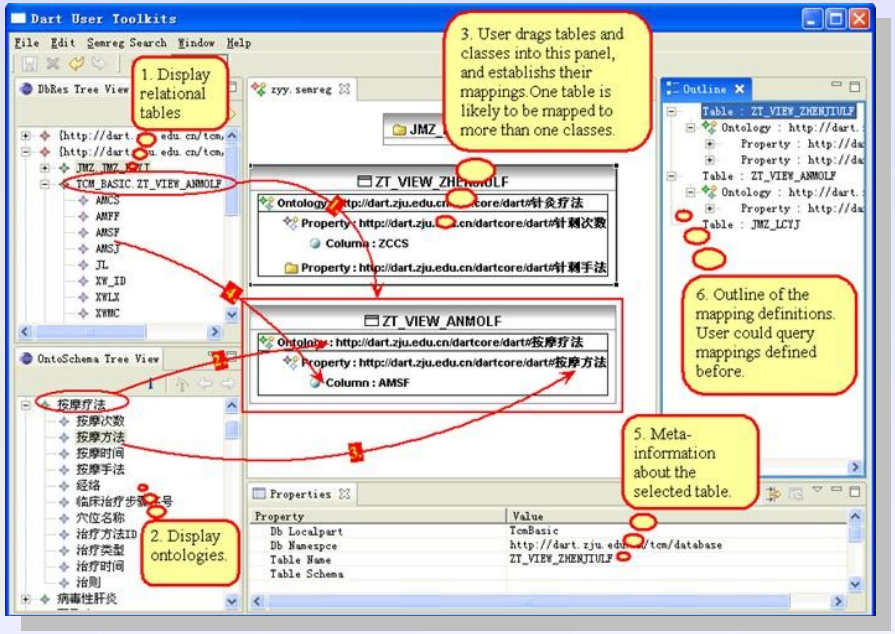
- SPARQL is usually used over the network
  - separate documents define the protocol and the result format
  - SPARQL Protocol for RDF with HTTP and SOAP bindings
  - SPARQL results in XML or JSON formats
- Big datasets usually offer “SPARQL endpoints” using this protocol
  - typical example: SPARQL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

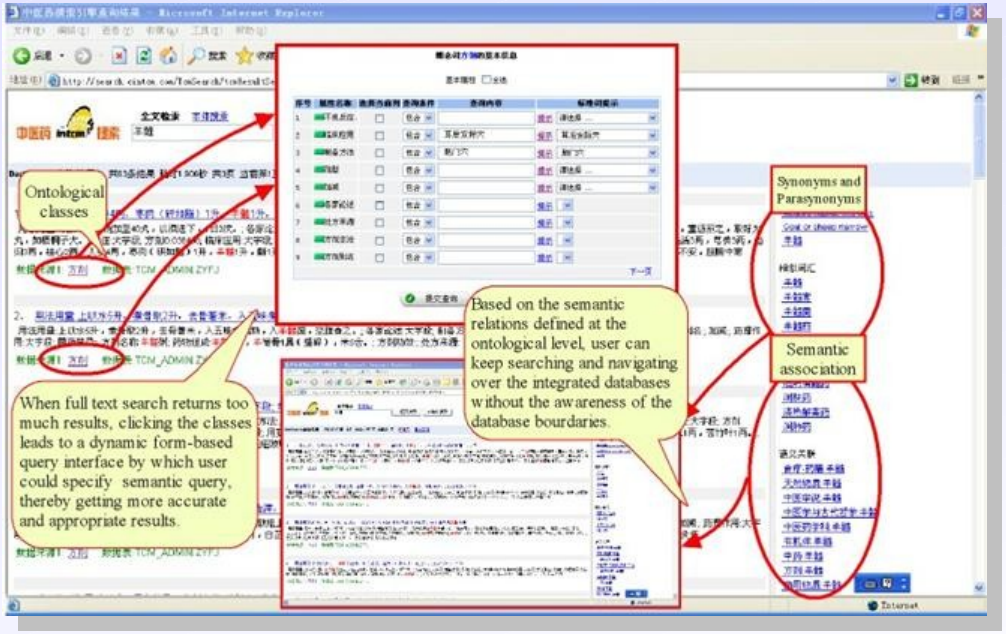
SPARQL as a unifying point



Example: integrate Chinese medical data

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





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

Ontologies
(OWL)

Ontologies

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

Ontologies (cont.)

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

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

- RDFS can be considered as a simple ontology language
- Languages should be a compromise between
  - rich semantics for meaningful applications
  - feasibility, implementability

Web Ontology Language = OWL

- OWL is an extra layer, a bit like RDF Schemas
  - own namespace, own terms
  - it relies on RDF Schemas
- It is a separate recommendation
  - actually… there is a 2004 version of OWL (“OWL 1”)
  - and there is an update (“OWL 2”) that should be finalized in 2009
  - you will surely hear about it at the conference…

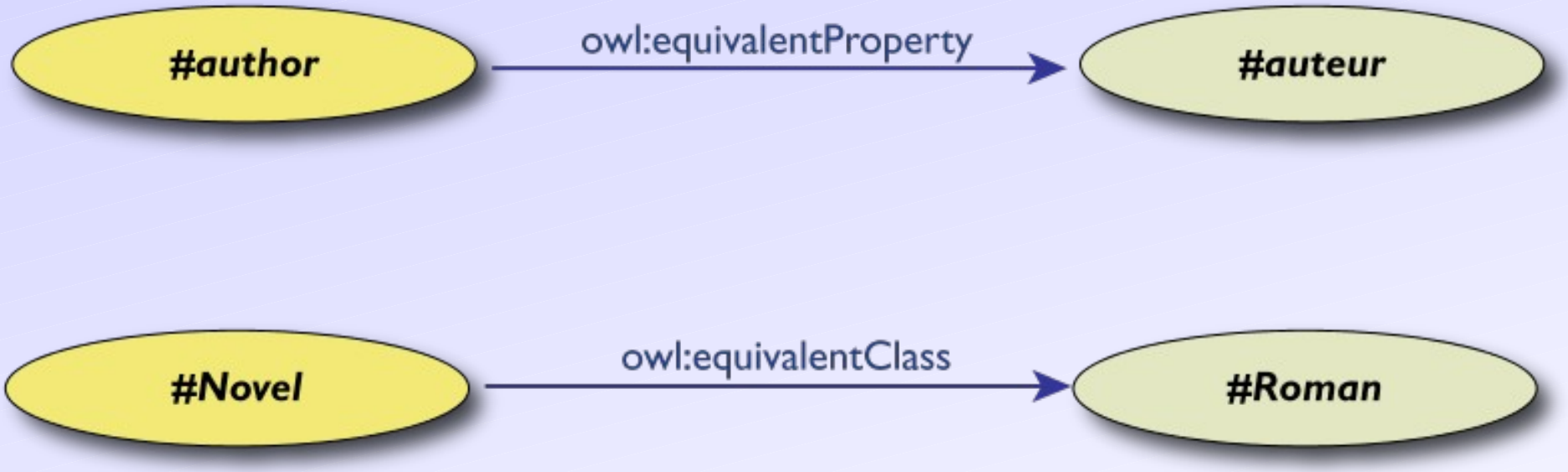
OWL is complex…

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

Term equivalences

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

Connecting to French…



Typical usage of owl:sameAs

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

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

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

Property characterization

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

What this means is…

- If the following holds in our triples:

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

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

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

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

Classes in OWL

- In RDFS, you can subclass existing classes… that's all
- In OWL, you can *construct* classes from existing ones:
  - enumerate its content
  - through intersection, union, complement
  - Etc

Classes in OWL (cont)

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

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

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

Classes contents can be enumerated

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

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

Union of classes can be defined

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

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

For example…

If:

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

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

then the following holds, too:

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

It can be a bit more complicated…

If:

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

fr:Roman owl:equivalentClass :Novel .

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

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

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

What we have so far…

- The OWL features listed so far are already fairly powerful
- E.g., various databases can be linked via **`owl:sameAs`**, functional or inverse functional properties, etc.
- Many inferred 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
  - a major issue: the way classes (i.e., “concepts”) are defined
- OWL includes those extra features but… the inference engines become (much) more complex 

Property value restrictions

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

Restrictions formally

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

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

Possible usage…

If:

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

:price rdf:type :Listed_Price .

:price p:currency <something> .
```

then the following holds:

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

Other restrictions

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

But: OWL is hard!

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

OWL "species"

- OWL species comes to the fore:
  - restricting *which* terms can be used and *under what circumstances (restrictions)*
  - if one abides to those restrictions, then simpler inference engines can be used
- They reflect compromises: expressibility vs. implementability

OWL Full

- No constraints on any of the constructs
  - **`owl:Class`** is just syntactic sugar for **`rdfs:Class`**
  - **`owl:Thing`** is equivalent to **`rdfs:Resource`**
  - this means that:
    - Class can also be an individual, a URI can denote a property as well as a Class
      - e.g., it is possible to talk about class of classes, apply properties on them
      - etc
  - etc.
- Extension of RDFS in all respects
- But: no system may exist that infers everything one might expect

OWL Full usage

- Nevertheless OWL Full is essential
  - it gives a generic framework to *express* many things
  - 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
    - thereby ensuring a manageable inference procedure

OWL DL

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

Examples for restrictions

- The following is not “legal” OWL DL:

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

<r> rdf:type <q>.          # error: q cannot be used for a class, too

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

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

OWL DL usage

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

OWL 2 defines further species a.k.a. "profiles"

- Further restrictions on how terms can be used and what inferences can be expected
  - Classification and instance queries in polynomial time: *OWL-EL*
  - Implementable on top of conventional relational database engines: *OWL-QL*
  - Implementable on top of traditional rule engines: *OWL-RL*

Ontology development

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

Must *I use large ontologies?*

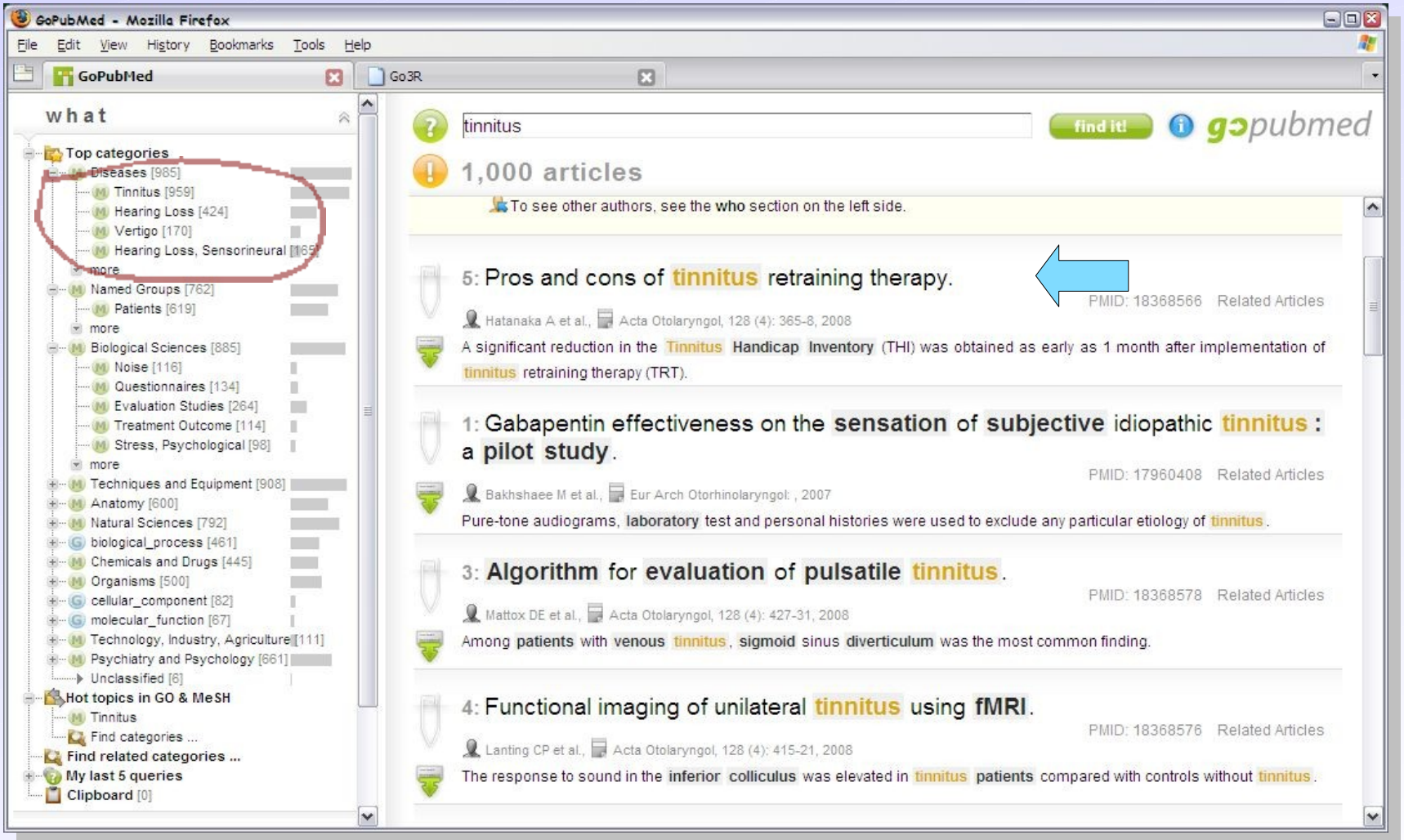
- NO!!!
- Many applications are possible with RDFS and a just a little bit of OWL
  - a few terms, whose meaning is defined in OWL, and that application can handle directly
  - OWL RL is a step to create such a generic OWL level
- Big ontologies can be expensive (both in time and money); use them only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

Ontologies examples

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

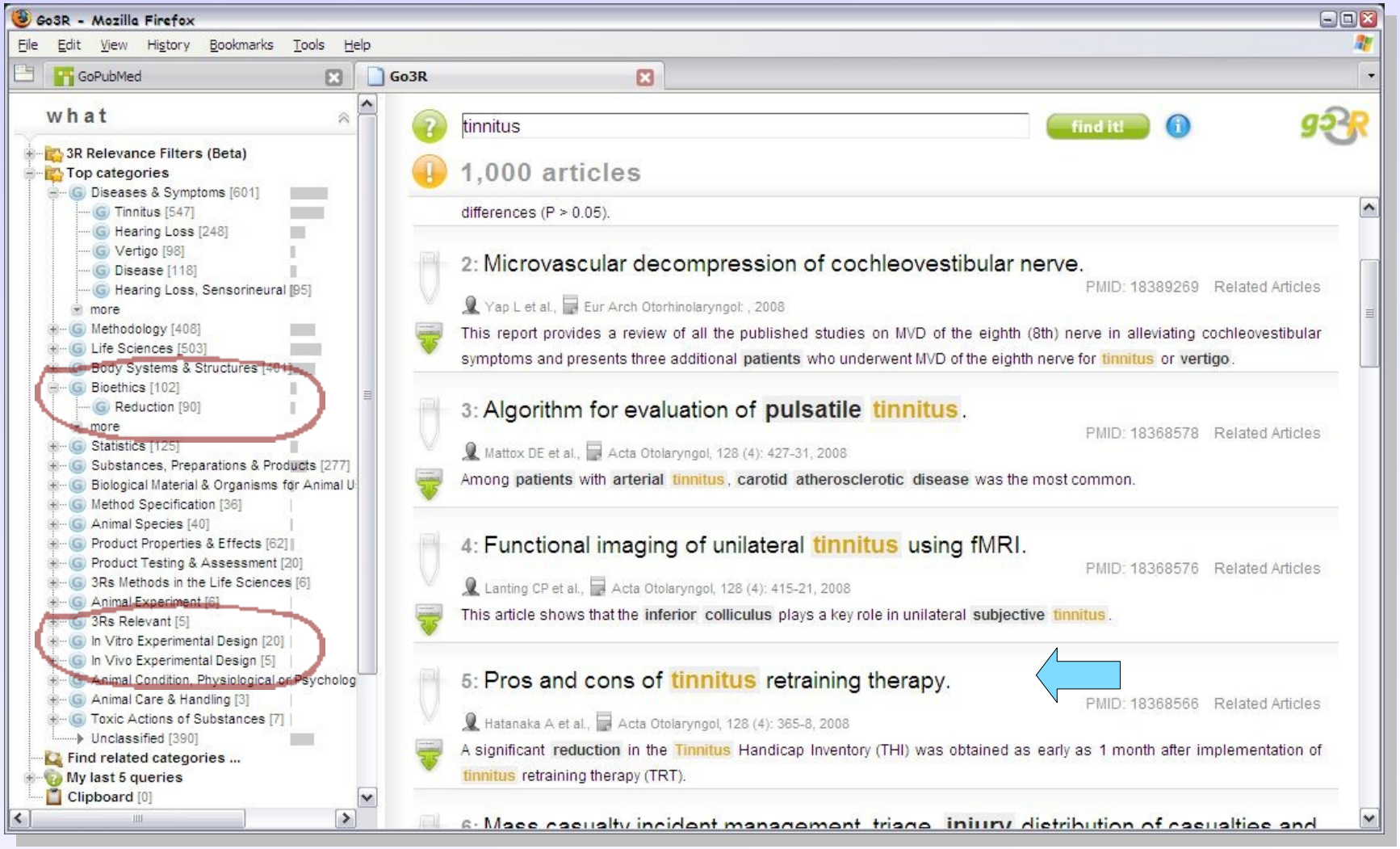
Example: improved search via ontology

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



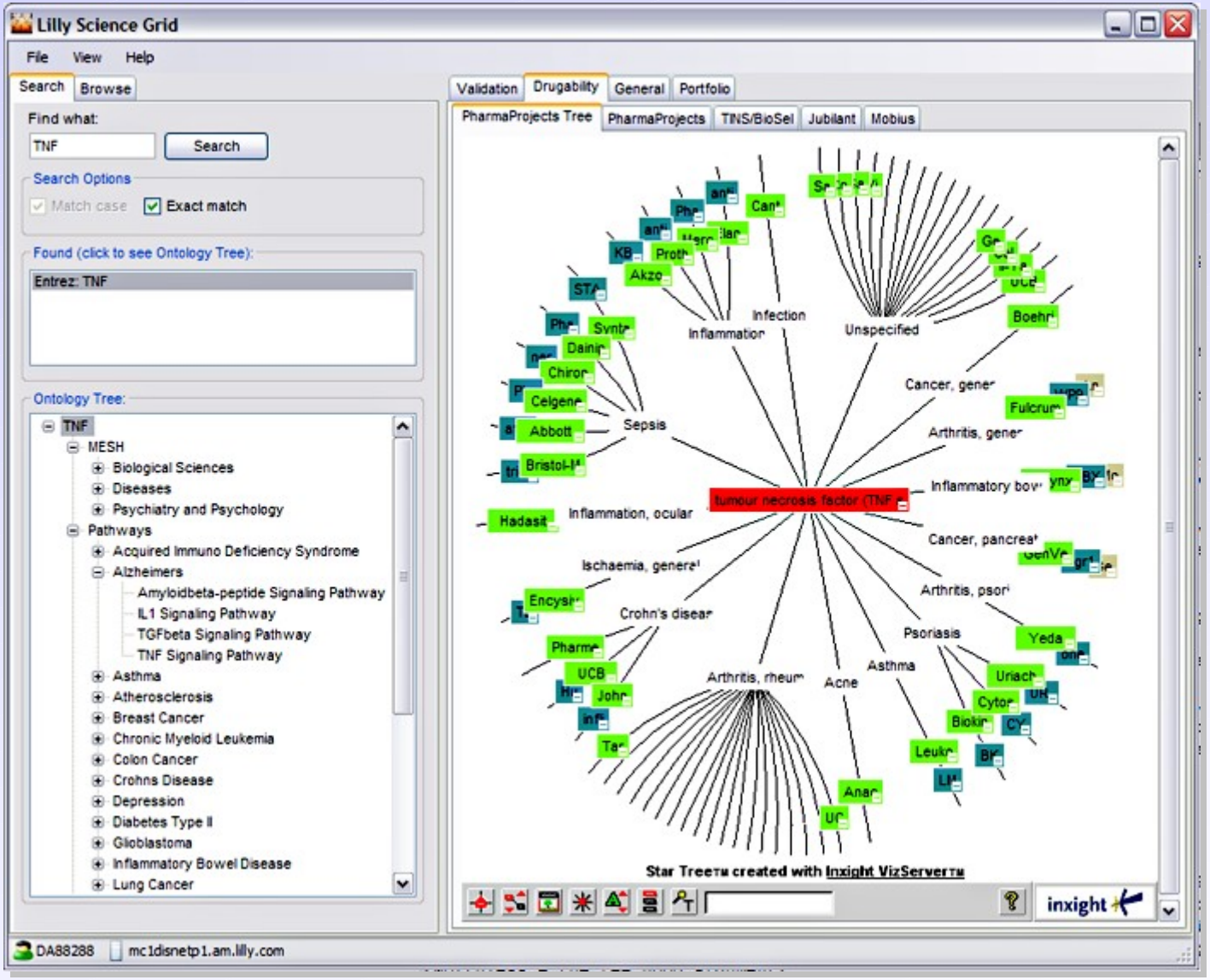
Example: improved search via ontology

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



Example: Eli Lilly's target assessment tool

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



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

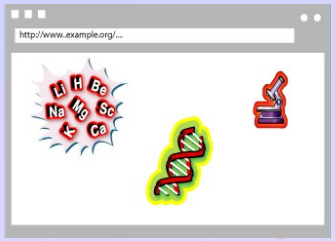
What have we achieved?

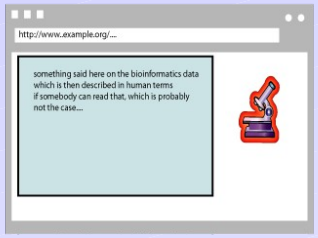
(putting all this together)

Other SW technologies

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

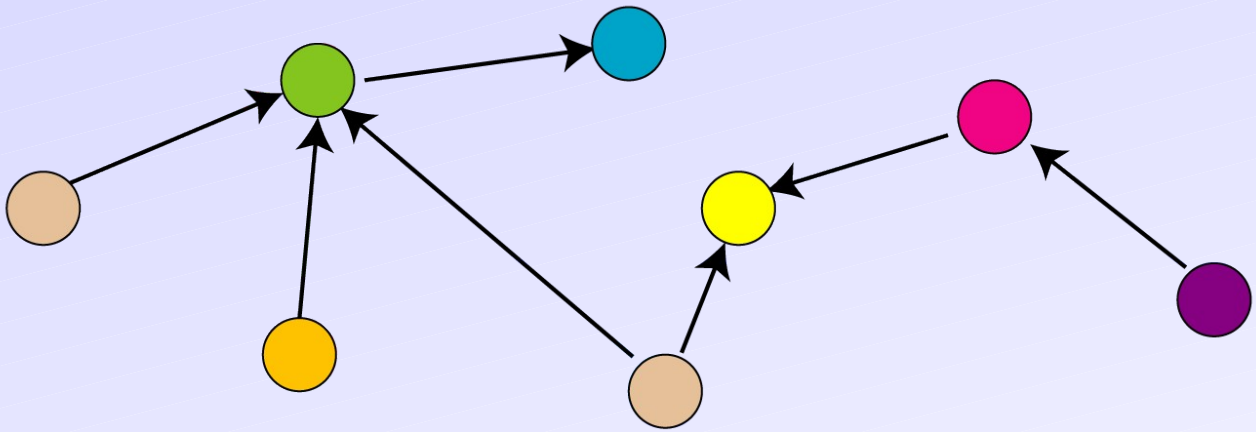
Remember the integration example?





Applications

Query, Manipulate, etc.

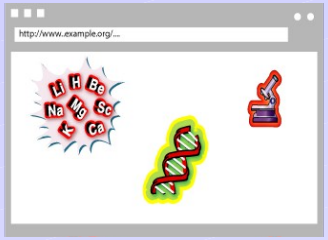


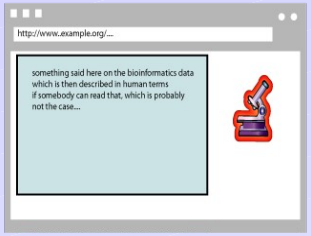
Data represented in abstract format

Map, Expose, etc.

Data in various formats

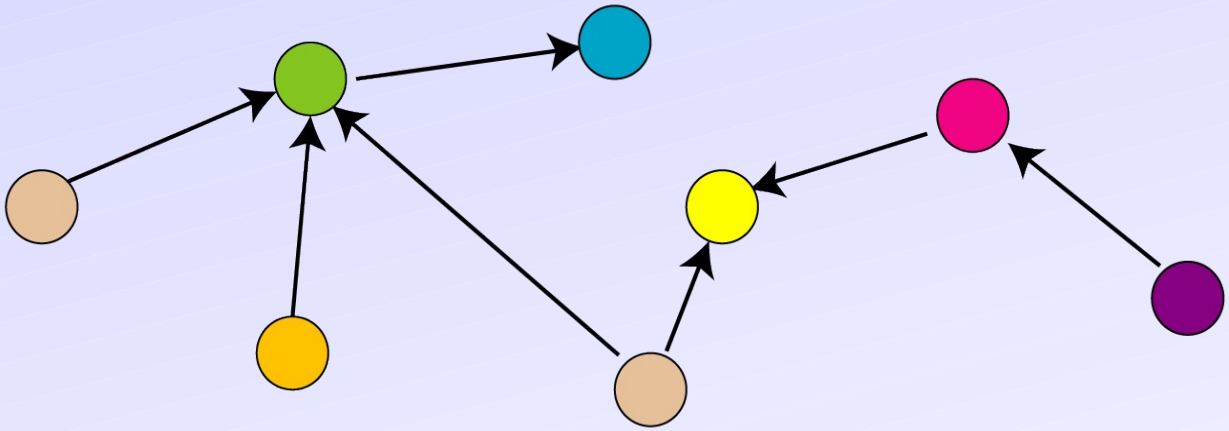
Same with what we learned





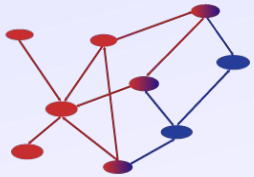
Applications

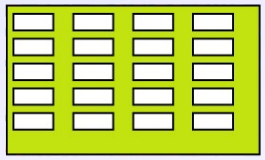
SPARQL,
OWL inferences,
etc.

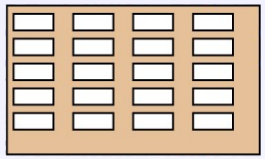


Data represented in RDF, possibly with extra knowledge (RDFS, OWL, SKOS, Rules, …)

SQL <=> RDF,
GRDDL, RDFa
etc.

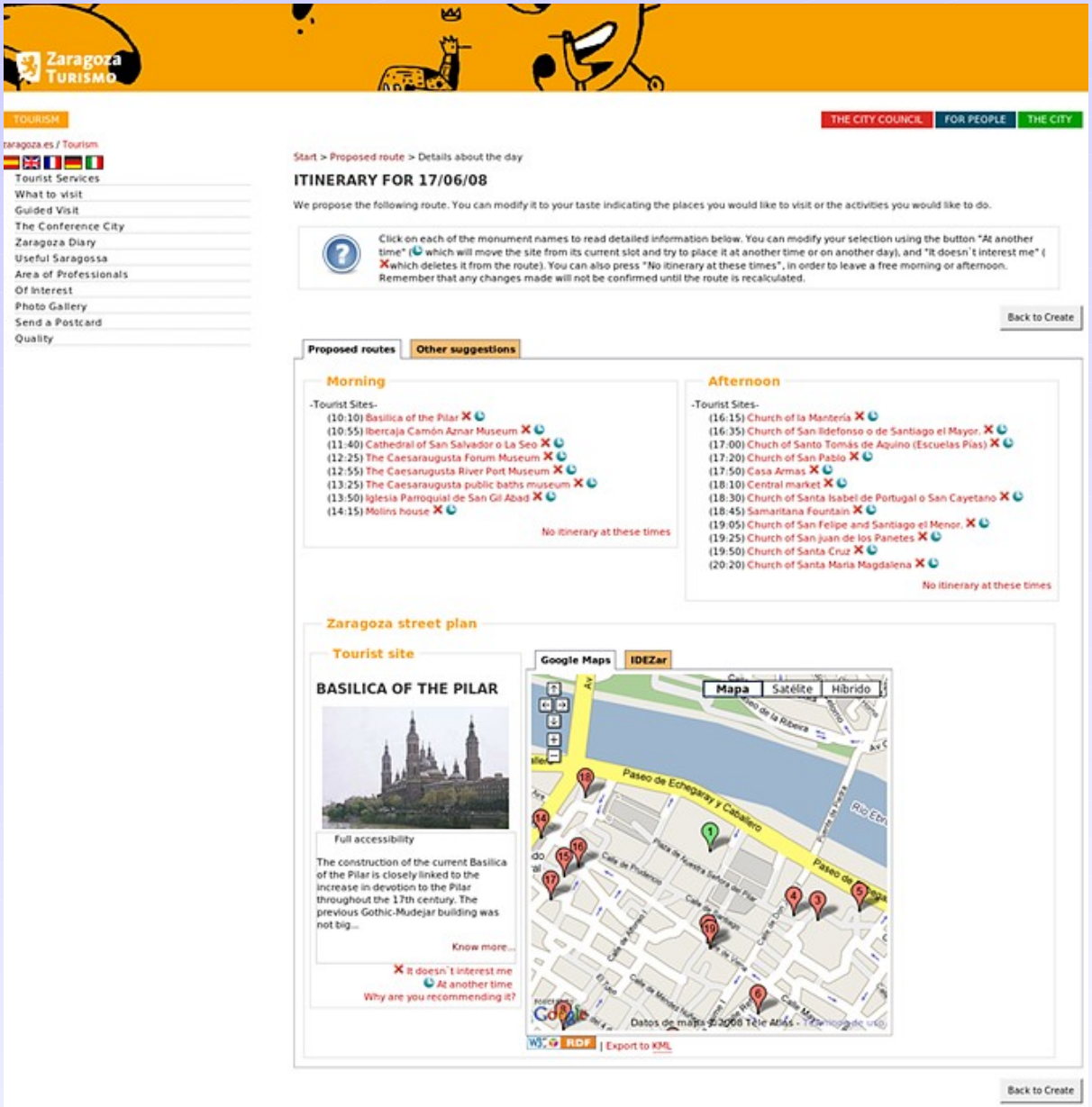






Data in various formats

Example: personalized tourist itinerary



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

Use rules on the RDF data to provide a proper itinerary

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

Available documents, resources

Available specifications: Primers, Guides

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

"Core" vocabularies

- There are also a number widely used "core vocabularies"
  - Dublin Core: about information resources, digital libraries, with extensions for rights, permissions, digital right management
  - FOAF: about people and their organizations
  - DOAP: on the descriptions of software projects
  - SIOC: Semantically-Interlinked Online Communities
  - vCard in RDF
  - …
- One should never forget: ontologies/vocabularies must be shared and reused!

Some books

- G. Antoniu and F. van Harmelen: Semantic Web Primer, 2nd edition in 2008
- D. Allemang and J. Hendler: Semantic Web for the Working Ontologist, 2008
- Jeffrey Pollock: Semantic Web for Dummies, 2009
- …

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

Further information

- Planet RDF aggregates a number of SW blogs:
  - http://planetrdf.com/
- Semantic Web Interest Group
  - a forum developers with archived (and public) mailing list, and a constant IRC presence on freenode.net#swig
  - anybody can sign up on the list:
    - http://www.w3.org/2001/sw/interest/

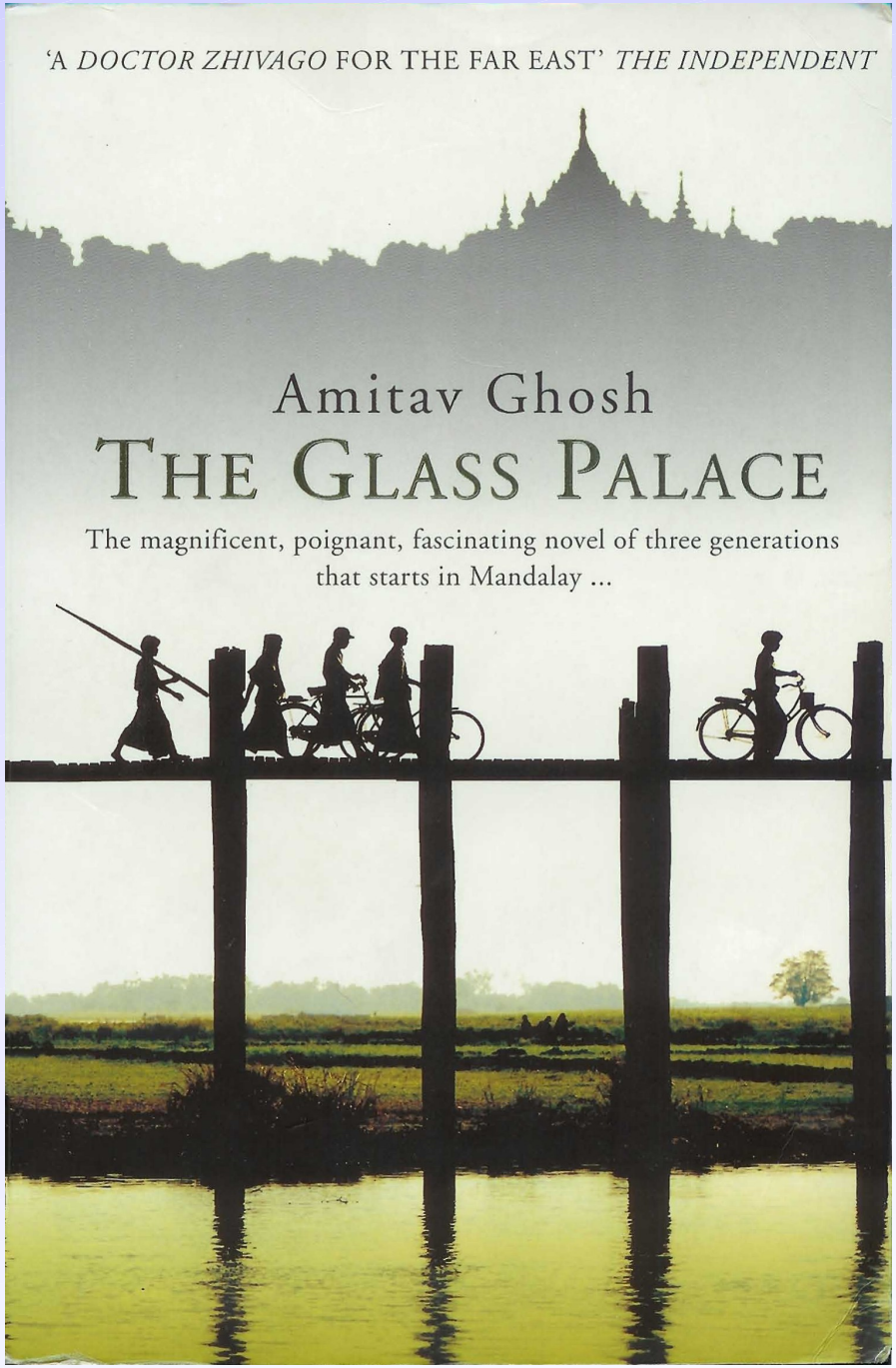
Lots of Tools (not an exhaustive list!)

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

Conclusions

- The Semantic Web is about creating a Web of Data
- There is a great and very active user and developer community, with new applications
  - witness the size and diversity of this event

By the way: the book is real



Thank you for your attention!

These slides are also available on the Web:

http://www.w3.org/2009/Talks/0615-SanJose-tutorial-IH/