

Introducing Linked Data

Irini Fundulaki¹

¹Institute of Computer Science
FORTH
&
W3C Greece Office Manager

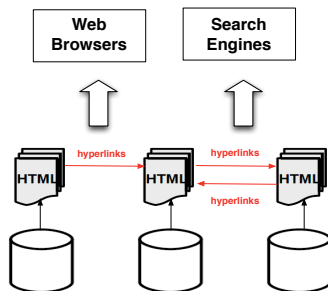
EICOS : 4th Meeting, Athens, Greece
March 29, 2013

Outline

- 1 Going from the Web of Documents to the Web of Data
- 2 Web Architecture
 - Items of Interest
 - Naming Resources: URIs
 - Resource Description Framework (RDF)
- 3 Adding Semantics to Linked Data
 - A Short Introduction
 - RDF Vocabulary Description Language (RDFS)
 - Ontology Web Language (OWL)
- 4 Linked Data LifeCycle
 - Extraction
 - Interlinking/Fusing
 - Storage/Querying
- 5 Conclusions

Traditional Web: Web of Documents

- single information space: **global filesystem**
- designed for **human consumption**
- **documents** are the primary objects with a **loose structure**
- **untyped hyperlinks** between **documents**
- URLs are the **globally unique IDs** and part of the **retrieval mechanism**
- **cannot ask expressive queries**

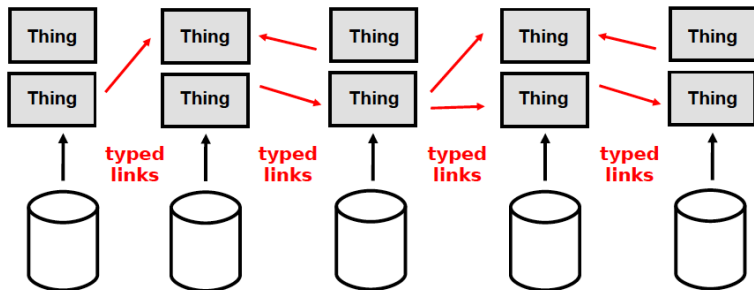


© Hartig, Cyganiac, Bizer, Hausenblas, Heath *How to Publish Linked Data on the Web*

Going from a Web of Documents to a Web of Data

- a **global database**
- designed for machines first, humans later
- **things** are the primary objects with a well **defined structure**
- **typed links** between **things**
- ability to express **structured queries**

Don't link the *documents*, link the *things*



The Web of Linked Data © Tom Heath, *An Introduction to Linked Data*

Publishing Data on the Web: Web APIs

- ✓ APIs expose structured data
- ✓ enable the development of new applications
- ✓ provide simple query access over the HTTP protocol
- proprietary interfaces
- based on a fixed set of sources
- deep structure is not visible



Web APIs create data silos in the Web

Publishing Data on the Web: Web APIs

- ✓ APIs expose structured data
- ✓ enable the development of new applications
- ✓ provide simple query access over the HTTP protocol
- proprietary interfaces
- based on a fixed set of sources
- deep structure is not visible



Web APIs create data silos in the Web

Publishing Data on the Web: Microformats

- ✓ embed structured data into HTML pages describing specific types of entities
- ✓ compatible with the idea of Web as a single information space
- ✓ applications can extract data from the pages using microformats
- only a fixed set of microformats exist (restricting the domain of interest)
- no way to connect to data items
- cannot share arbitrary data on the Web

```
1. <div class="geo">Tim ?Berners-?Lee's location is:  
2.     <span class="latitude">42.3633690</span>,  
3.     <span class="longitude">-71.091796</span>.  
4. </div>
```

geo (<http://microformats.org/wiki/geo>): denotes the *latitude* and *longitude* of the resource tied to the embedding web page

Linked Data

Beyond Web APIs and Microformats

- *publishing* and *interlinking* **structured data** on the Web using Semantic Web technologies
- *sharing structured data* on the Web as easily as documents are shared today
- is about using the Web to create *typed links* between different sources
- accessed through a *single, standardized access mechanism*

Web Of Data, Semantic Web (Data Commons)

a space where people and organizations can post and consume data about anything

Linked Data Principles

1. Use URIs as names for things
2. Use HTTPs URIs so that people can look up those names
3. When someone looks up a URI, provide useful RDF information
4. Include RDF statements that link to other URIs so that they can discover related things

Tim Berners-Lee, 2007

Is Linked Data Real?

Linking Open Data

Linking Open Datasets on the Web (LOD)

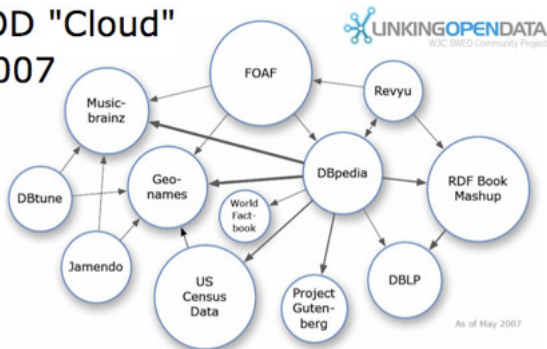
- **publish public open data** as **Linked Data** on the Web
- **interlink entities** between **heterogeneous** sources



Evolution of Linked Open Datasets

May 2007

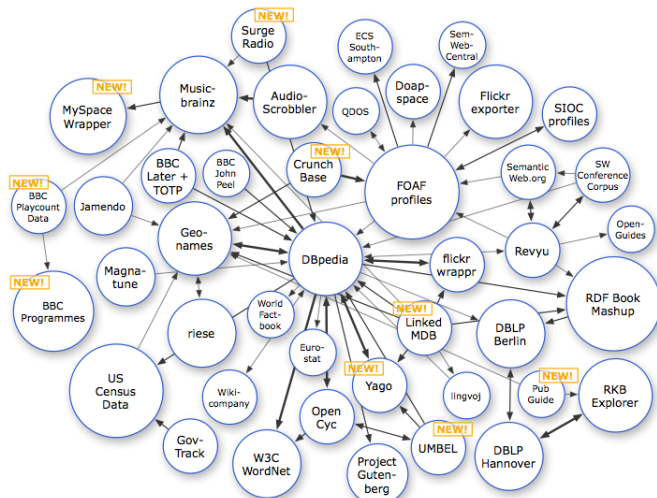
The LOD "Cloud" May 2007



Over 1 billion RDF triples served on the Web
Around 120,000 RDF links between data sources

Evolution of Linked Open Datasets

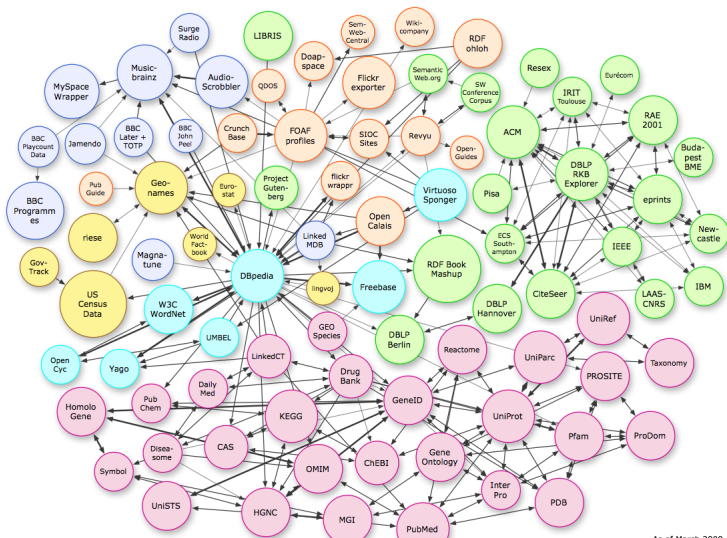
September 2008



As of September 2008

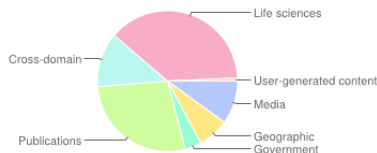
Evolution of Linked Open Datasets

March 2009

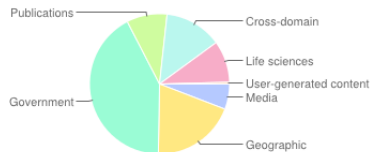


Linked Data in numbers (September 2011)

Domain	# datasets	Triples	%	(Out)-Links	%
Media	25	1,841,852,061	5.82%	50,440,705	10.01%
Geographic	31	6,145,532,484	19.43%	35,812,328	7.11%
Government	49	13,315,009,400	42.09%	19,343,519	3.84%
Publications	87	2,950,720,693	9.33%	139,925,218	27.76%
Cross-domain	41	4,184,635,715	13.23%	63,183,065	12.54%
Life sciences	41	3,036,336,004	9.60%	191,844,090	38.06%
User-gen. cont.	20	134,127,413	0.42%	3,449,143	0.68%
	295	31,634,213,770		503,998,829	



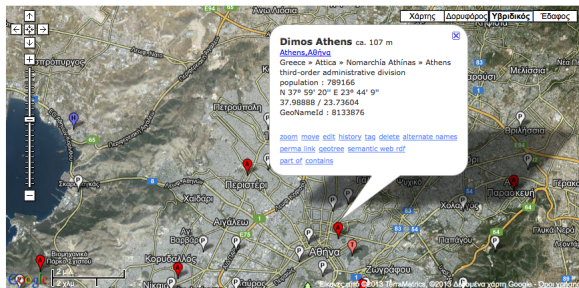
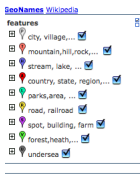
(a) Distribution of links per domain



(b) Distribution of triples per domain

LinkedData: GeoNames Geographical Database

- covers all countries
- contains over eight million placenames
- considers facets/features
- integrates data from various datasources: National Geospatial-Intelligence Agency's (NGA), U.S. Geological Survey Geographic Names Information System, Brasileiro de Geografia Estatística (IBGE), ...



only 50 objects displayed, zoom in or deselect some features


Name	country	feature	km to center
1 Athens	Greece	capital of a political entity	0.05 km

The DBpedia Knowledge Base

- crowd-sourced community effort to extract structured information from Wikipedia
- links available datasets on the Web to data extracted from Wikipedia
- versions of DBpedia in 111 languages
- data is published in a **consistent ontology**
- accessible through different SPARQL end-points

About: [Greece](#)

An Entity of Type : [populated place](#), from Named Graph : <http://dbpedia.org>, within Data Space : [dbpedia.org](#)



Greece Listen/ɡrɪs/, officially the Hellenic Republic, is a country in Southern Europe, politically considered part of Western Europe. Athens is the capital and the largest city in the country (its urban area also including Piraeus). The population of the country is around 11 million. Greece has land borders with Albania, the Republic of Macedonia and Bulgaria to the north, and Turkey to the east.

Property	Value
dbpedia-owl:PopulatedPlace/areaTotal	<ul style="list-style-type: none"> 131990.0 131944.35429295717
dbpedia-owl:PopulatedPlace/populationDensity	<ul style="list-style-type: none"> 85.32857703788055 85.3
dbpedia-owl:abstract	<ul style="list-style-type: none"> Grècia, oficialment la República Hel·lènica (en grec, Ελληνική Δημοκρατία Ellīnikī́ Dīmokratía) és un país del sud-est d'Europa, situat a la punta meridional de la península balcànica. Té frontera amb Albània, la República de Macedònia i Bulgària al nord, i Turquia a l'est. El mar Egeu és a l'est de Grècia, el mar Jònic a l'oest i el mar Mediterrani al sud. Grècia inclou un vast nombre d'illes (aproximadament 1.440 de les quals 227 són habitades), incloent-hi Creta, Dodecanès, les Cíclades i les Illes Jòniques, entre altres.


DBpedia Live enables such a continuous synchronization between DBpedia and Wikipedia: <http://live.dbpedia.org/>

The DBpedia Knowledge Base

- crowd-sourced community effort to extract structured information from Wikipedia
- links available datasets on the Web to data extracted from Wikipedia
- versions of DBpedia in 111 languages
- data is published in a **consistent ontology**
- accessible through different SPARQL end-points

About: [Greece](#)

An Entity of Type : [populated place](#), from Named Graph : <http://dbpedia.org>, within Data Space : [dbpedia.org](#)



Greece Listen/ɡrɪs/, officially the Hellenic Republic, is a country in Southern Europe, politically considered part of Western Europe. Athens is the capital and the largest city in the country (its urban area also including Piraeus). The population of the country is around 11 million. Greece has land borders with Albania, the Republic of Macedonia and Bulgaria to the north, and Turkey to the east.

Property	Value
dbpedia-owl:PopulatedPlace/areaTotal	<ul style="list-style-type: none"> 131990.0 131944.35429295717
dbpedia-owl:PopulatedPlace/populationDensity	<ul style="list-style-type: none"> 85.32857703788055 85.3
dbpedia-owl:abstract	<ul style="list-style-type: none"> Grècia, oficialment la República Hel·lènica (en grec, Ελληνική Δημοκρατία Ellīnikī́ Dīmokratía) és un país del sud-est d'Europa, situat a la punta meridional de la península balcànica. Té frontera amb Albània, la República de Macedònia i Bulgària al nord, i Turquia a l'est. El mar Egeu és a l'est de Grècia, el mar Jònic a l'oest i el mar Mediterrani al sud. Grècia inclou un vast nombre d'illes (aproximadament 1.440 de les quals 227 són habitades), incloent-hi Creta, Dodecanès, les Cíclades i les Illes Jòniques, entre altres.

DBpedia Live enables such a continuous synchronization between DBpedia and Wikipedia: <http://live.dbpedia.org/>

Linked Data Applications

- Linked Data Browsers
- Linked Data Mashups
- Search Engines

Linked Data Applications

- Linked Data Browsers
- Linked Data Mashups
- Search Engines

Linked Data Browsers

Linked Data Browser explores the Linked Data Cloud:
recognizes and follows RDF links to RDF resources

- Tabulator Browser
- Marbles (FU Berlin, DE)
- OpenLink RDF Browser (OpenLink, UK)
- Zitgist RDF Browser (Zitgist, USA)
- Disco Hyperdata Browser (FU Berlin, DE)
- Fenfire (DERI, Ireland)

installed and configured as add-ons to existing browsers

Linked Data Browsers

Linked Data Browser explores the Linked Data Cloud:
recognizes and follows RDF links to RDF resources

- Tabulator Browser
- Marbles (FU Berlin, DE)
- OpenLink RDF Browser (OpenLink, UK)
- Zitgist RDF Browser (Zitgist, USA)
- Disco Hyperdata Browser (FU Berlin, DE)
- Fenfire (DERI, Ireland)

installed and configured as add-ons to existing browsers

Linked Data Browsers

Linked Data Browser explores the Linked Data Cloud:
recognizes and follows RDF links to RDF resources

- Tabulator Browser
- Marbles (FU Berlin, DE)
- OpenLink RDF Browser (OpenLink, UK)
- Zitgist RDF Browser (Zitgist, USA)
- Disco Hyperdata Browser (FU Berlin, DE)
- Fenfire (DERI, Ireland)

installed and configured as add-ons to existing browsers

Linked Data Mashups

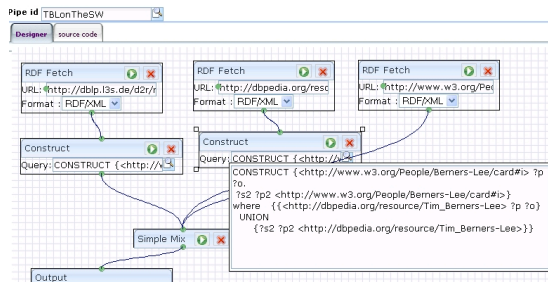
Linked Data Mashups: Domain-specific applications that use Linked Data

- **Revyu.com**: everything you can review, uses Linked Data to enrich ratings, accessed through SPARQL endpoint
- **DBtune Slashfacet**: Visualizes music-related Linked Data
 - data from LastFM, MySpace, BBC, Magnatune
 - 14 billion triples accessed through SPARQL endpoints

Linked Data Mashups

DERI Semantic Web Pipes

- build **open source, extendable, embeddable Web Data mashups** in the spirit of **Yahoo! Pipes**
 - produce as output a stream of data in different formats (XML, RDF, JSON) to be used by applications.



```
<pipe>
<construct>
<source><fetch><location>http://dblp.l3s.de/d2r/resource/authors/Tim_Berners-Lee</source></fetch></location>
<query>
<| CDATA[
  CONSTRUCT { <http://www.w3.org/People/Berners-Lee/card#i> ?p ?o.
    ?s2 ?p2 <http://www.w3.org/People/Berners-Lee/card#i> } where
    { { <http://dblp.l3s.de/d2r/resource/authors/Tim_Berners-Lee> ?p ?o } UNION
      { ?s2 ?p2 <http://dbpedia.org/resource/authors/Tim_Berners-Lee> } }
  ]
</|>
</construct>
```

Linked Data Search Engines

- **Falcons** (IWS, China)
 - supports keyword based search to objects, concepts (classes and properties), ontologies and RDF datasets
- **Sindice** (DERI, Ireland)
 - explores and integrates RDF content
 - keyword based search
 - structured search through SPARQL queries
- **MicroSearch** (Yahooo, Spain)
 - enriched Yahoo! search
 - extracts and displays the metadata of pages
- **Watson** (Open University, UK)
 - supports keyword based search
 - graphical representation of datasets
- **Semantic Web Search Engine (SWSE)** (DERI, Ireland)
- **Semantic Web Ontology Swoogle** (UMBC, USA)
 - supports keyword based search to objects, concepts (classes and properties), ontologies and RDF datasets

Outline

- 1 Going from the Web of Documents to the Web of Data
- 2 **Web Architecture**
 - **Items of Interest**
 - Naming Resources: URIs
 - Resource Description Framework (RDF)
- 3 Adding Semantics to Linked Data
 - A Short Introduction
 - RDF Vocabulary Description Language (RDFS)
 - Ontology Web Language (OWL)
- 4 Linked Data LifeCycle
 - Extraction
 - Interlinking/Fusing
 - Storage/Querying
- 5 Conclusions

Web Architecture: Items of Interest

Resources

- things whose **properties** and **relationships** we want to describe in the data
- **identified** using **Uniform Resource Identifiers (URIs)**
- Distinguished into:
 - **Information Resources:** documents, images, media files, ...
 - **Non-Information Resources:** people, places, proteins, cars, ...

URI Aliases

- different information providers talk about the **same** non-information resource using different URIs
 - DBpedia: <http://dbpedia.org/resource/Berlin> identifies *place* "Berlin"
 - GeoNames: <http://sws.geonames.org/2950159/> identifies *place* "Berlin"

Web Architecture: Items of Interest

Resources

- things whose **properties** and **relationships** we want to describe in the data
- **identified** using **Uniform Resource Identifiers (URIs)**
- Distinguished into:
 - **Information Resources**: documents, images, media files, ...
 - **Non-Information Resources**: people, places, proteins, cars, ...

URI Aliases

- different information providers talk about the **same** non-information resource using different URIs
 - DBpedia: <http://dbpedia.org/resource/Berlin> identifies *place* "Berlin"
 - GeoNames: <http://sws.geonames.org/2950159/> identifies *place* "Berlin"

Web Architecture: Items of Interest

Representation: stream of bytes in a certain *format* such as HTML, RDF/XML, Turtle, NTriples, JSON, JPEG, PDF,

- HTML chosen to represent descriptions read by humans
- RDF/XML, Turtle, NTriples, ... to represent descriptions managed by machines (RDF data serialization formats)

Web Architecture: Items of Interest

Dereferencing URIs looking up a URI on the Web in order to get information about the referenced resource.

- *Information Resource* server of the URI owner returns a snapshot of the resource's current state and returns it in the requested format
- *Non-Information Resource* are dereferenced indirectly.

Content Negotiation clients send HTTP requests with a header indicating the kind of preferred representation

Associated Description is the result of dereferencing the URI of a *non-information* resource

the description of a non-information resource can have multiple representations

Outline

- 1 Going from the Web of Documents to the Web of Data
- 2 **Web Architecture**
 - Items of Interest
 - **Naming Resources: URIs**
 - Resource Description Framework (RDF)
- 3 Adding Semantics to Linked Data
 - A Short Introduction
 - RDF Vocabulary Description Language (RDFS)
 - Ontology Web Language (OWL)
- 4 Linked Data LifeCycle
 - Extraction
 - Interlinking/Fusing
 - Storage/Querying
- 5 Conclusions

Naming Resource: URIs

- provide a simple way to create **globally unique names** in a **decentralized fashion**
- used as a **name** but also as a **means** of accessing information
- `http://..` is the only scheme that is supported by **all** browsers
- **stable** and **persistent**, defined in a **controlled namespace**
- defined **independently** of any implementation
 - `http://dbpedia.org/resource/Munich`
 - `http://www.demos/dbpedia/cgi-bin/resources.php?id=Munich`
- **dereferencable**:
 - HTTP clients can look up the URI using the HTTP protocol
 - URIs that refer to information resources, return documents
 - URIs that identify real-world objects and abstract concepts (non-information resources) return the **descriptions**

URIs for Information and Non-Information Resources by Example

- *Big Lynx* independent television production company with website

`http://biglynx.co.uk/`

- staff members: **non-information resources**

`http://biglynx.co.uk/people/libby-miller`

- RDF/XML file that stores information about staff members: **information resources**

`http://biglynx.co.uk/people/libby-miller.rdf`

Dereferencing URIs for non-information resources

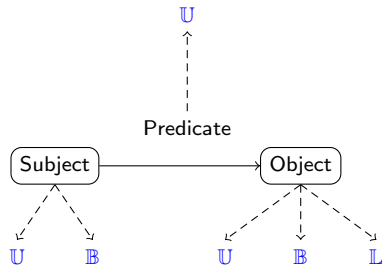


Outline

- 1 Going from the Web of Documents to the Web of Data
- 2 **Web Architecture**
 - Items of Interest
 - Naming Resources: URIs
 - **Resource Description Framework (RDF)**
- 3 Adding Semantics to Linked Data
 - A Short Introduction
 - RDF Vocabulary Description Language (RDFS)
 - Ontology Web Language (OWL)
- 4 Linked Data LifeCycle
 - Extraction
 - Interlinking/Fusing
 - Storage/Querying
- 5 Conclusions

Resource Description Framework

- RDF is the W3C standard for representing information in the Web



\mathcal{U} = set of **U**RLs
 \mathcal{B} = set of **B**lank nodes
 \mathcal{L} = set of **L**iterals

$(s, p, o) \in (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$ is called an **RDF triple**

set of RDF triples is an **RDF graph**

RDF graph is a **node** and **edge-labeled directed graph**

RDF by Example

	<i>(subject)</i>	<i>(predicate)</i>	<i>(object)</i>
t_1	csail:libby	rdf:type	foaf:Person
t_2	csail:libby	foaf:name	"Libby Miller"
t_3	csail:libby	owl:sameAs	webwho:libby_miller

- *subject*: the URI identifying the described resource
- *object*: can either be a simple literal value or the URI of another resource
- *predicate*: the URI indicating the relation between subject and object
- csail:libby: *subject* URI of the resource of interest
- rdf:type, foaf:name, owl:sameAs: *predicate* URIs coming from vocabularies
 - Resource Description Framework - RDF
 - Friend Of A Friend - FOAF
 - Ontology Web Language - OWL.

RDF by Example

	<i>(subject)</i>	<i>(predicate)</i>	<i>(object)</i>
t_1	csail:libby	rdf:type	foaf:Person
t_2	csail:libby	foaf:name	"Libby Miller"
t_3	csail:libby	owl:sameAs	webwho:libby_miller

- *subject*: the URI identifying the described resource
- *object*: can either be a simple literal value or the URI of another resource
- *predicate*: the URI indicating the relation between subject and object
- csail:libby: *subject* URI of the resource of interest
- rdf:type, foaf:name, owl:sameAs: *predicate* URIs coming from vocabularies
 - Resource Description Framework - RDF
 - Friend Of A Friend - FOAF
 - Ontology Web Language - OWL.

RDF by Example

	<i>(subject)</i>	<i>(predicate)</i>	<i>(object)</i>
t_1	csail:libby	rdf:type	foaf:Person
t_2	csail:libby	foaf:name	"Libby Miller"
t_3	csail:libby	owl:sameAs	webwho:libby_miller

- *subject*: the URI identifying the described resource
- *object*: can either be a simple literal value or the URI of another resource
- *predicate*: the URI indicating the relation between subject and object
- csail:libby: *subject* URI of the resource of interest
- rdf:type, foaf:name, owl:sameAs: *predicate* URIs coming from vocabularies
 - Resource Description Framework - RDF
 - Friend Of A Friend - FOAF
 - Ontology Web Language - OWL.

RDF by Example

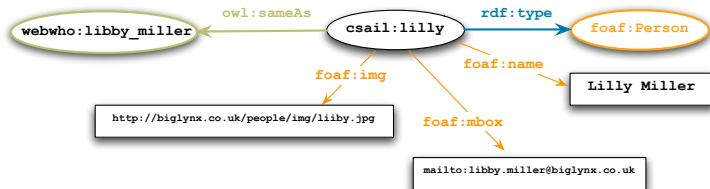
	<i>(subject)</i>	<i>(predicate)</i>	<i>(object)</i>
t_1	csail:libby	rdf:type	foaf:Person
t_2	csail:libby	foaf:name	"Libby Miller"
t_3	csail:libby	owl:sameAs	webwho:libby_miller

- *subject*: the URI identifying the described resource
- *object*: can either be a simple literal value or the URI of another resource
- *predicate*: the URI indicating the relation between subject and object
- csail:libby: *subject* URI of the resource of interest
- rdf:type, foaf:name, owl:sameAs: *predicate* URIs coming from vocabularies
 - Resource Description Framework - RDF
 - Friend Of A Friend - FOAF
 - Ontology Web Language - OWL.

RDF by Example

	(<i>subject</i>)	(<i>predicate</i>)	(<i>object</i>)
t_1	csail:libby	rdf:type	foaf:Person
t_2	csail:libby	foaf:name	"Libby Miller"
t_3	csail:libby	foaf:mbox	"mailto:libby.miller@biglynx.co.uk"
t_4	csail:libby	foaf:img	"http://biglynx.co.uk/people/img/libby.jpg"
t_5	csail:libby	owl:sameAs	webwho:libby_miller

the above set of RDF triples form an RDF graph



Resource Description Framework

Benefits of the RDF Data Model

- **generic** and **simple graph-based** data model
- information from **heterogeneous** sources merges naturally: resources with the same URIs denote the same non-information resource
- **schemas are transparent**: information is in the URIS
- **structure** is added using **schema languages** and represented as RDF triples
- Web browsers use URIs to retrieve information

Resource Description Framework

Benefits of the RDF Data Model

- **generic** and **simple graph-based** data model
- information from **heterogeneous** sources merges naturally: resources with the same URIs denote the same non-information resource
- **schemas are transparent**: information is in the URIS
- **structure** is added using **schema languages** and represented as RDF triples
- Web browsers use URIs to retrieve information

Resource Description Framework

Benefits of the RDF Data Model

- **generic** and **simple graph-based** data model
- information from **heterogeneous** sources merges naturally: resources with the same URIs denote the same non-information resource
- **schemas are transparent**: information is in the URIS
- **structure** is added using **schema languages** and represented as RDF triples
- Web browsers use URIs to retrieve information

Resource Description Framework

Benefits of the RDF Data Model

- **generic** and **simple graph-based** data model
- information from **heterogeneous** sources merges naturally: resources with the same URIs denote the same non-information resource
- **schemas are transparent**: information is in the URIS
- **structure** is added using **schema languages** and represented as RDF triples
- Web browsers use URIs to retrieve information

Resource Description Framework

Benefits of the RDF Data Model

- **generic** and **simple graph-based** data model
- information from **heterogeneous** sources merges naturally: resources with the same URIs denote the same non-information resource
- **schemas are transparent**: information is in the URIS
- **structure** is added using **schema languages** and represented as RDF triples
- Web browsers use URIs to retrieve information

Resource Description Framework

Benefits of the RDF Data Model

- **generic** and **simple graph-based** data model
- information from **heterogeneous** sources merges naturally: resources with the same URIs denote the same non-information resource
- **schemas are transparent**: information is in the URIS
- **structure** is added using **schema languages** and represented as RDF triples
- Web browsers use URIs to retrieve information

Resource Description Framework

Benefits of the RDF Data Model

- **generic** and **simple graph-based** data model
- information from **heterogeneous** sources merges naturally: resources with the same URIs denote the same non-information resource
- **schemas are transparent**: information is in the URI
- **structure** is added using **schema languages** and represented as RDF triples
- Web browsers use URIs to retrieve information

RDF Serialization Formats

Serialization Formats: Creating information resources out of non-information resources

- **RDF/XML**: standardized by the W3C and is widely used to publish Linked Data on the Web but is heavy and difficult for humans to write and read

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <rdf:RDF
3   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4   xmlns:foaf="http://xmlns.com/foaf/0.1/"
5   xmlns:csail="http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf#">
6
7   <rdf:Description rdf:about="csail:libby">
8     <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
9     <foaf:name>Libby Miller</foaf:name>
10  </rdf:Description>
11 </rdf:RDF>
```

RDF Serialization Formats

- **RDFa**: RDF triples are interwoven in HTML code but publishing infrastructure is not yet in place

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
    "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:csail="http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf#">
3
4  <head>
5    <meta http-equiv="Content-Type" content="application/xhtml+xml; charset=UTF-8"/>
6    <title>Profile Page for Libby Miller
7  </head>
8
9  <body>
10   <div about="csail:libby" typeof="foaf:Person">
11     <span property="foaf:name">Libby Miller</span></div>
12   </body>
13 </html>
```

RDF Serialization Formats

- Turtle is a plain text format and is the format of choice for reading RDF triples or writing them by hand

```
1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2 @prefix foaf: <http://xmlns.com/foaf/0.1/> .
3 @prefix csail: <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf#> .
4 csail:libby_miller
5   rdf:type foaf:Person ;
6   foaf:name "Libby Miller" .
```

- N-Triples is a subset of Turtle, minus features such as namespace prefixes and shorthands leading to **redundancy** but allows
 - parsing and loading one line at a time, compression to handle big files, exchange, main memory parsing

```
1 <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf#libby_miller>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  <http://xmlns.com/foaf/0.1/Person> .
2 <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf#libby_miller>
  <http://xmlns.com/foaf/0.1/name> "Libby Miller" .
```

Creating Linked Data: RDF Links

Literal Triples: have an RDF literal as the object used to describe the **properties** of resources

Object Triples: represent **typed links between two resources** (the *subject* and *object* are URIs) \Rightarrow **RDF Links**

- **Relationship Links:** point at related things in other data sources
- **Identity Links:** point at URI aliases used by other data sources to identify the same real-world object or abstract concept.
- **Vocabulary Links** point from data to the definitions of the vocabulary terms that are used to represent the data

Creating Linked Data: RDF Links

Literal Triples: have an RDF literal as the object used to describe the **properties** of resources

Object Triples: represent **typed links between two resources** (the *subject* and *object* are URIs) \Rightarrow **RDF Links**

- **Relationship Links:** point at related things in other data sources
- **Identity Links:** point at URI aliases used by other data sources to identify the same real-world object or abstract concept.
- **Vocabulary Links** point from data to the definitions of the vocabulary terms that are used to represent the data

Creating Linked Data: RDF Links

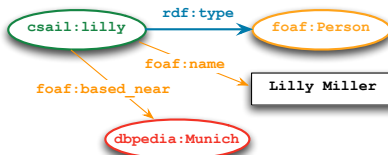
Literal Triples: have an RDF literal as the object used to describe the **properties** of resources

Object Triples: represent typed links between two resources (the *subject* and *object* are URIs) \Rightarrow **RDF Links**

- **Relationship Links:** point at related things in other data sources
- **Identity Links:** point at URI aliases used by other data sources to identify the same real-world object or abstract concept.
- **Vocabulary Links** point from data to the definitions of the vocabulary terms that are used to represent the data

Creating Linked Data: RDF Links

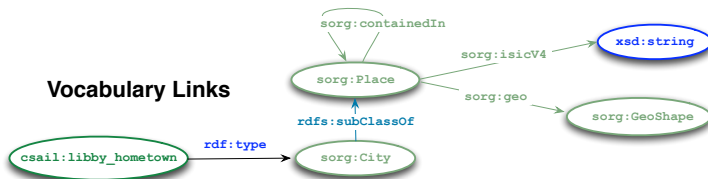
Relationship Links



Identity Links

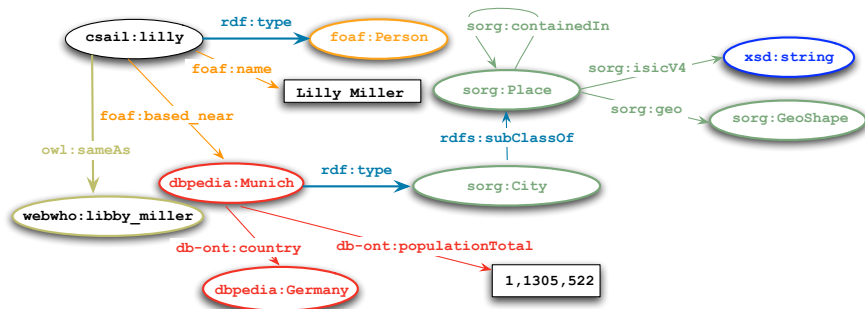


Vocabulary Links



Creating Linked Data: RDF Links

RDF links: Foundation of Linked Data



Outline

- 1 Going from the Web of Documents to the Web of Data
- 2 Web Architecture
 - Items of Interest
 - Naming Resources: URIs
 - Resource Description Framework (RDF)
- 3 Adding Semantics to Linked Data
 - A Short Introduction
 - RDF Vocabulary Description Language (RDFS)
 - Ontology Web Language (OWL)
- 4 Linked Data LifeCycle
 - Extraction
 - Interlinking/Fusing
 - Storage/Querying
- 5 Conclusions

Adding Semantics to Linked Data

- **RDF** is a **generic, abstract** data model for describing resources in the form of **triples**
- **does not** provide ways of defining **classes, properties, constraints** etc.

Languages

- Simple Knowledge Organization System (SKOS) to define **taxonomies/thesauri**
- RDF Vocabulary Description Language (RDF Schema - RDFS) to define **vocabularies**
- Ontology Web Language (OWL) to define **ontologies**

Adding Semantics to Linked Data

- **RDF** is a **generic, abstract** data model for describing resources in the form of **triples**
- **does not** provide ways of defining **classes, properties, constraints** etc.

Languages

- Simple Knowledge Organization System (SKOS) to define **taxonomies/thesauri**
- RDF Vocabulary Description Language (RDF Schema - RDFS) to define **vocabularies**
- Ontology Web Language (OWL) to define **ontologies**

Outline

- 1 Going from the Web of Documents to the Web of Data
- 2 Web Architecture
 - Items of Interest
 - Naming Resources: URIs
 - Resource Description Framework (RDF)
- 3 Adding Semantics to Linked Data
 - A Short Introduction
 - **RDF Vocabulary Description Language (RDFS)**
 - Ontology Web Language (OWL)
- 4 Linked Data LifeCycle
 - Extraction
 - Interlinking/Fusing
 - Storage/Querying
- 5 Conclusions

RDF Vocabulary Description Language: RDF Schema

- designed to introduce useful **semantics** to RDF triples
- typing**: defining *classes*, *properties*, *instances*
- relationships between types and instances: **subsumption** and **instantiation**
- constraints**: *domain* and *range* for properties
- inference rules** to entail **new, inferred** knowledge
- schemas are represented as RDF triples

	(<i>subject</i>)	(<i>predicate</i>)	(<i>object</i>)
t_1	sorg:Place	rdf:type	rdfs:Class
t_2	sorg:City	rdfs:subClassOf	sorg:Place
t_3	sorg:containedIn	rdf:type	rdf:Property
t_4	sorg:containedIn	rdfs:domain	sorg:Place
t_5	sorg:containedIn	rdfs:range	sorg:Place
t_6	csail:libby_home_town	rdf:type	sorg:City

RDF Vocabulary Description Language: RDF Schema

- designed to introduce useful **semantics** to RDF triples
- typing**: defining *classes*, *properties*, *instances*
- relationships between types and instances: **subsumption** and **instantiation**
- constraints**: *domain* and *range* for properties
- inference rules** to entail **new, inferred** knowledge
- schemas are represented as RDF triples

	(<i>subject</i>)	(<i>predicate</i>)	(<i>object</i>)
t_1	sorg:Place	rdf:type	rdfs:Class
t_2	sorg:City	rdfs:subClassOf	sorg:Place
t_3	sorg:containedIn	rdf:type	rdf:Property
t_4	sorg:containedIn	rdfs:domain	sorg:Place
t_5	sorg:containedIn	rdfs:range	sorg:Place
t_6	csail:libby_home_town	rdf:type	sorg:City

RDF Vocabulary Description Language: RDF Schema

- designed to introduce useful **semantics** to RDF triples
- typing**: defining *classes*, *properties*, *instances*
- relationships between types and instances: **subsumption** and **instantiation**
- constraints**: *domain* and *range* for properties
- inference rules** to entail **new, inferred** knowledge
- schemas are represented as RDF triples

	(<i>subject</i>)	(<i>predicate</i>)	(<i>object</i>)
t_1	sorg:Place	rdf:type	rdfs:Class
t_2	sorg:City	rdfs:subClassOf	sorg:Place
t_3	sorg:containedIn	rdf:type	rdf:Property
t_4	sorg:containedIn	rdfs:domain	sorg:Place
t_5	sorg:containedIn	rdfs:range	sorg:Place
t_6	csail:libby_home_town	rdf:type	sorg:City

RDF Vocabulary Description Language: RDF Schema

- designed to introduce useful **semantics** to RDF triples
- typing**: defining *classes*, *properties*, *instances*
- relationships between types and instances: **subsumption** and **instantiation**
- constraints**: *domain* and *range* for properties
- inference rules** to entail **new**, **inferred** knowledge
- schemas are represented as RDF triples

	(<i>subject</i>)	(<i>predicate</i>)	(<i>object</i>)
t_1	sorg:Place	rdf:type	rdfs:Class
t_2	sorg:City	rdfs:subClassOf	sorg:Place
t_3	sorg:containedIn	rdf:type	rdf:Property
t_4	sorg:containedIn	rdfs:domain	sorg:Place
t_5	sorg:containedIn	rdfs:range	sorg:Place
t_6	csail:libby_home_town	rdf:type	sorg:City

RDF Vocabulary Description Language: RDF Schema

- designed to introduce useful **semantics** to RDF triples
- typing**: defining *classes*, *properties*, *instances*
- relationships between types and instances: **subsumption** and **instantiation**
- constraints**: *domain* and *range* for properties
- inference rules** to entail **new, inferred** knowledge
- schemas are represented as RDF triples

	(<i>subject</i>)	(<i>predicate</i>)	(<i>object</i>)
t_1	sorg:Place	rdf:type	rdfs:Class
t_2	sorg:City	rdfs:subClassOf	sorg:Place
t_3	sorg:containedIn	rdf:type	rdf:Property
t_4	sorg:containedIn	rdfs:domain	sorg:Place
t_5	sorg:containedIn	rdfs:range	sorg:Place
t_6	csail:libby_home_town	rdf:type	sorg:City

RDF Vocabulary Description Language: RDF Schema

- designed to introduce useful **semantics** to RDF triples
- typing**: defining *classes*, *properties*, *instances*
- relationships between types and instances: **subsumption** and **instantiation**
- constraints**: *domain* and *range* for properties
- inference rules** to entail **new**, **inferred** knowledge
- schemas are represented as RDF triples

	(<i>subject</i>)	(<i>predicate</i>)	(<i>object</i>)
t_1	sorg:Place	rdf:type	rdfs:Class
t_2	sorg:City	rdfs:subClassOf	sorg:Place
t_3	sorg:containedIn	rdf:type	rdf:Property
t_4	sorg:containedIn	rdfs:domain	sorg:Place
t_5	sorg:containedIn	rdfs:range	sorg:Place
t_6	csail:libby_home_town	rdf:type	sorg:City

RDFS Inference

Used to entail new information from the one that is explicitly stated

- transitive closure along the class and property hierarchies

$$R_1 : \frac{(P_1, \text{rdfs:subPropertyOf}, P_2), (P_2, \text{rdfs:subPropertyOf}, P_3)}{(P_1, \text{rdfs:subPropertyOf}, P_3)}$$

$$R_2 : \frac{(C_1, \text{rdfs:subClassOf}, C_2), (C_2, \text{rdfs:subClassOf}, C_3)}{(C_1, \text{rdfs:subClassOf}, C_3)}$$

- transitive closure along the type and class/property hierarchies

$$R_3 : \frac{(P_1, \text{rdfs:subPropertyOf}, P_2), (r_1, P_1, r_2)}{(r_1, P_2, r_2)}$$

$$R_4 : \frac{(C_1, \text{rdfs:subClassOf}, C_2), (r_1, \text{rdf:type}, C_1)}{(r_1, \text{rdf:type}, C_2)}$$

RDFS Inference

	<i>(subject)</i>	<i>(predicate)</i>	<i>(object)</i>
t_1	sorg:Place	rdf:type	rdfs:Class
t_2	sorg:City	rdfs:subClassOf	sorg:Place
t_6	csail:libby_home_town	rdf:type	sorg:City

- transitive closure along the type and class/property hierarchies

$$R_4 : \frac{(C_1, \text{rdfs:subClassOf}, C_2), (r_1, \text{rdf:type}, C_1)}{(r_1, \text{rdf:type}, C_2)}$$

	<i>(subject)</i>	<i>(predicate)</i>	<i>(object)</i>
t_7	sorg:City	rdf:type	rdfs:Class
t_8	csail:libby_home_town	rdf:type	sorg:Place

RDFS Inference

	<i>(subject)</i>	<i>(predicate)</i>	<i>(object)</i>
t_1	sorg:Place	rdf:type	rdfs:Class
t_2	sorg:City	rdfs:subClassOf	sorg:Place
t_6	csail:libby_home_town	rdf:type	sorg:City

- transitive closure along the type and class/property hierarchies

$$R_4 : \frac{(C_1, \text{rdfs:subClassOf}, C_2), (r_1, \text{rdf:type}, C_1)}{(r_1, \text{rdf:type}, C_2)}$$

	<i>(subject)</i>	<i>(predicate)</i>	<i>(object)</i>
t_7	sorg:City	rdf:type	rdfs:Class
t_8	csail:libby_home_town	rdf:type	sorg:Place

RDFS Inference

	<i>(subject)</i>	<i>(predicate)</i>	<i>(object)</i>
t_1	sorg:Place	rdf:type	rdfs:Class
t_2	sorg:City	rdfs:subClassOf	sorg:Place
t_6	csail:libby_home_town	rdf:type	sorg:City

- transitive closure along the type and class/property hierarchies

$$R_4 : \frac{(C_1, \text{rdfs:subClassOf}, C_2), (r_1, \text{rdf:type}, C_1)}{(r_1, \text{rdf:type}, C_2)}$$

	<i>(subject)</i>	<i>(predicate)</i>	<i>(object)</i>
t_7	sorg:City	rdf:type	rdfs:Class
t_8	csail:libby_home_town	rdf:type	sorg:Place

Outline

- 1 Going from the Web of Documents to the Web of Data
- 2 Web Architecture
 - Items of Interest
 - Naming Resources: URIs
 - Resource Description Framework (RDF)
- 3 Adding Semantics to Linked Data
 - A Short Introduction
 - RDF Vocabulary Description Language (RDFS)
 - **Ontology Web Language (OWL)**
- 4 Linked Data LifeCycle
 - Extraction
 - Interlinking/Fusing
 - Storage/Querying
- 5 Conclusions

Ontology Web Language: OWL

- RDFS is useful but has limited capabilities regarding schema constraints
 - class/property hierarchies, constraining property domain and range

Ontologies define the concepts, relationships and complex constraints to describe and represent an area of knowledge

Ontology Web Language used to **create** and **reason** about ontologies:

- terminology/vocabulary used in a specific context
- constraints on terms and properties
- equivalence of vocabulary terms etc.
- rich semantics but must be computational tractable and implementable

Ontology Web Language: OWL

- RDFS is useful but has limited capabilities regarding schema constraints
 - class/property hierarchies, constraining property domain and range

Ontologies define the concepts, relationships and complex constraints to describe and represent an area of knowledge

Ontology Web Language used to **create** and **reason** about **ontologies**:

- terminology/vocabulary used in a specific context
- constraints on terms and properties
- equivalence of vocabulary terms etc.
- rich semantics but must be computational tractable and implementable

Ontology Web Language: OWL

- RDFS is useful but has limited capabilities regarding schema constraints
 - class/property hierarchies, constraining property domain and range

Ontologies define the concepts, relationships and complex constraints to describe and represent an area of knowledge

Ontology Web Language used to **create** and **reason** about **ontologies**:

- terminology/vocabulary used in a specific context
- constraints on terms and properties
- equivalence of vocabulary terms etc.
- rich semantics but must be computational tractable and implementable

Ontology Web Language

OWL expresses a small subset of First Order Logic

- defines a **structure** (classes, properties, datatypes) and axioms
- inference based on OWL is restricted in this framework **but** can be extremely complex

Ontologies can be hard

- full ontology-based application is a complex system
- hard to implement and run
- three layers of OWL are defined (in decreasing levels of complexity)

OWL sublanguages

- OWL Full: superset of RDFS, but an OWL Full ontology can be undecidable
- OWL DL (Description Logic): maximal subset of OWL Full but decidable reasoning procedures
- OWL Lite: minimal, useful subset, easily implementable

Ontology Web Language: OWL

class construction

- content enumeration
- intersection, union, complementation
- property restrictions

property restrictions

- value constraints (interpreted as restrictions on value ranges)
- cardinality constraints: maximum, minimum, exact

property characterization

- symmetric, transitive, functional, etc.
- differentiation between **datatype** and **object** properties

term equivalence

- classes are equivalent (`owl:equivalentClass`) or disjoint (`owl:disjointWith`)
- properties are equivalent (`owl:equivalentProperty`) or inverse (`owl:inverseOf`)
- URIs refer to the same (`owl:sameAs`) or distinct individual (`owl:differentFrom`)

Ontology Web Language: OWL

class construction

- content enumeration
- intersection, union, complementation
- property restrictions

property restrictions

- value constraints (interpreted as restrictions on value ranges)
- cardinality constraints: maximum, minimum, exact

property characterization

- symmetric, transitive, functional, etc.
- differentiation between **datatype** and **object** properties

term equivalence

- classes are equivalent (`owl:equivalentClass`) or disjoint (`owl:disjointWith`)
- properties are equivalent (`owl:equivalentProperty`) or inverse (`owl:inverseOf`)
- URIs refer to the same (`owl:sameAs`) or distinct individual (`owl:differentFrom`)

Ontology Web Language: OWL

class construction

- content enumeration
- intersection, union, complementation
- property restrictions

property restrictions

- value constraints (interpreted as restrictions on value ranges)
- cardinality constraints: maximum, minimum, exact

property characterization

- symmetric, transitive, functional, etc.
- differentiation between **datatype** and **object** properties

term equivalence

- classes are equivalent (`owl:equivalentClass`) or disjoint (`owl:disjointWith`)
- properties are equivalent (`owl:equivalentProperty`) or inverse (`owl:inverseOf`)
- URIs refer to the same (`owl:sameAs`) or distinct individual (`owl:differentFrom`)

Ontology Web Language: OWL

class construction

- content enumeration
- intersection, union, complementation
- property restrictions

property restrictions

- value constraints (interpreted as restrictions on value ranges)
- cardinality constraints: maximum, minimum, exact

property characterization

- symmetric, transitive, functional, etc.
- differentiation between **datatype** and **object** properties

term equivalence

- classes are equivalent (`owl:equivalentClass`) or disjoint (`owl:disjointWith`)
- properties are equivalent (`owl:equivalentProperty`) or inverse (`owl:inverseOf`)
- URIs refer to the same (`owl:sameAs`) or distinct individual (`owl:differentFrom`)

Ontology Web Language: OWL

class construction

- content enumeration
- intersection, union, complementation
- property restrictions

property restrictions

- value constraints (interpreted as restrictions on value ranges)
- cardinality constraints: maximum, minimum, exact

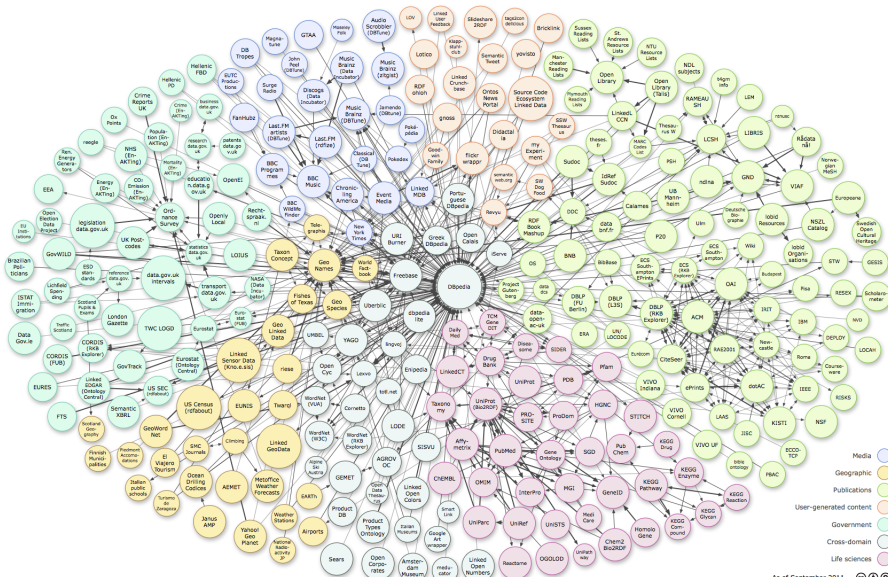
property characterization

- symmetric, transitive, functional, etc.
- differentiation between **datatype** and **object** properties

term equivalence

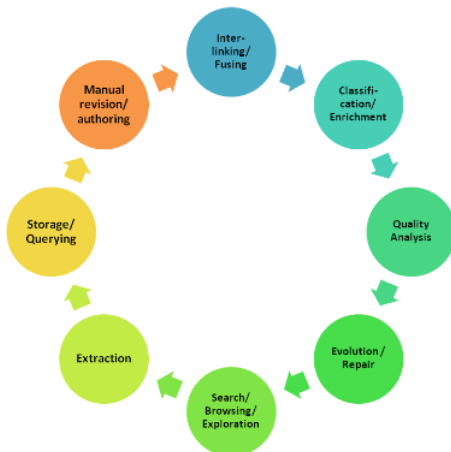
- classes are equivalent ([owl:equivalentClass](#)) or disjoint ([owl:disjointWith](#))
- properties are equivalent ([owl:equivalentProperty](#)) or inverse ([owl:inverseOf](#))
- URIs refer to the same ([owl:sameAs](#)) or distinct individual ([owl:differentFrom](#))

Linked Open Data Cloud: The Challenge



The Challenge

- Classical data management approaches assume complete control over schema, data and data generation
- But the Web is distributed and open \Rightarrow control is lacking
- Linked Data LifeCycle



Outline

- 1 Going from the Web of Documents to the Web of Data
- 2 Web Architecture
 - Items of Interest
 - Naming Resources: URIs
 - Resource Description Framework (RDF)
- 3 Adding Semantics to Linked Data
 - A Short Introduction
 - RDF Vocabulary Description Language (RDFS)
 - Ontology Web Language (OWL)
- 4 **Linked Data LifeCycle**
 - **Extraction**
 - Interlinking/Fusing
 - Storage/Querying
- 5 Conclusions

Extraction

Extraction: refers to the process of extracting data from **unstructured** and **structured** sources and representing it in the RDF data model

Extraction from Unstructured Sources: sub-disciplines of NLP

- **Named Entity Extraction:** extracting entity labels from text
- **Keyword/Keyphrase Extraction:** recognition of central topics
- **Relationship Extraction:** mining the properties which link the entities and keywords described in the input data
- **Linked Data Context: extraction of suitable URIs for the discovered entities and relationships**
 - extracted from knowledge bases such as DBPedia by comparing the name of the entity and relationship with the *label* or *accompanied comments* of the DBPedia entities

Extraction

Extraction: refers to the process of extracting data from **unstructured** and **structured** sources and representing it in the RDF data model

Extraction from Unstructured Sources: sub-disciplines of NLP

- **Named Entity Extraction:** extracting entity labels from text
- **Keyword/Keyphrase Extraction:** recognition of central topics
- **Relationship Extraction:** mining the properties which link the entities and keywords described in the input data
- **Linked Data Context: extraction of suitable URIs for the discovered entities and relationships**
 - extracted from knowledge bases such as DBPedia by comparing the name of the entity and relationship with the *label* or *accompanied comments* of the DBPedia entities

Extraction

Extraction: refers to the process of extracting data from **unstructured** and **structured** sources and representing it in the RDF data model

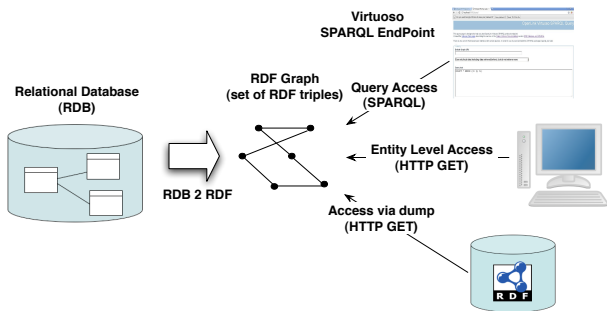
Extraction from Unstructured Sources: sub-disciplines of NLP

- **Named Entity Extraction:** extracting entity labels from text
- **Keyword/Keyphrase Extraction:** recognition of central topics
- **Relationship Extraction:** mining the properties which link the entities and keywords described in the input data
- **Linked Data Context: extraction of suitable URIs for the discovered entities and relationships**
 - extracted from knowledge bases such as DBPedia by comparing the name of the entity and relationship with the *label* or *accompanied comments* of the DBPedia entities

Extraction

Extraction from Structured Sources (relational and XML)

R2RML Relational to RDF Mapping Language



Relational to RDF Extraction

RDF to RDF Mapping Language (R2RML)

- W3C Standard Language for expressing **customized mappings** from relational databases to RDF datasets
- mappings provide the ability to view relational data as RDF data
- **R2RML mappings** define **RDF graphs**
- R2RML enables different types of mapping implementations
 - offer a virtual SPARQL endpoint over mapped relational data
 - generate RDF dumps
 - offer access through a Linked Data API

Outline

- 1 Going from the Web of Documents to the Web of Data
- 2 Web Architecture
 - Items of Interest
 - Naming Resources: URIs
 - Resource Description Framework (RDF)
- 3 Adding Semantics to Linked Data
 - A Short Introduction
 - RDF Vocabulary Description Language (RDFS)
 - Ontology Web Language (OWL)
- 4 **Linked Data LifeCycle**
 - Extraction
 - **Interlinking/Fusing**
 - Storage/Querying
- 5 Conclusions

Interlinking/Fusing

connecting things that are somehow related

- serves the Linked Open Data principle “ **Include links to other URIs, so that they can discover more things**”
- enables the paradigm change from data silos to interoperable data distributed across the Web
- cross-ontology question answering
- data integration

Link Discovery Frameworks:

- **Ontology Matching:** aims to establish links between the ontologies underlying two data sources.
- **Instance Matching:** aims to discover links between instances contained in two data sources.

Instance Matching for Linked Data

- more generic and complex than schema matching and record linkage
- not limited to finding **only equivalent entities** in knowledge bases
- aims at finding semantically related entities
- establishing links between them with formal properties (transitivity, symmetry) used by reasoners
 - to infer new knowledge
 - perform data integration tasks
 - answer cross-ontology queries

Link Discovery Frameworks

- domain-specific
 - RKBExplorer (academic domain): <http://www.rkbexplorer.com>
 - GNAT (music domain)
- universal
 - RDF-AI : <http://code.google.com/p/rdfai/>
 - SILK: <http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/>
 - LIMES: <http://aksw.org/Projects/LIMES.html>

LIMES and SILK focus on time-efficient instance matching techniques

Outline

- 1 Going from the Web of Documents to the Web of Data
- 2 Web Architecture
 - Items of Interest
 - Naming Resources: URIs
 - Resource Description Framework (RDF)
- 3 Adding Semantics to Linked Data
 - A Short Introduction
 - RDF Vocabulary Description Language (RDFS)
 - Ontology Web Language (OWL)
- 4 Linked Data LifeCycle
 - Extraction
 - Interlinking/Fusing
 - **Storage/Querying**
- 5 Conclusions

Querying Linked Data

SPARQL: W3C Standard Language for Querying Linked Data

Building Block is the **SPARQL triple pattern**: an RDF triple with variables

- an element of the set: $(\underline{\mathbb{V}} \cup \mathbb{U}) \times (\underline{\mathbb{V}} \cup \mathbb{U}) \times (\underline{\mathbb{V}} \cup \mathbb{U} \cup \mathbb{L})$
 \mathbb{U} = set of URIs, \mathbb{L} = set of Literals, $\underline{\mathbb{V}}$ = set of Variables

Main idea: Pattern Matching

- describe subgraphs of the queried RDF graph
- subgraphs that match the query yield a set of *mappings*: assignment of *variables* to *URIs* and *literals*

SPARQL: Querying Linked Data

Intuitively a triple pattern denotes the triples in an RDF graph that are of a specific form

- $tp_1 = (?j, \mathbf{rdf:type}, ?y)$: matches all triples with predicate **rdf:type**

Set of RDF Triples

	S	P	O
t_1	Starbucks	type	Cafe
t_2	Starbucks	branchLoc	53 rd St
t_3	MoMA	address	53 rd St

Mappings

	$?j$	$?y$
μ_1	Starbucks	Cafe

SPARQL Queries

Types of Queries

- **SELECT**: return a **set of variable mappings**
- **CONSTRUCT**: return an **RDF graph**
- **ASK**: boolean queries

SQL-like syntax: `SELECT ?v1, ?v2, ... WHERE GP`, with ?v₁, ?v₂ variables

SPARQL graph patterns (GP) are formulated using the **join** (AND), **union** (UNION) and **optional** (OPTIONAL) operators on **triple** patterns with **filter** conditions (FILTER)

SPARQL semantics: expressed through an algebra that operates on **sets of mappings**

- unary operators σ for FILTER, π for SELECT
- binary operators, \cup for UNION, \bowtie for AND and \Join for OPTIONAL

Compatible Mappings: mappings are compatible if they agree on their common variables

SPARQL Queries

$\Omega = \text{evaluation of } (?x, ?y, ?z) \text{ over } T$

?x	?y	?z
Starbucks	type	Cafe
Starbucks	branchLoc	53 rd St
MoMA	address	53 rd St

(a)

$\Omega_2 = \pi_{?y, ?z}(\sigma_{?x = \text{"Starbucks"}}(\Omega))$

	?y	?z
$\mu_3 :$	type	Cafe
$\mu_4 :$	branchLoc	53 rd St

(c)

$(?x, ?y, ?z)$

$\Omega_1 \cup \Omega_2$

	?x	?y	?z
$\mu_5 :$	Starbucks	branchLoc	—
$\mu_6 :$	MoMA	address	—
$\mu_7 :$	—	type	Cafe
$\mu_8 :$	—	branchLoc	53 rd St

(e) $\{ (?x, ?y, 53^{\text{rd}} \text{ St}) \} \text{ UNION } (\{ ?x, ?y, ?z \})$

$\Omega_1 = \pi_{?x, ?y}(\sigma_{?z = \text{"53rd St"}}(\Omega))$

	?x	?y
$\mu_1 :$	Starbucks	branchLoc
$\mu_2 :$	MoMA	address

(b)

$(?x, ?y, 53^{\text{rd}} \text{ St})$

$\Omega_3 = \pi_{?x, ?y, ?z}(\Omega_1 \bowtie \Omega_2)$

	?x	?y	?z
$\mu_9 :$	Starbucks	branchLoc	53 rd St

(d)

$\{ (?x, ?y, 53^{\text{rd}} \text{ St}) \} . (\{ ?x, ?y, ?z \})$

SPARQL

Additional SPARQL 1.0 Features

- duplicate elimination **DISTINCT**
- ordering results (**ORDER BY**) with an optional **LIMIT** clause

SPARQL 1.0 Missing: aggregates, subqueries of any type, **inference**

SPARQL 1.1: Extends SPARQL 1.0

- aggregations (**COUNT**, **AVG** ...)
- Subqueries
- Negation (already expressed in SPARQL 1.0 with a combination of **OPTIONAL** and **BOUND**)
- **Regular Path Expressions**
- **Updates**
- **Federation**: supports queries that merge data distributed across the Web.

Storing Linked Data

Logical Storage Schemas for RDF data

schema agnostic: triples are stored in a large *triple table* where the attributes are *subject*, *predicate* and *object*

Triple Table

<i>subject</i>	<i>predicate</i>	<i>object</i>
Starbucks	type	Cafe
Starbucks	branchLoc	53 rd St
MoMA	address	53 rd St

schema aware: one table is created per property with *subject* and *object* attributes

address	
<i>subject</i>	<i>object</i>
MoMA	53 rd St

Cafe	
<i>subject</i>	
Starbucks	

branchLoc	
<i>subject</i>	<i>object</i>
Starbucks	53 rd St

Storing Linked Data

hybrid: combines elements of the previous approaches.

Triples of properties with range string

<i>subject</i>	<i>predicate</i>	<i>object</i>
Starbucks	branchLoc	53 rd St
MoMA	address	53 rd St

Cafe
<i>subject</i>
Starbucks

Triples of properties with range date

<i>subject</i>	<i>predicate</i>	<i>object</i>
Starbucks	founded	1990

Storing Linked Data

Updates

- **Schema Agnostic:** *addition/deletion of triples* in the triple table
- **Hybrid and Schema-aware:**
 - schema updates \Rightarrow *addition/deletion of property/class tables*
 - data updates \Rightarrow *addition/deletion of tuples* in property/class tables

Type Information

- **Schema Agnostic:** typing information is lost since everything is a string
- **Hybrid:** *explicit*
- **Schema-aware:** *implicit*

Query Processing

- **Schema Agnostic:**
 - algebraic plan obtained for a query involves a large number of **self joins**
 - queries are favorable when the predicate is a variable
- **Hybrid Approach and Schema-aware:**
 - algebraic plan contains operations over the appropriate property/class tables (more in the spirit of existing relational schemas)
 - saves many self-joins over triple tables
 - if the predicate is a variable, then one query per property/class must be expressed

Storing Linked Data

Updates

- **Schema Agnostic:** *addition/deletion of triples* in the triple table
- **Hybrid and Schema-aware:**
 - schema updates \Rightarrow *addition/deletion of property/class tables*
 - data updates \Rightarrow *addition/deletion of tuples* in property/class tables

Type Information

- **Schema Agnostic:** typing information is lost since everything is a string
- **Hybrid:** *explicit*
- **Schema-aware:** *implicit*

Query Processing

- **Schema Agnostic:**
 - algebraic plan obtained for a query involves a large number of **self joins**
 - queries are favorable when the predicate is a variable
- **Hybrid Approach and Schema-aware:**
 - algebraic plan contains operations over the appropriate property/class tables (more in the spirit of existing relational schemas)
 - saves many self-joins over triple tables
 - if the predicate is a variable, then one query per property/class must be expressed

Storing Linked Data

Updates

- **Schema Agnostic:** *addition/deletion of triples* in the triple table
- **Hybrid and Schema-aware:**
 - schema updates \Rightarrow *addition/deletion of property/class tables*
 - data updates \Rightarrow *addition/deletion of tuples* in property/class tables

Type Information

- **Schema Agnostic:** typing information is lost since everything is a string
- **Hybrid:** *explicit*
- **Schema-aware:** *implicit*

Query Processing

- **Schema Agnostic:**
 - algebraic plan obtained for a query involves a large number of **self joins**
 - queries are favorable when the predicate is a variable
- **Hybrid Approach and Schema-aware:**
 - algebraic plan contains operations over the appropriate property/class tables (more in the spirit of existing relational schemas)
 - saves many self-joins over triple tables
 - if the predicate is a variable, then one query per property/class must be expressed

Storing Linked Data

Updates

- **Schema Agnostic:** *addition/deletion of triples* in the triple table
- **Hybrid and Schema-aware:**
 - schema updates \Rightarrow *addition/deletion of property/class tables*
 - data updates \Rightarrow *addition/deletion of tuples* in property/class tables

Type Information

- **Schema Agnostic:** typing information is lost since everything is a string
- **Hybrid:** *explicit*
- **Schema-aware:** *implicit*

Query Processing

- **Schema Agnostic:**
 - algebraic plan obtained for a query involves a large number of **self joins**
 - queries are favorable when the predicate is a variable
- **Hybrid Approach and Schema-aware:**
 - algebraic plan contains operations over the appropriate property/class tables (more in the spirit of existing relational schemas)
 - saves many self-joins over triple tables
 - if the predicate is a variable, then one query per property/class must be expressed

Storing Linked Data

Updates

- **Schema Agnostic:** *addition/deletion of triples* in the triple table
- **Hybrid and Schema-aware:**
 - schema updates \Rightarrow *addition/deletion of property/class tables*
 - data updates \Rightarrow *addition/deletion of tuples* in property/class tables

Type Information

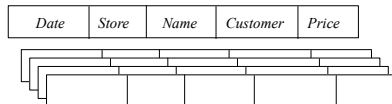
- **Schema Agnostic:** typing information is lost since everything is a string
- **Hybrid:** *explicit*
- **Schema-aware:** *implicit*

Query Processing

- **Schema Agnostic:**
 - algebraic plan obtained for a query involves a large number of **self joins**
 - queries are favorable when the predicate is a variable
- **Hybrid Approach and Schema-aware:**
 - algebraic plan contains operations over the appropriate property/class tables (more in the spirit of existing relational schemas)
 - saves many self-joins over triple tables
 - if the predicate is a variable, then one query per property/class must be expressed

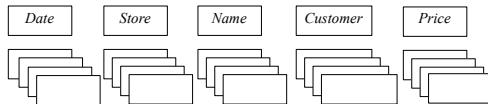
Physical Storage Schema for RDF Data

Row store: store relations as complete rows (PostgreSQL, MySQL, Oracle RDF)



- fully compliant solution for the *schema agnostic* approach
- row-store solutions that follow the schema agnostic approach, outperform those that follow the schema-aware approach

Column store: each attribute of a relational table is stored in a column (MonetDB)

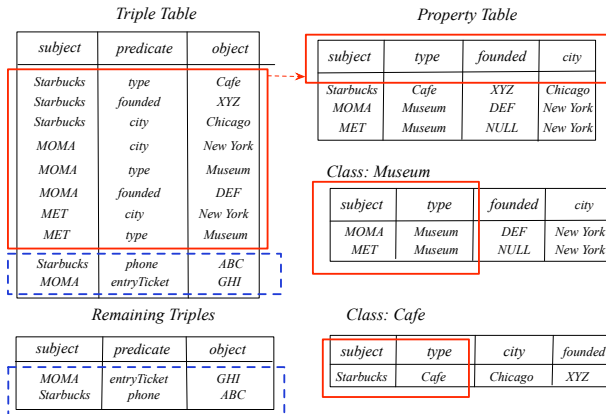


- fully compliant solution for *vertical partitioning*

Physical Storage Schema for RDF data

Property Tables & Property-Class Tables

- Clustered Property Tables: contain clusters of properties that tend to be defined together
- Property-Class: exploits the type property of subjects to cluster similar sets of subjects together



Dictionary & Ordered Relations

- ✓ to avoid processing long strings URLs and Literals are mapped to unique identifiers
- ✓ regular size, much easier to handle and compress
- ✓ the dictionary is small and can be kept in main memory
- — additional join should be performed when retrieving the results
- — filter conditions can be more expressive (include translating the identifiers into values and back)

Physical Storage Schema for RDF Data

Native RDF-engines: Sesame, Jena, Virtuoso, RDF-3X, Hexastore, ...

Indexes that support different **SPARQL query patterns** aim at

- the efficient retrieval of triples during query evaluation
- the computation of the cost of joins

Point Query: graph patterns consists of a *single triple pattern*

SELECT ?y WHERE { s_0 p_0 ?x }

Chain Query: graph pattern is a sequence of triple patterns

SELECT ?y WHERE { s_0 p_0 ?x } . { ?x p_1 ?y }

Star Query:

SELECT ?y, ?z WHERE { ?x p_0 ?y } . { ?x p_1 ?z } . { ?x p_2 ?z }

Hierarchical Queries: special case of *chain queries* that consider the traversal of RDFS **rdfs:subClassOf** and **rdfs:subPropertyOf** relations

Indexes

Point Queries

- **schema agnostic approach** indexes on all possible permutations of the triple table that stores triples of the form (*subject*, *predicate*, *object*)
 - triples in each index are sorted **lexicographically**
 - (subject, object, property) for index spo,
 - (object, subject, property) for index osp
 - indexes are implemented as B+-trees
- ✓ benefit of having all possible indexes is that any triple pattern can be answered with a single index scan
- ✓ facilitate the use of sort-merge join operator for evaluating joins
- **vertical partitioning**
 - indexes are built on the *subject* and *object* columns of property tables [SAB+05]
 - sextuple indexing scheme with *subject* and *object*-headed indexes [WKB08]

Indexes

Hierarchical Queries

- supported by indexes implemented as B+ trees [CPS+07]
- indexes on subgraphs [BB07] defined by the **rdfs:subClassOf** and **rdfs:subPropertyOf** relations

Star Queries & Chain Queries: supported by **aggregated indexes** [NW08, NW09] to compute the *selectivity* of a set of different access patterns

- for every combination of values for the *subject*, *predicate* and *object* values that store the *number* of triples that match this combination
- indexes that store the number of triples that can be joined with triples that contain a specific combination of RDF terms \Rightarrow approximate the cardinality of join results improve the performance of joins
- compute the **frequent** paths in the RDF dataset to estimate the cost of a join in the case of path and chain queries and use the size of the intermediate results to calculate the cost of a join.

Where does existing Database Technology fit?

Existing Relational Stores are problematic for querying the Web of Data

- due to the fragmentation of information, absence of schema and constraints for RDF data leads to query plans that are hard to optimize
 - **schema agnostic** :a large number of self joins over a large triple table is required
 - **vertical partitioning**: multiple joins over different tables
- relational optimizers are not designed for processing RDF data:
 - compute the cost of scans but not the join hit ratio(s)
 - the subject-subject and subject-object join hit ratio will have the same estimate since correlations between triple components are not considered
- *correlated cost estimation over many selections and self joins*
 - **the rule** and not the exception for RDF query processing

Where does existing Database Technology fit?

SQL-based engines: SW-store, Oracle RDF, Sesame, Virtuoso RDF

- SPARQL queries are translated into SQL queries and are evaluated by the underlying engine
- rely optimizations related to relational data

Native Engines: Hexastore, Yars2, RDF-3X

- rely mostly on **merge joins** over sorted index lists over **hash joins** when possible
- **merge joins** operate on sorted inputs
 - sequentially scan both inputs ✓
 - immediately join matching triples ✓
 - skip over parts without matches ✓
 - supports pipelining ✓
- **hash joins** operate on unsorted data
 - needs to touch every triple —
 - breaks pipelining —

SPARQL Query Processing

Issues

- decide among the multiple (subject, predicate, object) **indexes** the one to evaluate one triple pattern (*spo*, *sop*, etc.)
- decide the *join order* and *join algorithms per join variable* (merge or hash joins)

Approaches

- **Cost-based** based on dynamic programming [NW08, NW09]
- **Heuristic-based** do not statistics but are based on data characteristics [TSF+12]

Conclusions

addressed in this talk

- From Web of Documents to Web of Data \sim Linked Data
- Linked Data Principles: URIs, RDF
- Adding Semantics to data: defining schemas & ontologies
- Linked Data Life Cycle: Extraction, Intelinking, Storage and Querying

not addressed in this talk

- LifeCycle processes
 - Manual revision/authoring: principles for creating RDF datasets
 - Classification/Enrichment: how to add semantics to existing datasets
 - Quality Analysis (Provenance Tracking, Vocabulary Mapping)
- Storage and Querying Linked Data in a distributed setting

References

- [BB07] L. Baolin and H. Bo. HPRD: A High Performance RDF Database. In *Network and Parallel Computing*, 2007.
- [CPS+07] V. Christophides, D. Plexousakis, M. Scholl, and S. Tourtounis. On labeling schemes for the Semantic Web. In *ISWC*, 2003.
- [NW08] T. Neumann and G. Weikum. RDF-3X: a RISC-style engine for RDF. *PVLDB*, 1(1), 2008.
- [NW09] T. Neumann and G. Weikum. Scalable join processing on very large rdf graphs. In *SIGMOD*, June 2009.
- [SAB+05] M. Stonebraker, D. Abadi, A. Batkin, X. Chen, M. Cherniak, M. Ferreira, E. Lau, A. Lin, S. Madden, P. E. O’Neil, A. Rasin, N. Tran, and S. Zdonik. C-Store: A Column Oriented DBMS. In *VLDB*, 2005.
- [TSF+12] P. Tsalamanis, L. Sidirourgos, I. Fundulaki, P. Boncz and V. Christophides. Heuristics-based Query Optimisation for SPARQL. In *EDBT*, 2012.
- [WKB08] C. Weiss, P. Karras, and A. Bernstein. Hexastore: sextuple indexing for semantic web data management. *PVLDB*, 1(1), 2008.

References

Material for the slides was used from :

- Tom Heath and Christian Bizer. Linked Data: Evolving the Web into a Global Data Space (1st edition). *Synthesis Lectures on the Semantic Web: Theory and Technology, 1:1*, 1-136. Morgan & Claypool, 2011.
- Christian Bizer, Richard Cyganiak and Tom Heath. How to Publish Linked Data on the Web (Tutorial). Available at <http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/LinkedDataTutorial/>.
- Tim Berners-Lee. Design Issues: Linked Data. Available at: <http://linkeddata.org/guides-and-tutorials>.
- Andreas Harth, Katja Hose and Ralf Schenkel. Database Techniques for Linked Data Management. In *SIGMOD*, 2012.