Linked Open Data

Dan Vodislav

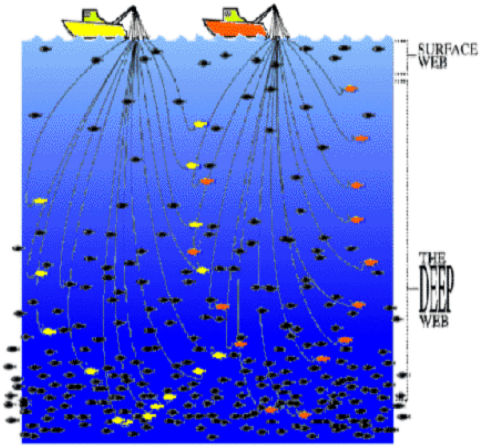
ETIS, Université de Cergy-Pontoise

Evolution of the Web

- Web 1.0
  - Unstructured content (text/HTML)
  - Passive consumers
- Web 2.0
  - More structured content (XML, JSON)
  - Active consumers
  - Some big web sites managing huge volumes of specialized content types

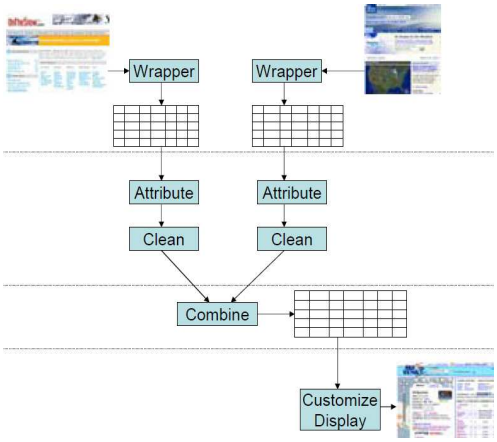
Access to data

- HTML documents vs. Databases
  - Dynamic generation of HTML documents
  - Web forms
  - Data = hidden web >> surface web
    - In 2001, 60 hidden web sites contained together more than 40 times the size of the surface web
- Problems
  - The "meaning" of the data exported on the web (identifiers, attributes) is lost
  - Data quality, coherence



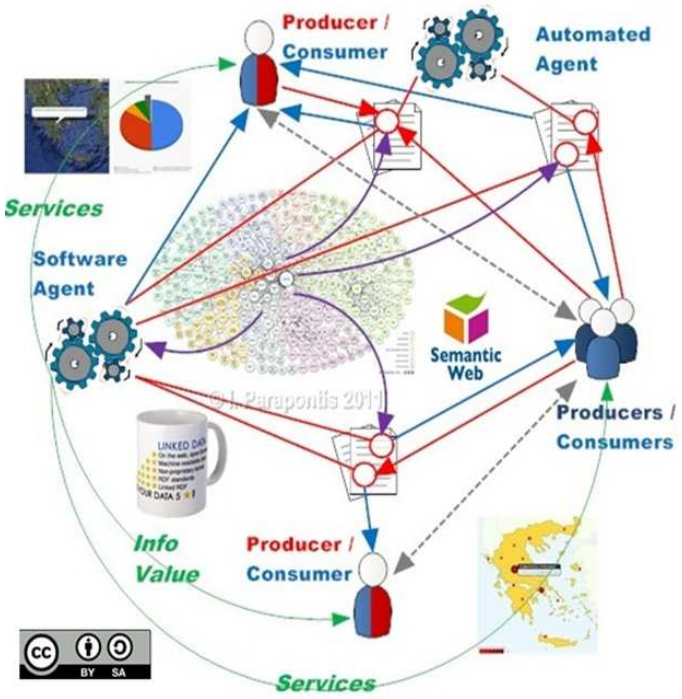
Exploiting web data

- Surface Web → search engines
- Data / hidden web → mashups
- Mashup
  - Simple integration of web data
    - Data « instances » (entities)
    - Union, no joins
  - Service-oriented approaches
  - Mashup integration steps
    - Data extraction (wrappers)
    - Calibration / cleaning
    - Integration
    - Visualization



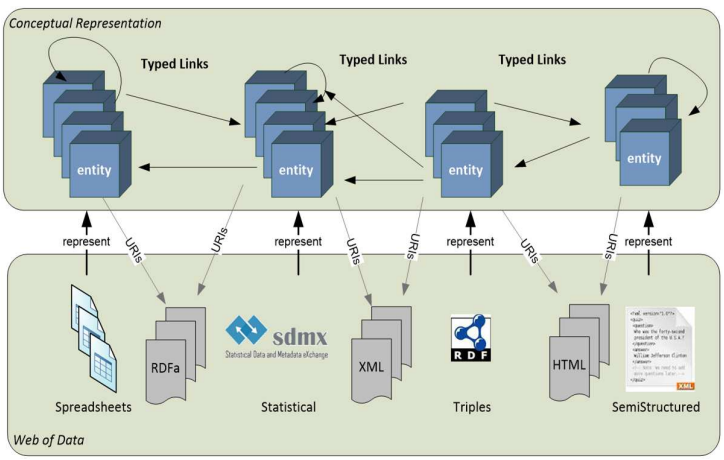
Web 3.0

- Semantic Web
  - Web 2.0 (diversity of contents, producers/ consumers) + **semantics**
  - Towards an automatic processing of web data: programs, services, reasoning
- How?
  - First step: Web of Data



Web of Data

- Web of objects (entities) described by web data
  - Descriptions of real world objects
  - Links (relations) between these objects
  - ***Global Dataspace*** gathering all this data



Open Data

- Publicly available data
  - Data already available on the web + data made public by various institutions
  - « Open Data » movement supported by governmental initiatives
- Various categories of data formats

★ Available on the web (any format), but with an Open Data license

★★ Additionally, structured format (e.g. Excel vs. image of a table)

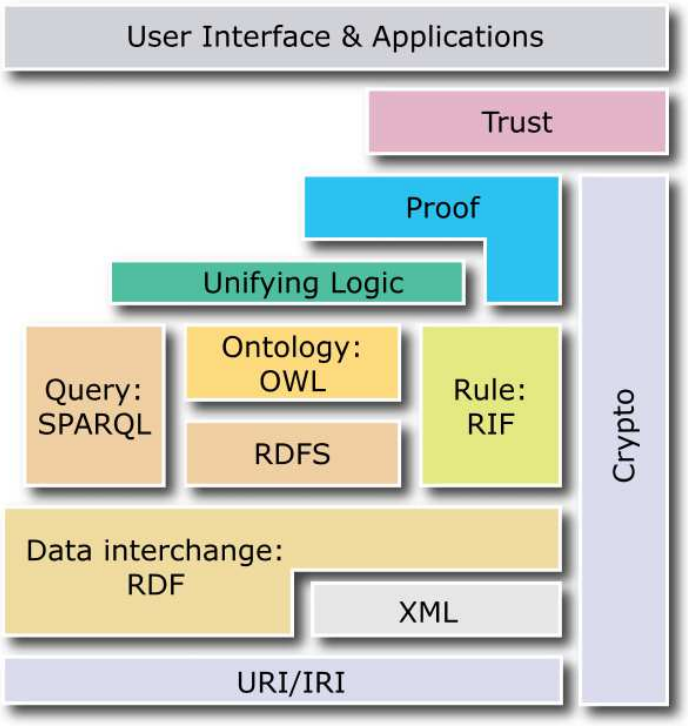
★★★ Additionally, non-proprietary format (e.g. CSV instead of Excel)

★★★★ Additionally, using open standards of the W3C (RDF and SPARQL) to identify and make accessible objects through dereferenceable URIs

★★★★★ Additionally, providing links to equivalent elements in other sources

Format	Recommendation (scale from 0 to 5)
csv	★★★
xls	★
pdf	★
doc	★
xml	★★★★
rdf	★★★★★
shp	★★★
ods	★★
tiff	★
jpeg	★
json	★★★
txt	★
html	★★

Semantic Web levels

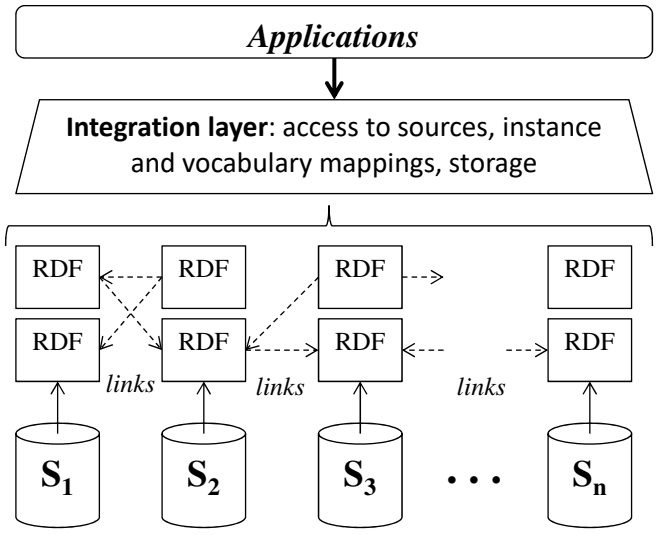


Linked Open Data (LOD)

- ★★★★★ part of the Web of Data
  - RDF data published by different sources
  - Links (also expressed in RDF) between RDF data of these sources
- The four principles of the LOD (Tim Berners-Lee)
  - Use URIs to name (identify) objects (resources)
  - Use HTTP URLs as URIs, to make resources accessible on the Web (dereferenceables)
  - When such a URI is accessed, something useful is found (in principle in RDF format)
  - Links to other resources must be included, to be able to discover new information

LOD Architecture

- Dataspace: three kinds of "actors" relative to each source
  - The publisher
  - The publishers of the other sources
  - The data consumers

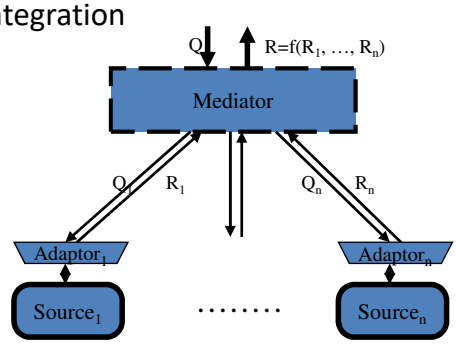


Dataspace

- Specific architecture for data integration
  - No global schema defined
  - Each publisher uses his own specific structure/schema for data and specifies the mapping to other data sources
  - Progressive improvement of the quality of the global system
  - The quality is proportional to the integration effort

→ adapted to very large scale and very dynamic data integration

- In comparison: mediator architecture
  - Big effort to define the global schema and the mappings to all the sources
  - Big effort to maintain the schema and the mappings
  - The quality is guaranteed



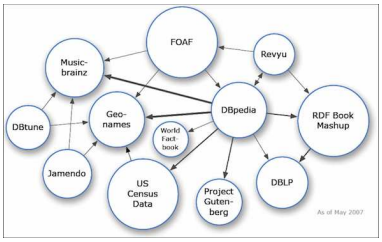
- Linked Open Data: RDF dataspace
  - Integration effort: the links between sources
  - Two kind of links: for *instance identity* ("sameAs") and for *vocabulary / concepts* (equivalence or subsumption of classes/properties)

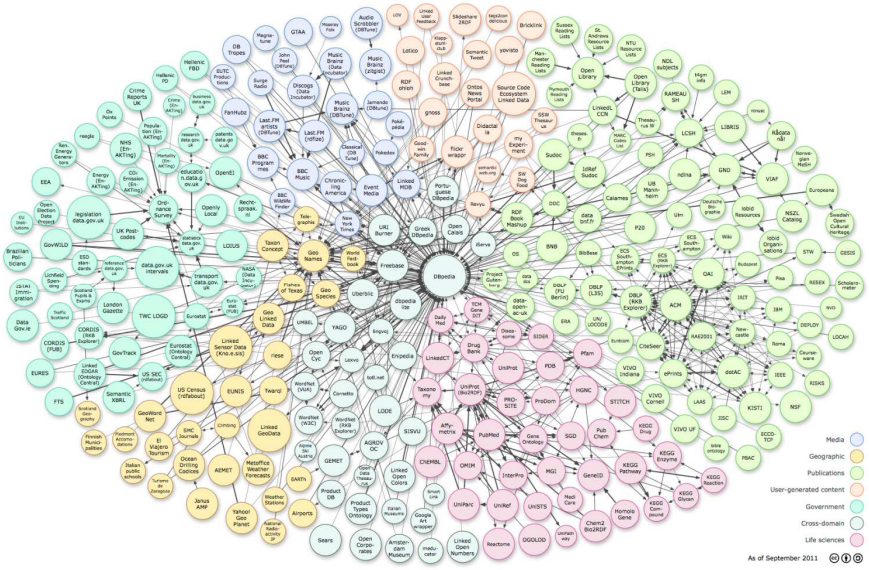
Who makes the integration effort?

- *Shared effort* between the publisher, other publishers and the consumers
  - The publisher of source S
    - Choses the vocabularies (new or reused)
    - Publishes data in RDF
    - Publishes identity links to other sources
    - Publishes vocabulary links to other vocabularies
  - The other publishers
    - Publish identity links to data in S
    - Publish vocabulary links to vocabularies in S
  - The consumer = programmer of the data integration application
    - Defines the way of accessing LOD data in the different sources
    - Defines or deduces identity links between sources (with specific tools)
    - Defines or deduces vocabulary links between sources (with specific tools)
    - Cleans the data
    - Integrates data (RDF warehouse)
- In comparison, with a mediator architecture: the consumer makes (almost) everything

The LOD “cloud”

- The evolution of the LOD Web of Data
  - May 2007: 500 million RDF triples, 120 000 RDF links
  - September 2011: 31.6 billion RDF triples, 503 million links
  - April 2015: the number of sources is multiplied by 4 compared to 2011





LOD sources

- What kind of sources compose the LOD cloud ?
  - RDF sources respecting the LOD constraints
  - At least 1000 triples et 50 links to other sources in the cloud
  - Access in HTML+RDFa, or RDF file, or SPARQL endpoint
- Who publishes open data ?
  - Governments: European Union, USA, France (*data.gouv.fr*), …
  - Cultural institutions: national libraries, museums, archives
  - Other institutions
  - See: http://linkeddata.org/, http://www.w3.org/wiki/SparqlEndpoints
- At the heart of the LOD cloud: ***DBpedia*** (http://dbpedia.org/)
  - Advantage of DBpedia: covers a large set of concepts that other sources can refer

DBpedia

- LOD source obtained from Wikipedia
  - Use of the “info boxes” on the Wikipedia pages
  - Use of the Wikipedia categories
- English version of DBpedia
  - 4.58 million entities, out of which 4.22 million are instances of the DBpedia ontology
  - 580 million of RDF triples
  - Other languages: DBpedia versions in 125 languages with links between entities in the different languages
    - Altogether: 3 billion RDF triples
- DBpedia Ontology
  - 685 classes
  - 2795 properties
- Access: SPARQL endpoint, download, other tools

RDF: Resource Description Framework

- Language of the Semantic Web
  - Web resource description: web pages, images, videos, …
  - Describes the properties of the resources or the relations between resources
  - Several possible syntaxes
  - RDF Schema (RDFS): concepts, classes, schemas → ontologies
- RDF model levels
  - Physical level : triples / statements
    - Base types : resources, properties, statements
    - Complex types: collections, lists
  - Schemas (RDFS): classes, property types
    - OWL: more advanced elements

RDF triples

- Statement : triple (S, P, V)
  - Knowledge “atom”
  - Meaning: subject **S** has for property **P** the value **V**
    - Or (**Subject**, **Predicate**, **Object**)
- Example
  - `(ETISPage, author, Michel)`
  - `(ETIS, WebPage, ETISPage)`
  - `(Michel, WebPage, PageMichel)`
  - `(ETIS, director, Mathias)`
  - `(Michel, name, "Michel Jordan")`
- Comparison with the relational model : `(ETIS, director, Mathias)`

Laboratory

identifier	director	WebPage	…
…	…	…	…
ETIS	Mathias	ETISPage	…
…	…	…	

People

identifier	name	…
Michel	Michel Jordan	…
Mathias	Mathias Quoy	…
…		

Resources and URI

- Resources and properties are identified by URIs
  - **S**, **P** and **V** are given by URIs
  - **V** may be also a literal value
- Remark: URI ≠ URL, URI not necessary a real web address
- Example (various possible notations)
  - `(http://www-etis.ensea.fr, dc:creator, #Michel)`
  - `(#ETIS, #WebPage, http://www-etis.ensea.fr)`
  - `(#Michel, #WebPage, http://perso-etis.ensea.fr/~jordan)`
  - `(#ETIS, #director, #Mathias)`
  - `(#Michel, #name, "Michel Jordan")`
- Local URIs : `#Michel, #ETIS, #WebPage, #director, #Mathias, #name`
- External URIs : `http://www-etis.ensea.fr, dc:creator, http://perso-etis.ensea.fr/~jordan`
- Literal values: `"Michel Jordan"`
  - One may specify the type (`"32"^^xsd:integer`), or the language (`"Eiffel Tower"@en`)

Namespace usage

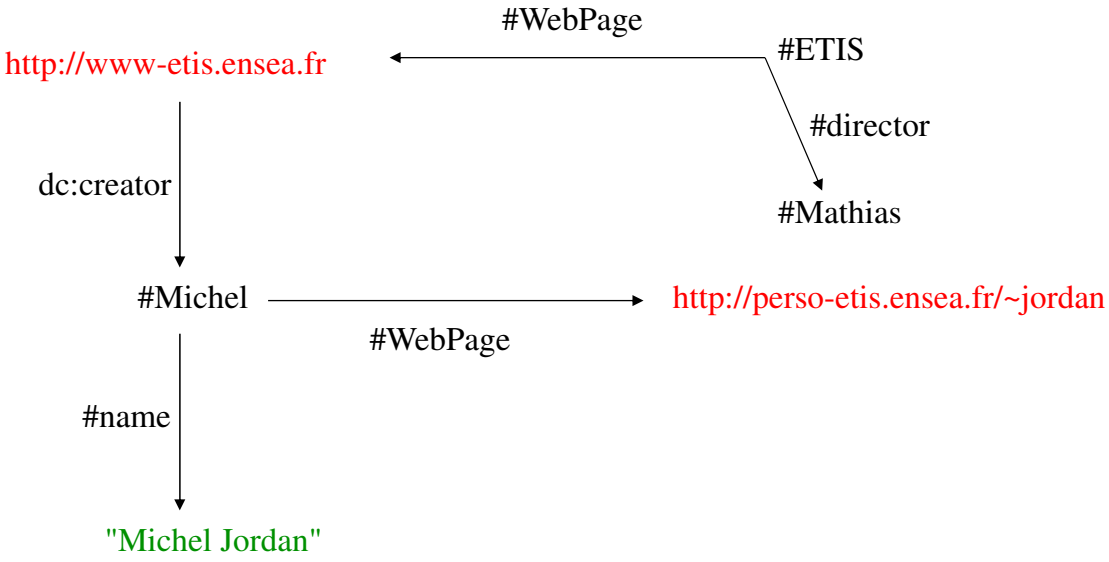
- Local resources: specific, local namespace
  - Groups and identifies local resource names: (`#ETIS, #WebPage, …`)
    `xmlns:mine="http://myapp.myorg.com"`
  - `#ETIS` means `http://myapp.myorg.com#ETIS`
  - Alternate notation:
    `mine:ETIS` or `http://myapp.myorg.com/ETIS`
- External resources: reference to specific namespaces
  - Goal: use "standard" resources/properties
  - E.g. Dublin Core: standard concepts about documents
    `xmlns:dc="http://purl.org/dc/elements/1.1"`
    `dc:creator` = the creator of a document/resource
- For the data types: XML Schema namespace
  - `xmlns:xsd="http://www.w3.org/2001/XMLSchema"`

Some common vocabularies

- Dublin Core: description of documents/resources
  - Content: *title, subject, description, source, language, relation, coverage*
  - Intellectual property: *creator, contributor, publisher, rights*
  - Other: *date, type, format, identifier*
  - Namespace: http://purl.org/dc/elements/1.1/
- FOAF (Friend of a Friend): description of persons
  - Classes (*Person, Group, Organization, Document, Image,* …)
  - Properties for Person: *name, firstName, lastName, knows, homepage, …*
  - Namespace: http://xmlns.com/foaf/0.1/
- SKOS (Simple Knowledge Organization System): taxonomies
  - *Concept* class
  - Properties: *broader, narrower, related, prefLabel, altLabel, …*
  - Namespace: http://www.w3.org/2004/02/skos/core#

RDF graph

- Triple = two nodes (S, V) + the oriented edge (P) that connects them
- Set of triples → oriented graph

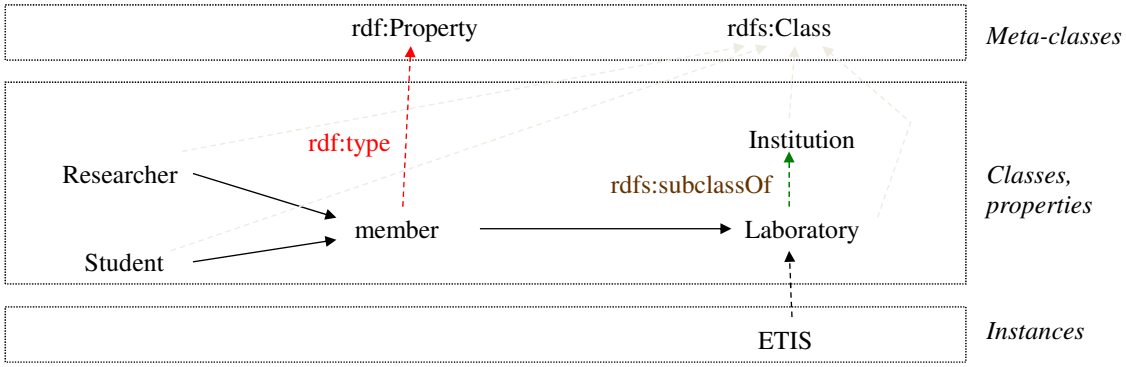


Predefined elements

- `rdf` or `rdfs` namespaces
  - `xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"`
  - `xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"`
- For the types
  - Property `rdf:type`
  - Base types: `rdf:Resource`, `rdf:Property`, `rdf:Statement`
- For a statement (triple)
  - `rdf:subject`, `rdf:predicate`, `rdf:object` refer to the three elements of the triple
- Other examples further

RDF Schema

- Description of *classes* and *property types*
  - Classes: *rdfs:Class*, *rdfs:subclassOf*
  - Properties: *rdfs:subpropertyOf* , *rdfs:domain, rdfs:range*




```
(#Institution, rdf:type, rdfs:Class)
(#Laboratory, rdf:type, rdfs:Class)
(#Laboratory, rdfs:subclassOf, #Institution)
(#member, rdf:type, rdf:Property)
(#member, rdfs:domain, #Student)
(#member, rdfs:domain, #Researcher)
(#member, rdfs:range, #Institution)
(#ETIS, rdf:type, #Laboratory)
```

OWL

- OWL (Web Ontology Language) = extension of RDFS
  - Can express more powerful constraints
  - Reasoning possibilities
- RDF/RDFS
  - Only constraints: *rdfs:subClassOf* and *rdfs:subPropertyOf*
  - Class definition: by reference (URI) + declaration of instances
    - Open world assumption: a missing info is not necessarily false

    → The set of instances of a class is not known
  - Limited possibilities for reasoning

Class definition with OWL

- Several ways of defining a class
  - Through a reference (URI)
  - By enumerating the instances
  - Through its properties
  - As a union, intersection, difference of other classes
- Example of enumeration of instances

```
<owl:Class rdf:ID="mycontinents">
    <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Eurasia"/>
        <owl:Thing rdf:about="#Africa"/>
        <owl:Thing rdf:about="#NorthAmerica"/>
        <owl:Thing rdf:about="#SouthAmerica"/>
        <owl:Thing rdf:about="#Australia"/>
        <owl:Thing rdf:about="#Antarctica"/>
    </owl:oneOf>
</owl:Class>
```

Class definition with OWL (cont'd)

- Through the properties
  - Property values: *owl:allValuesFrom*, *owl:someValuesFrom, owl:hasValue*
  - Cardinality : *owl:maxCardinality*, *owl:minCardinality*, *owl:Cardinality*

Example: class whose elements have for property *member* only values of type *Student*

```
<owl:Restriction>
    <owl:onProperty rdf:resource="#member" />
    <owl:allValuesFrom rdf:resource="#Student" />
</owl:Restriction>
```

- By computation : *owl:intersectionOf, owl:unionOf, owl:complementOf*

```
<owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="mycontinents">
        <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
                <owl:Thing rdf:about="#Europe" />
                <owl:Thing rdf:about="#Africa" />
                <owl:Thing rdf:about="#America" />
            </owl:oneOf>
        </owl:Class>
    </owl:intersectionOf>
</owl:Class>
```

Relations between classes in OWL

- *rdfs:subClassOf*
  - The instances of a class belong to the other class also
- *owl:equivalentClass*
  - Classes having the same instances, but not addressing the same concept

  <footballTeam owl:equivalentClass us:soccerTeam />
- *owl:disjointWith*
  - Two disjoint classes

Definition of OWL properties

- RDF Schema : *rdfs:subPropertyOf*, *rdfs:domain* et *rdfs:range*
- Relations between properties
  - *owl:equivalentProperty*: the two properties have the same values, but are not identical
  - *owl:inverseOf* : a property is the inverse of the other one

```
<owl:ObjectProperty rdf:ID="child">
    <owl:inverseOf rdf:resource="#parent"/>
</owl:ObjectProperty>
```

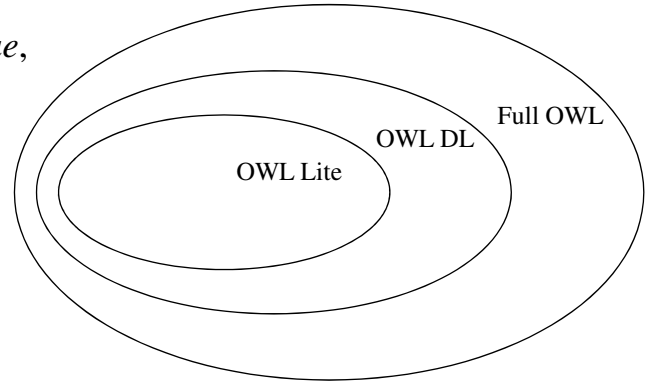
- Cardinality constraints
  - Mono-valuated properties:
    `<owl:FunctionalProperty rdf:about="#spouse" />`
  - Inverse mono-valuated properties:

```
<owl:InverseFunctionalProperty rdf:ID="biologicalMother">
    <rdfs:domain rdf:resource="#woman"/>
    <rdfs:range rdf:resource="#person"/>
</owl:InverseFunctionalProperty>
```

- Logical constraints
  - *owl:SymmetricProperty* (e.g. spouse)
  - *owl:TransitiveProperty* (e.g. ancestor)

Hierarchies of OWL languages

- Full OWL: RDF/RDFS + new OWL operators
  - Powerful, but undecidable reasoning
- OWL DL (Description Logic)
  - Restrictions on Full OWL that insure a decidable reasoning
    - E.g. a class or a property cannot be an instance
- OWL Lite
  - Restrictions on OWL DL that insure an efficient reasoning
  - E.g. eliminating *owl:unionOf*, *owl:complementOf*, *owl:hasValue*, *owl:disjointWith*, …



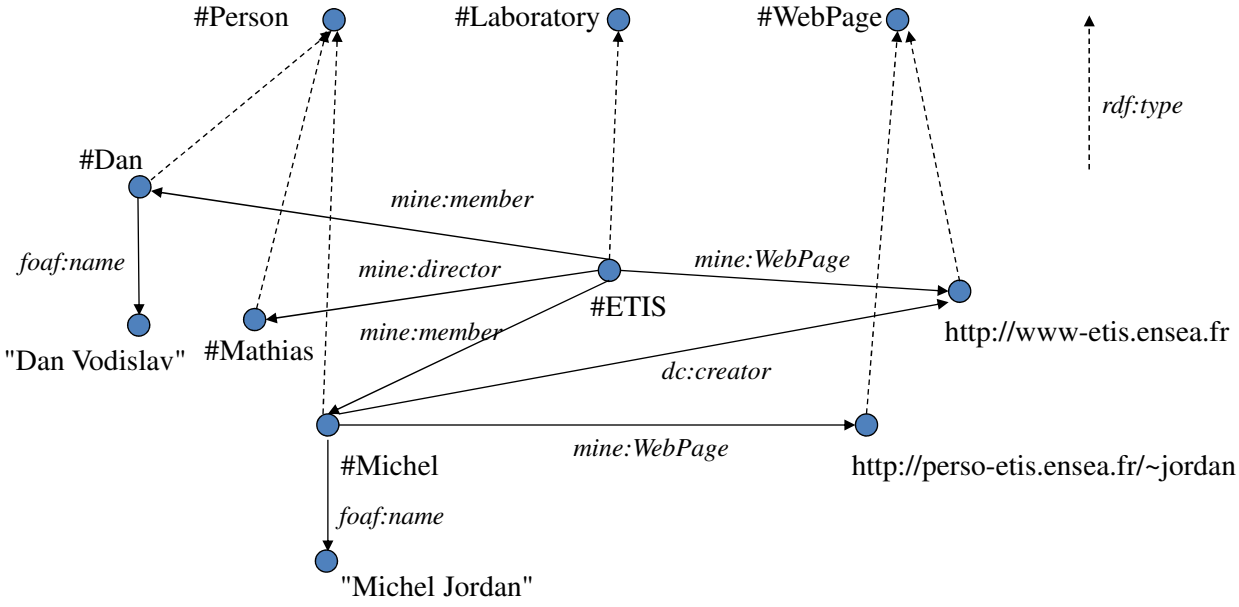
SPARQL

- Query language for RDF data
  - W3C Recommendation 2013 (SPARQL 1.1)
- The most common form of a SPARQL query:

```
SELECT [DISTINCT] ?var1 ?var2 … ?varm
WHERE { pattern1 .
        pattern2 .
        …
        patternn }
```

  - Patterns are triples in TURTLE format
  - Variables are used in the patterns
  - Conjunctive queries
  - Result: table of values (bindings) corresponding to ($?var_1$, …, $?var_m$)

Example



SELECT queries

- One or several patterns
  - Any element of a pattern (triple) may be a variable

  E.g. *The members of the ETIS laboratory*

```
SELECT ?x
WHERE {
  <http://myapp.myorg.com/ETIS> <http://myapp.myorg.com/member> ?x .
}
```

or

```
PREFIX mine: <http://myapp.myorg.com/>
PREFIX : <http://myapp.myorg.com/>
SELECT ?x
WHERE {
  :ETIS mine:member ?x .
}
```

Result

x
<http://myapp.myorg.com/Michel>
<http://myapp.myorg.com/Dan>

SELECT queries (cont'd)

E.g. *Who has a web page and what is that page*

```
PREFIX mine: <http://myapp.myorg.com/>
PREFIX : <http://myapp.myorg.com/>
SELECT ?x ?y
WHERE {
  ?x mine:WebPage ?y .
}
```

x	y
<http://myapp.myorg.com/ETIS>	<http://www-etis.ensea.fr>
<http://myapp.myorg.com/Michel>	<http://perso-etis.ensea.fr/~jordan>

E.g. *Who created the web page of the ETIS laboratory*

```
PREFIX mine: <http://myapp.myorg.com/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://myapp.myorg.com/>
SELECT ?x
WHERE {
  ?x dc:creator ?y .
  :ETIS mine:WebPage ?y .
}
```

x
<http://myapp.myorg.com/Michel>

Grouping the patterns by subject

E.g. *The name and the web page of Michel*

```
PREFIX mine: <http://myapp.myorg.com/>
PREFIX : <http://myapp.myorg.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
  :Michel foaf:name ?name ;
          mine:WebPage ?page .
}
```

name	page
"Michel Jordan"	<http://perso-etis.ensea.fr/~jordan>

Optional patterns

E.g. *The name and the web page of ETIS members*

```
PREFIX mine: <http://myapp.myorg.com/>
PREFIX : <http://myapp.myorg.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
  :ETIS mine:member ?x .
  ?x foaf:name ?name .
  OPTIONAL{
    ?x mine:WebPage ?page .
  }
}
```

name	page
"Michel Jordan"	<http://perso-etis.ensea.fr/~jordan>
"Dan Vodislav"	

Union

E.g. *The ETIS members, including its director*

```
PREFIX mine: <http://myapp.myorg.com/>
PREFIX : <http://myapp.myorg.com/>
SELECT ?x
WHERE {
  {
    :ETIS mine:member ?x .
  }
  UNION
  {
    :ETIS mine:director ?x .
  }
}
```

x
<http://myapp.myorg.com/Michel>
<http://myapp.myorg.com/Dan>
<http://myapp.myorg.com/Mathias>

Sorting, limit, offset

E.g. *The second person in alphabetical order of the name*

```
PREFIX : <http://myapp.myorg.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?x
WHERE {
  ?x a :Person ;
     foaf:name ?y .
}
ORDER BY ASC(?y)
LIMIT 1
OFFSET 1
```

- Remarks
  - (?x a *Type)* is a shortcut for (?x rdf:type *Type*)
  - ORDER BY may use ASC or DESC (ASC is the default value)
  - LIMIT *n* limits the number of returned results to *n*
  - OFFSET *m* discards the first *m* results

Filtering

E.g. *The persons whose name starts by "M"*

```
PREFIX mine: <http://myapp.myorg.com/>
PREFIX : <http://myapp.myorg.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?x
WHERE {
  ?x a :Person ;
     foaf:name ?y .
  FILTER (regex(?y, "^m", "i"))
}
```

- FILTER : Boolean condition on the value of the variables
  - Arithmetic operators for numerical values
  - Tests : *isURI*, *isBlank*, *isLitteral*
  - Comparison operators
  - Logical operators to combine conditions : &&, | |, !
  - ***regex***(*text*, *pattern* [, *option*])

Other query types

- ASK: existence test

  E.g. *Is there a web page for the ETIS laboratory?*

```
PREFIX mine: <http://myapp.myorg.com/>
PREFIX : <http://myapp.myorg.com/>
ASK {
  :ETIS mine:WebPage ?x .
}
```

- DESCRIBE: returns a description of the queried resources
  - Non standard description returned by the SPARQL service
  - In general: the values of the properties of the resource

  E.g. *Description of persons and of their web pages*

```
PREFIX mine: <http://myapp.myorg.com/>
PREFIX : <http://myapp.myorg.com/>
DESCRIBE ?x ?y
WHERE {
  ?x a :Person ;
     mine:WebPage ?y .
}
```

CONSTRUCT

- Builds a graph as a result

  E.g. *Laboratory members, with their name, their director and the laboratory web page*

```
PREFIX mine: <http://myapp.myorg.com/>
PREFIX : <http://myapp.myorg.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
CONSTRUCT {
  ?m foaf:name ?n ;
     mine:director ?d ;
     mine:LabWebPage ?p .
}
WHERE {
  ?l a :Laboratory ;
     mine:member ?m ;
     mine:director ?d ;
     mine:WebPage ?p .
  ?m foaf:name ?n .
}
```

Federated queries

- Query several SPARQL endpoints in a single query

  E.g. ETIS *laboratory members born at the same place as Claude Monet*

```
PREFIX mine: <http://myapp.myorg.com/>
PREFIX : <http://myapp.myorg.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?name
WHERE {
  :ETIS mine:member ?x .
  ?x foaf:name ?name ;
     mine:bornAt ?place .
  SERVICE <http://dbpedia.org/sparql> {
  <http://dbpedia.org/resource/Claude_Monet> dbo:birthPlace ?place.
  }
}
```

RDF on the Web

- Goal: semantic description of web pages content
  - Web of documents → web of data → semantic web
  - Support for data integration at the web scale
- How?
  - Micro-formats and micro-data
  - JSON-LD
  - RDFa
  - Linked Open Data
- Prerequisite : common standard vocabularies
  - Dublin Core, FOAF, SKOS, etc.

Micro-formats and micro-data

- Semantic information added to HTML documents
  - Used by web browsers, search engines, etc.
- Micro-formats: *class* attribute indicating predefined classes
  - Predefined micro-formats for: person, institution, event, etc.
  - See http://www.microformats.org/
  - Drawback: not any kind of object can be described

E.g. Use of the *hCard* micro-format to describe persons

```
<div class="vcard">
  <em class= "fn">Jean Dupont</em>
  <span class= "title">Ingénieur</span> chez <span
class="org">Google</span>
  <span class= "adr">
    <span class="street-address">2 rue du Moulin</span>
    <span class="locality">Village-sur-Eau</span>
    <span class="postal-code">54321</span>
  </span>
</div>
```

Micro-formats and micro-data (cont'd)

- Micro-data: extensible, more powerful than micro-formats, types and properties are distinguished
  - Predefined vocabularies (e.g. *http://data-vocabulary.org* - Google, http://ogp.me/ns# - Open Graph from Facebook)
  - Recently: *Schema.org* initiative (*http://schema.org*), to unify micro-data types between the various web browsers

E.g. Use of *Person* and *PostalAddress* types

```
<div itemscope itemtype="http://schema.org/Person">
  <span itemprop="name">Jean Dupont</span>
  <span itemprop= "jobTitle">Ingénieur</span> chez
<span itemprop="affiliation">Google</span>
  <span itemprop="address" itemscope
itemtype="http://schema.org/PostalAddress">
    <span itemprop="streetAddress">2 rue du Moulin</span>
    <span itemprop="addressLocality">Village-sur-Eau</span>
    <span itemprop="postalCode">54321</span>
  </span>
</div>
```

RDFa

- RDFa = “RDF in attributes”
  - RDF descriptions in (X)HTML pages through HTML attributs
  - Can express all the RDF constructs: URI, namespaces, types, …
  - Used by specialized web browsers, applications, …
    - Richer results in search engines

E.g. (subject, property, value) triple

```
<p about="http://myapp.myorg.com/Michel"
   property="http://xmlns.com/foaf/0.1/name">
  Michel Jordan
</p>
```

  - *about* (subject), *property* (property) attributes
  - Value in the text within the HTML tag

RDFa (cont’d)

E.g. Relation between resources (subject, predicate, object)

```
<a about="http://myapp.myorg.com/Michel"
   rel="http://purl.org/dc/elements/1.1/creator"
   href="http://www-etis.ensea.fr">
  Page created by Michel
</a>
```

  - *about* (subject), *rel* (predicate), *href* (object) attributes

E.g. Object resources other than HTML links

```
<span about="http://myapp.myorg.com/ETIS"
   rel="http://myapp.myorg.com/director"
   resource="http://myapp.myorg.com/Mathias">
  director: Mathias Quoy
</span>
```

  - *resource* attribute instead of *href*

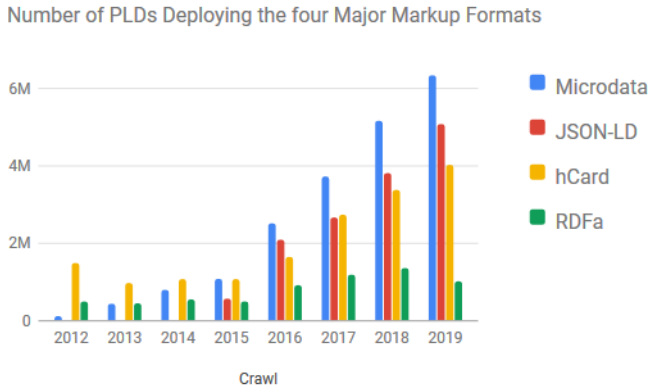
JSON-LD

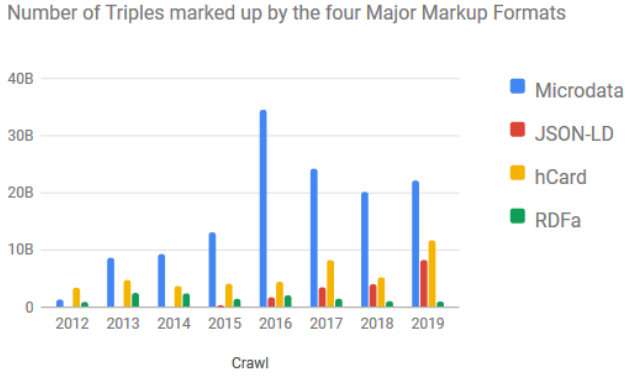
- JSON for Linking Data
  - Became popular with the expansion of the JSON format
  - Data in JSON format as scripts within the HTML page
- Simple example

```
{
  "@context": "http://myapp.myorg.com",
  "@id": "http://myapp.myorg.com/ETIS",
  "@type": "Laboratory",
  "name": "ETIS",
  "WebPage": "http://www-etis.ensea.fr",
  "member": ["http://myapp.myorg.com/Michel", "http://myapp.myorg.com/Dan"]
}
```

RDFa, JSON-LD, micro-data, micro-formats

- Analysis in time on a representative sample of web sites
  - 2012: micro-formats, RDFa and much less micro-data
  - Growing of micro-data (schema.org effect)
  - After 2015: JSON-LD, with a strong growing
  - RDFa: good start, but slow increase and now decreasing
- Linked Open Data: publish data, not annotated documents





Challenges for exploiting web data

- Variety
  - Automatic mapping between instances and vocabularies
  - Hidden web → semantic web
- Volume
  - Distributed management and search on the web
  - Cloud computing RDF storage and querying

References

- T. Heath, C. Bizer, "*Linked Data: Evolving the Web into a Global Data Space*", http://linkeddatabook.com/editions/1.0/
- *linkeddata.org*