4.2 KNOWLEDGE REPRESENTATION

4.2.1 Representation of Semantics

Information retrieval and data mining aim at extracting and using semantics that is implicitly represented in documents and data

Alternatively, semantics can, and is, also explicitly represented and manipulated ☞ knowledge representation

Historically speaking, the first information systems were based on explicit representation of the semantic models used. In business information systems, the data relevant to businesses was and is till today explicitely modelled in a structured data model, e.g. the relational data modelled. The data of the model is manipulated either manually, e.g. through manual entry, or processed by algorithms, e.g. for accounting. Only at a later stage with the advent of more computing power the use of implicitly represented models, as found in text documents, became feasible. The methods we have covered so far in this lecture fall in this later class. Interestingly, as algorithms become increasingly powerful in extracting information from unstructured content, the explicit representation of the extracted models and their use for further inference is now gaining in interest and importance.

Database Schemas

Schemas define data structures for databases

- Relational database schemas, XML Schemas

Agreement on data structures

- Well understood meaning of data values
- Data consistency, e.g., integrity constraints

Optimizing query evaluation and data storage

- e.g., indexing, query optimization

STUDENTID	NAME	COURSE
1234	John	DIS
3456	Bob	physics
2345	Ann	DIS

Schema

The probably most widely used approach for explicitly representing semantic models is through database schemas in database management systems. Schemas in database management systems play two important roles. First, they define data structures that capture that application semantics. They label data values with names (e.g., attribute names) that have well-understood semantics, or of which the semantics is provided by additional documentation. Second, by providing well-defined data structures and additionally imposing integrity constraints, database schemas additionally assure that data stored in the database remains structurally and semantically coherent. Exploiting knowledge on data structures also enables query and storage optimizations.

However, in many contexts of information management, e.g., on the Web, schemas are not always readily available, and imposing the use of schemas might be too rigid.

Data on the Web

Example: Searching biological data

- Based on textual content of Web pages (e.g. Google)

Searching for data on "anglerfish"

- Results will be precise

This seems easy, but the same for "leech"

- Organism leech
- Software: LeechFTP
- Rock band: Leech
- Authors: Leech
- Protein sequences: …MNTS**LEECH**MPKGD…

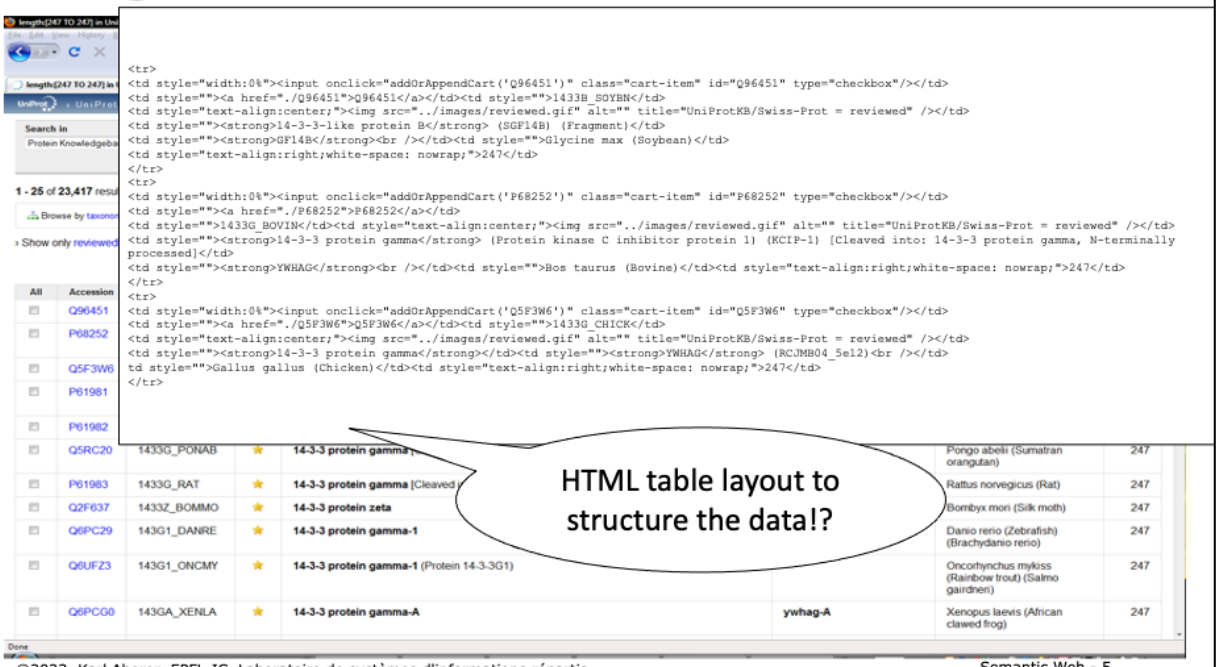
Search for "257" …

When publishing information on the web, no schemas are used, even if the published data could follow a regular data structure. Consequently, when searching the Web, search is content-based, e.g., using keyword queries.

This may work well for very characteristic search terms, such as illustrated for the example “anglerfish”, but might become problematic, when the search term has an ambiguous meaning, as illustrated for the example “leech”. Since a text search the type of the information that is searched cannot be specified – in our example we would need to specify that we meant the “leech” is an organism - the search algorithm cannot disambiguate the different potential meanings of the search terms. In the extreme case, if we would try to search for a numerical value as a search term, the results become essentially useless.

This suggest that the use of schema information for Web search could have some merit.

Using HTML to Structure Data



A typical approach to handle data in the Web is to publish documents with a regular structured layout, e.g., using HTML tables. In this way a context is provided that allows to correctly interpret data values. When inspecting the source code of a typical HTML page publishing structured data, we recognize that it might be difficult to automatically process such a page and to correctly interpret the data contained in it. Such processing might be very involved and error-prone, even more so as in the context of editing Web page layout no mechanism assures that the intended page structure is correctly maintained. The problem is that a document layout language, such as HTML, never has been conceived to specify semantics of data.

Web Scraping

Encoding data semantics through layout is a bad idea

- Generations of “Web scrapers” experience this

FOR E-COMMERCE DATA SCIENTISTS: LESSONS LEARNED SCRAPING 100 BILLION PRODUCTS PAGES

July 02, 2018 Ian Kerins 11 Comments

Challenge #1 - Sloppy and Always Changing Website Formats

It might be obvious and it might not be the sexiest challenge, but sloppy and always changing website formats is by far the biggest challenge you will face when extracting data at scale. Not necessarily because of the complexity of the task, but the time and resources you will spend dealing with it.

https://www.zyte.com/blog/price-intelligence-web-scraping-at-scale-100-billion-products/

The widely adopted approach of publishing data through HTML documents has given rise to a whole industry of web scraping. It is of interest to gather data from the Web, for example, for comparison of prices from ecommerce sites. However, experience shows that this is a complex and time-consuming task. It is also a never-ending task as formats of Web sites can change at any time and there exist no standards on how the data is represented in HTML.

Application-specific Markup

Limitations of HTML

- Structure of data expressed as layout:
  `<tr><td> leech </td><tr>`
- Semantics of data hard to analyse and difficult to share
- No schemas, no constraints

Application-specific markup (tags)

Making the meaning of data explicit

- SwissProt: `<Species> leech </Species>`
- EMBLChange: `<Organism> leech </Organism>`

Embedding of schema information into the data

Therefore, an alternative approach for specifying the meaning of data contained in Web documents, has been conceived. As for HTML, the approach is based on the idea of using tags, i.e., markup text that associates parts of a document with a name. However, instead of using tags to provide layout instructions (e.g., a table format), tags are used to provide domain-specific meaning to data contained in a document. For example, markup can be used to specify that a specific name (such as leech) denotes an organism. This approach of using markup is widely used in different contexts. For example, email formats contain different standard markups to indicate the meaning of different parts of a mail. In the Web, the use of markup has found wide-spread adoption through the XML format. XML generalizes the markup approach from HTML to allow applications to specify their application-specific tags and to embed them into documents. If you are not familiar with XML, it would be helpful to refresh your knowledge on XML and schemas in XML.

4.2.2 Semi-structured Data

Data that embeds schema information (through tags, markup etc.) to explain the meaning of data values and to relate different data values (e.g. hierarchically) = semi-structured data

No predefined data structure, no schema required!

Examples of semi-structured data

- Email
- Microformats (e.g. hCard)
- JSON
- XML (Extensible Markup Language)

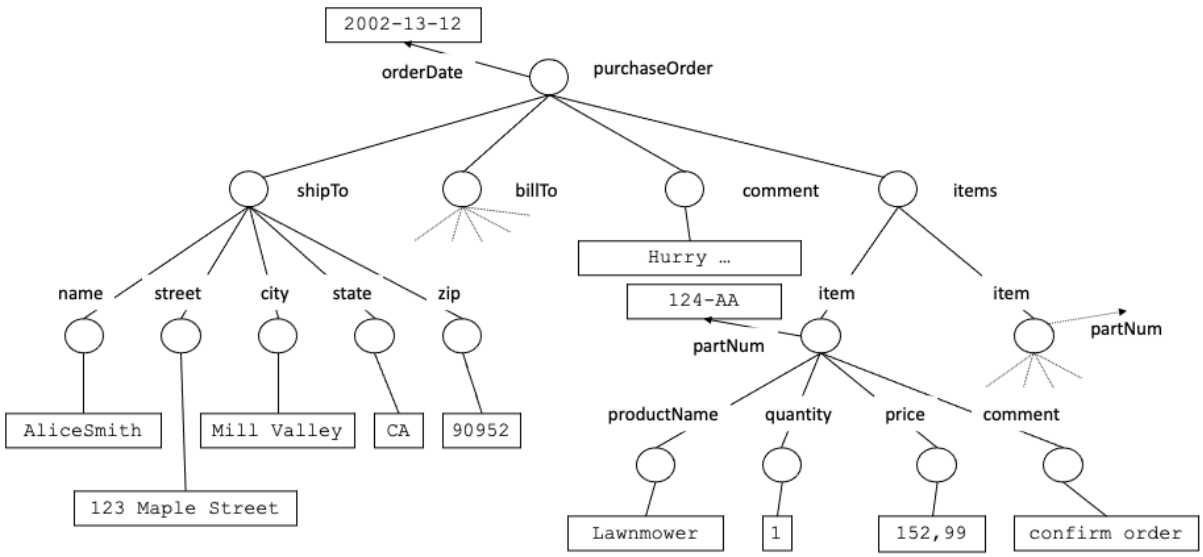
Data that contains embedded schema information, such as tags or markup, is generally called semi-structured data. Semi-structured data allows to indicate the meaning of data values, as in structured data with schemas, without necessarily requiring predefined schemas. This combines flexibility of modelling with the ability to specify meaning. The fact that semi-structured data does not require schemas, does not imply that we cannot define schemas for semi-structured data. XML is an example of this. We can have XML structured documents without schema, so-called well-formed XML documents, but define in addition schemas for XML (document type definitions, so-called XML DTDs). Documents that follow such a schema are then called valid XML documents.

XML – a Document Markup Language

```
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
  <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <city>Old Town</city>
    <state>PA</state>
    <zip>95819</zip>
  </billTo>
  <comment>Hurry, my lawn is going wild!
  </comment>
  <items>
    <item partNum="872-AA">
      <productName>Lawnmower
      </productName>
      <quantity>1</quantity>
      <price>148.95</price>
      <comment>Confirm this is electric
      </comment>
    </item>
    <item partNum="926-AA">
      <productName>BabyMonitor
      </productName>
      <quantity>1</quantity>
      <price>39.98</price>
      <shipDate>1999-05-21</shipDate>
    </item>
  </items>
</purchaseOrder>
```


Originally XML has been conceived as a document model. In this example of an XML document, we can clearly see the document nature of XML. An XML document consists of hierarchically nested tags which enclose textual content.

XML- a Semi-structured Data Model



However, the same XML document can also be viewed as structured data. An XML parser would typically generate a hierarchical data structure as illustrated. This explains why we can understand XML also as a data model, more precisely a semi-structured data model, as it embeds schema information directly into the data.

Schema-less data

Benefits of schema-less data

Increased flexibility

- e.g., dynamically adding or dropping structural elements such as attributes

Self-contained data

- e.g., context (schema information) directly encoded into data (markup)

Drawbacks

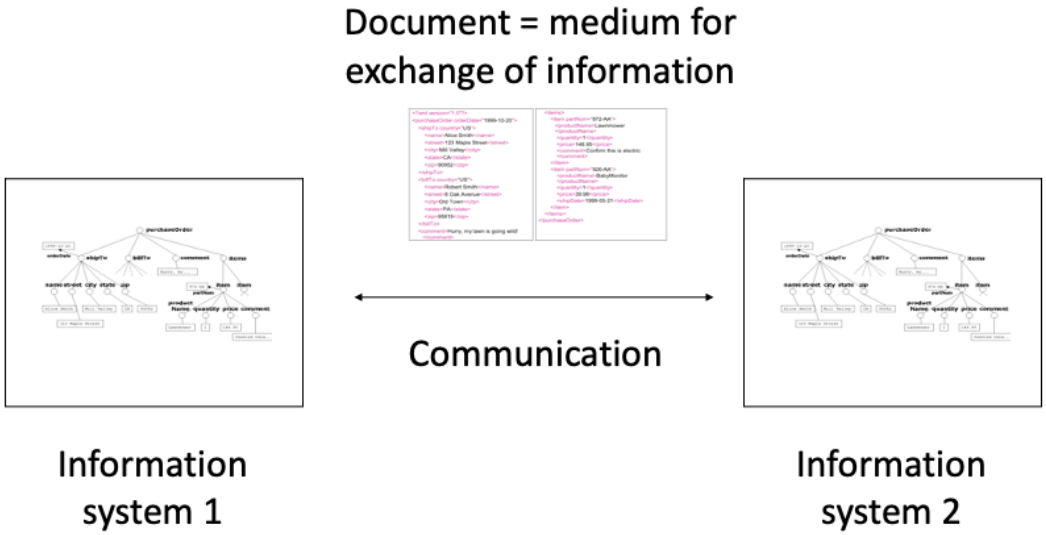
Loss of consistency

Certain optimizations not feasible

Semi-structured data can be schema-less. This has significant advantages for applications that do not know the structural elements in advance. For example, often it is helpful if new types of attributes can be added in a database as application requirements evolve. Another advantage is that by embedding schema information into the data, data can be interpreted without requiring additional context information in the form of database schemas. This can be useful in applications where data is exchanged.

Having no schema has however also its drawbacks as the additional flexibility also can be at the detriment of data consistency. Applications might, for example, not be able to properly process attributes that have been arbitrarily added. Also many optimizations that rely on the availability of a schema, and on the assumption that the schema is stable, cannot be applied.

Serialization: Data and Documents



The duality of XML, being a document model and a data model at the same time makes it particularly suitable for applications that exchange data over communication networks. For every data collection or database that is represented in XML there exists a canonical encoding into a text (document). The encoding is called serialization. Since documents are sequences of symbols, they can be naturally exchanged over communication networks. There exist many other similar models for serializing data into documents. Recently JSON has become widely used for exchanging data among applications and has to a certain extent substituted XML in this role in many contexts. However, JSON is less suited for applications that enrich natural language text with structured data.

Semi-structured data …

A. is always schema-less
B. always embeds schema information into the data
C. must always be hierarchically structured
D. can never be indexed

Why is XML a document model and not just a general data model?

A. It supports application-specific markup
B. It supports domain-specific schemas
C. It has a serialized representation
D. It uses HTML tags

Serialised vs. Semi-structured Data

Many of the semi-structured data models have been conceived to structure text documents (XML) or serialise data (JSON)

However, there exist semi-structured data models that are not based on a serialized representation

→ knowledge graphs

Serialisibility and semi-structured data are concept that are strongly correlated but not the same. Many of the semi-structured data models, like XML and JSON, have been introduced to either enrich documents with data semantics, or to encode data in a serialized format. However, it is perfectly possible to define a semi data model, that does not necessarily aim at a serialized representation. Knowledge graphs are an example for this - though they of course also can be and are serialized. Knowledge graphs have been conceived to provide a flexible way to model data and capture semantics. We will next review the development of this concept.

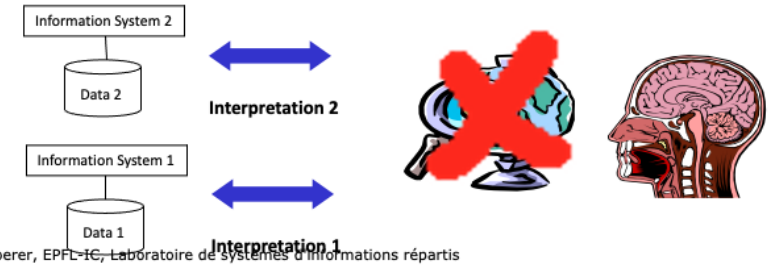
4.2.3 The Semantic Web

User-defined markup (schemas): provides possibility to share interpretation of data across various applications

Different databases – Different schemas

- SwissProt: Find `<Species> leech </Species>`
- EMBLChange: Find `<Organism> leech </Organism>`

Problem: Semantic Heterogeneity → Semantic Web



XML provides a framework to encode data on the Web in a standardized fashion, to deal with irregular Web data structures and to exchange Web data among information systems. Thus, it is an approach to address syntactic heterogeneity on the Web.

The markup used in XML can be interpreted by applications. This enables automated processing of Web data. However, for properly interpreting the markup, semantic heterogeneity needs to be addressed as well. The same concepts can ^represented in different application contexts differently, e.g., by using different terms to denote the same meaning. This is a major obstacle to enable interoperability on the Web and indeed in industry this is considered as one of the major problems in the use of information technology. Already in 2010, Mike Brodie at the time at GTE, estimated that the total cost of semantic integration world-wide amounts to 1 trillion$ / year. Very likely, this figure has in the meantime increased.

The Vision of W3C: Semantic Web

The *Semantic Web* is an extension of the current Web in which *information is given well-defined meaning*, better enabling computers and people to work in cooperation. The mix of content on the Web has been shifting from exclusively human-oriented content to more and more data content. The Semantic Web brings to the Web the idea of having data defined and linked in a way that it can be used for more effective *discovery, automation, integration, and reuse across various applications*. For the Web to reach its full potential, it must evolve into a Semantic Web, providing a universally accessible platform that allows data to be shared and processed by automated tools as well as by people. The Semantic Web is an initiative of the World Wide Web Consortium (W3C), with the goal of extending the current Web to facilitate Web automation, universally accessible content, and the 'Web of Trust'.
(http://www.w3.org/2001/sw/Activity)

In order to address the problem of semantic interoperability and thus enable automated processing of Web data, the W3C initiated the “Semantic Web” initiative. The Semantic Web is a framework that builds on Web technology, including the XML framework and extends it with technologies that facilitate semantic interoperability.

Three Ways to Overcome Semantic Heterogeneity

1. Standardization: agree on common user-defined markup (schemas)
   - great if no pre-existing applications
   - great if power player enforces it
2. Translation: create mappings among different schemas and databases
   - requires human interpretation and reasoning
   - mappings can be difficult, expensive to establish
3. Annotation: create relationships to agreed upon conceptualizations
   - requires human interpretation and reasoning
   - annotation can be difficult, expensive to establish
   - reasoning over the conceptualization can provide added value

There exist three basic approaches to tackle the problem of automating semantic processing of the data on the Web:

1. Standardization avoids the problem of semantic heterogeneity a priori. This approach is used when there exists already (historically) a wide agreement on the structure of relevant data and their interpretation. For example, terminology in the financial industry is largely standardized, and therefore it is not a major problem to produce agreed upon formal specifications of such terminology and corresponding schemas to represent the data. This is even more the case as there exist in this type of business environment typically strong players that can enforce the standards.
2. Translation or mapping between different schemas and databases is the second possibility to establish semantic interoperability. This is the approach that has been extensively adopted for data integration problems in relatively small and controlled domains, such as inside businesses and organizations. The requirements in these domains are typically changing not too quickly, thus much effort can be invested into developing the necessary mappings in order to properly map data from one representation into another, or to map data from multiple representations into one common global representation.
3. The third possibility is slightly different from the second: instead of engineering mappings between heterogeneous schemas for each integration problem, one first agrees on a common conceptualization of the domain, covering relevant aspects for a large class of applications. This conceptualization is normally called an ontology and is assumed to be application independent. Since ontologies have a formal representation, they are machine-processable. Once such an ontology is in place, existing information system can relate their structural elements for expressing certain concepts (e.g., element names) to concepts from the ontology. This then ideally enables other applications to properly interpret the contents of the information system. In addition, ontologies in the general case should include reasoning capabilities, which would permit to use not only hard-coded, or pre-canned knowledge, e.g., in form of explicitly relating concepts from an information system to the ontology, but also to derive new knowledge by combining existing knowledge.

This differentiation among different approaches to semantic interoperability has been standardized: according to ISO 14258 (Concepts and rules for enterprise models), there are three basic ways to relate entities together:

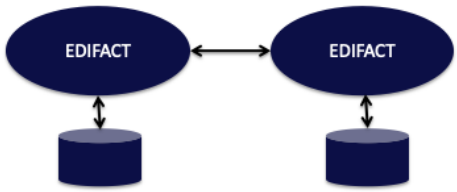
Integrated approach. There exists a common format for all models. This format must be as detailed as the models themselves. The common format is not necessarily a standard but must be agreed upon by all parties in order to elaborate models and build systems. (-> Standardization)

Federated approach. There is no common format. In order to establish interoperability parties must accommodate on the fly. Federated approach implies that no partner imposes their models, languages and methods of work. This means they must share a common ontology. (-> Translation)

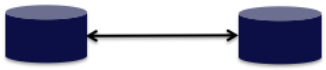
Unified approach. There exists a common format but only at a meta-level. This meta-model is not an executable entity as in the case of the integrated approach but provides a way for semantic equivalence to allow mapping between models. (-> Annotation)

Mapping: Three Approaches

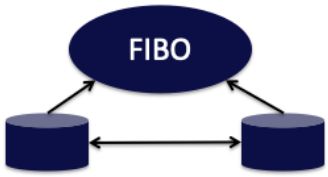
1. Standardization
   - Mapping through standards



2. Mapping
   - Direct mapping



3. Ontologies
   - Mediated mapping



Conceptually there exist three principal approaches of how to address semantic heterogeneity. A first approach is to map all the models to one common global model. This approach is taken with standardization. For example, EDIFACT (https://www.unece.org/cefact/edifact/) is an international standard that models all concepts that are commonly used in business and trade. For exchanging information, the information systems map their data to EDIFACT and can thus share their data with other information systems.

A second approach consists of constructing directly a mapping among two information systems, without using any additional, shared knowledge in form of standards or ontologies

A third approach is to relate the model of an information system to a common model, frequently called ontology, and use this mapping to construct a direct mapping among the different models used in the information systems. . For example, in the financial domain there exists a standard reference ontology called FIBO (https://fib-dm.com/finance-ontology-transform-data-model/)

Direct Schema Mapping

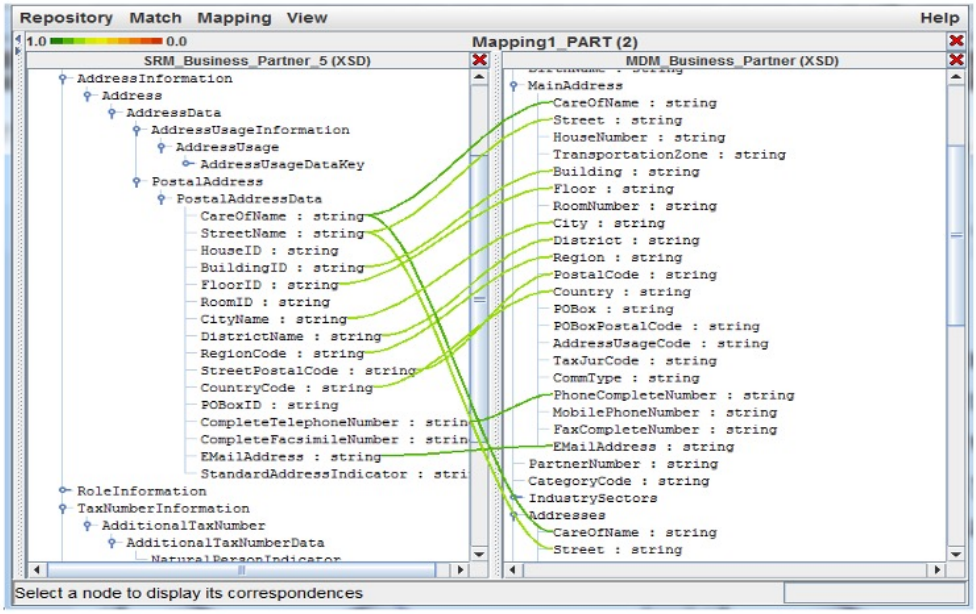
Assume all data represented in canonical data model (e.g. relational)

- detect correspondences (schema matching)
- resolve conflicts
- integrate schemas (schema mapping)

Mappings are frequently expressed as queries (e.g. SQL Query)

Creating direct mappings among information systems has been widely applied for mapping data that is stored in relational, object-oriented and XML databases, where the model is represented as a database schema. The problem is known as the **schema mapping** problem. Given two schemas of databases, first schema elements are identified that are likely to correspond to each other, i.e., that are likely representing the same real-world aspect. This can be done by using many types of analysis using structural and content features of the database schema and the database. Tools for supporting these steps are called schema mapping tools. Once this step is performed some conflicting correspondences may occur, e.g., mapping some concept in one schema to two different ones in the other. These need to be resolved typically through decisions taken by humans. Once the correspondences are consistent, mappings can be automatically derived. The mappings are typically expressed as queries of the data manipulation language.

Schema Matching Tools



This screenshot shows a schema matching tool that maps between different XML schemas. The lines indicate which attributes might be related to each other. One can easily see a lot of conflicts where the same attribute corresponds to multiple attributes in the other schema.

Matching with LLMs

It is possible to generate schema transformation code with GPT

	name	description	incident_type	receiver_name	receiver_country	receiver_region	receiver_category	receiver_category_subcode	initiator_nam
0	Cyberwar Azerbaijan-Armenia 2000/Azerbaijani A...	Azerbaijani Hackers took down and defaced webp...	Disruption	[['Not available']]	['Armenia']	[['ASIA', 'CENTAS', 'CSTO']]	[['State institutions / political system', 'Me...	NaN	['N availabl
1	Cyberwar Azerbaijan-Armenia 2000/Armenian Coun...	In response to the previous attack Armenian ha...	Disruption	NaN	['Azerbaijan']	[['ASIA', 'CENTAS']]	[['Social groups',		

Code properly working on first run!

	victim	damage	attack	attacker	affected	time	fi
0	Cyberwar Azerbaijan-Armenia 2000/Azerbaijani A...	operational disruption	Disruption	Not available	None	Unknown	
1	Cyberwar Azerbaijan-Armenia 2000/Armenian Coun...	operational disruption	Disruption	Lazor	None	Unknown	
2	Taiwan Election Hack 2000	operational disruption	Disruption	Unknown	None	Unknown	None

	State institutions / political system	Taiwan	Chinese hackers succeeded in attacking several...

I have two datasets describing cyber-attacks.

The first is structured as follows (attribute names and 3 samples)

... give here the head of the table ...

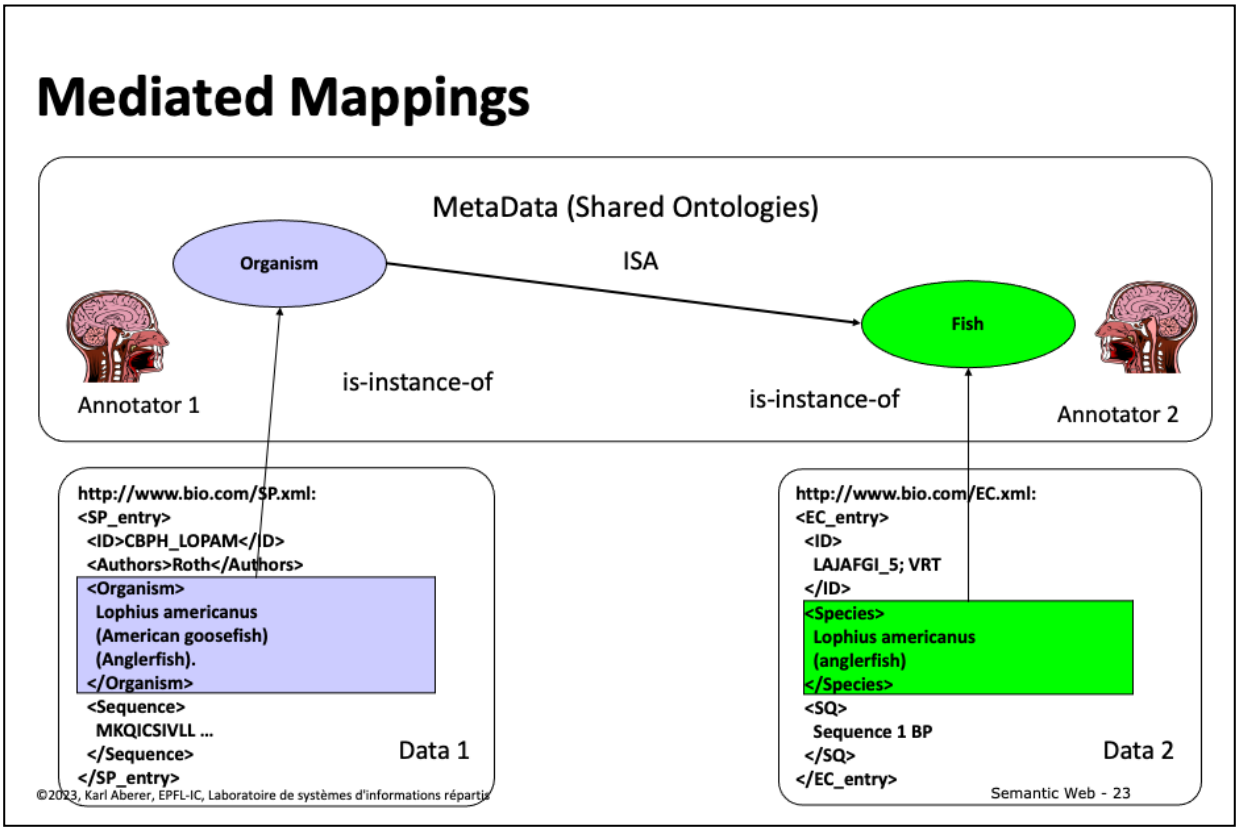
The second has the following format

... give here the head of the table ...

Provide a python code that transforms the first format into the second format.

This was the code generated for this case

```
def transform_to_json_format(record):
return { "victim": record.get("name", "Unknown"),
"damage": "operational disruption", # Assuming
operational disruption for all "attack":
record.get("incident_type", "Unknown"), "attacker":
record["initiator_name"].strip("[]'") if
isinstance(record.get("initiator_name"), str) else
"Unknown", "affected": "None", # Default value
"time": record.get("start_date", "Unknown")[:7],
"financial loss": "None", # Default value "sector":
record.get("receiver_category",
"Unknown").split(',')[0].strip("[]'"), "country":
record["receiver_country"].strip("[]'") if
isinstance(record.get("receiver_country"), str)
else "Unknown", "summary":
record.get("description", "No description
available"), "doc_id": "None" # Default value }
```



This simple example illustrates of how ontologies can help to increase semantic interoperability and how new knowledge can be generated by inference. Take the earlier example of biological databases. These can use different schemas to represent the same or related facts. For example, System 1 uses the term Organism to denote an organism, and System 2 uses the term Species to do the same. Two annotators, who share the same ontology, now inspect the document and each of them associates the elements with terms taken from the ontology. So, Annotator 1 decides that the element Organism corresponds to information related to an organism, whereas annotator 2 concludes by analyzing the content that the element is related to information about a fish and annotates correspondingly. The annotation can be perfoermed by adding an is-instance-of relationship. At this point, the facts that both annotators used the same ontology, and that reasoning is possible in this ontology come into play. Since in the ontology a Fish is a sub-concept of Organism (a fact represented formally by an ISA relationship in the ontology) an automated processing tool (e.g., for searching for information) can exploit this relationship and correctly identify for a request for information on Organisms in both databases the related elements.

Ontologies

Ontologies are an explicit specification of a conceptualization of the real world (Gruber, 1993)

Ideally

- different information systems agree on the same ontology
- relate their model/schema/data elements to the ontology
- mapping can be constructed via the ontology

Requires agreement on the conceptualization of the real world!

Thus, ontologies provide a “proxy representation” for the real world, to which the interpretation of data in information systems refers to and thus provide an interpretation of the data in an information system that can be automatically processed. In order to realize this approach, different technical challenges need to be addressed.

Creating Ontologies

Different approaches to create ontologies

1. Ontology engineering
   - Manual effort
   - Tools for editing and checking consistency
2. Automatic generation of ontologies
   - From large document collections or existing structured data sources

In order to make ontologies available, they need to be constructure. This is the practice of ontology engineering. Ontology engineering is largely a manual process, that can be supported by tools for checking consistency and facilitating editing. Ontology are often not only constructed once but need to be maintained to stay up to date with developments in a given domain. For example, important entities, like organizations, can change, technology evolves etc.

More recently, also automated methods based on information extraction from large document collections are becoming available to that end. We will see later examples of such methods using statistical and machine-learning approaches.

Modeling and Encoding of Ontologies

Issues

- Modeling primitives and their semantics
  - what does an arrow mean?
  - what does "instance-of" mean?
  - what does ISA mean?
- Standardized encodings of the model
  - Into document language (XML, HTML)
  - Enriching document content with semantic markup

For representing ontologies, a mathematical model for the constructs used in modelling the application domain is required, as well as a data model to represent such a model in a data management system. The choice of the formal mathematical is important, as the model should at the same time be expressive to capture a wide range of real-world aspects, but at the same time understandable by the user, have a well-defined semantics and relevant reasoning capabilities.

Finally, the models should also provide a serialized representations, e.g., by encoding them in standard document languages such as XML or HTML. Such encodings can at the same time be used to enrich text documents with semantic annotations.

Model Requirements for Ontologies

	HTML	XML	RDF	OWL
Simplicity	+	+	+	+
Exchangeability	+	+	+	+
Non-intrusive annotation	+	-	+	+
Domain-specific vocabularies	-	+	+	+
Modeling primitives	-	-	+	+
Reasoning capabilities	-	-	-	+

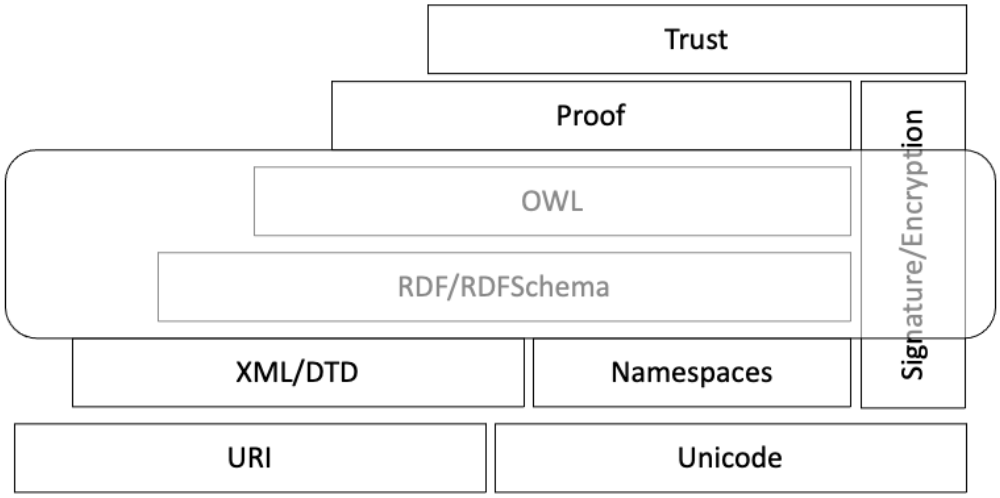
"separate structure from presentation"

"separate interpretation from structure"

We have identified so far different models for representing information in the Web, HTML, XML, and RDF. They all serve specific purposes and exhibit characteristic properties. This table summarizes those properties and highlights interesting “separation of concerns”. RDF and OWL, an extension of RDF to provide more reasoning capabilities separate the concern of reasoning at a semantic level from structuring of data. XML has been serving to separate the concern of structure of data, from providing layout information to documents. The table highlights related design objectives.

1. Simplicity: the success of the Web was always founded on the principle of simplicity of concepts to encourage wide-spread use. Early version of ontology models, such as CyC, where partly also not widely adopted due to their complexity. Similarly, XML was developed as a simplification of a predecessor standard, called SGML, which was more complex.
2. Exchangeability: Since the Web is a communication environment, any kind of data that is processed must be easily exchangeable. This is what motivated the use of XML as a data representation format in the first place and should apply for ontology models as well.
3. Non-intrusive annotation: machine-processable knowledge required for the interpretation of data will be associated with the data typically a-posteriori. Also there does not always exist a unique interpretation for the same data. Therefore, any attempt to encode the knowledge required for interpretation directly into the data is not practical. This excludes, for example, the approach to use XML elements for annotation.
4. Domain-specific vocabularies: an ontology model must provide a mechanism that permits to introduce vocabulary or terminology that is specific to a domain, in other words the possibility to specify schemas for different domains.
5. Modeling primitives: since an ontology model will be used in many different contexts, they have to offer a sufficiently rich set of possibilities to model complex structures and relationships. There exists a rich experience in modeling (e.g., from data modeling in databases with the entity-relationship models) that can be applied to ontology models.
6. Reasoning Capabilities: even simple forms of reasoning within the ontology layer can make the interpretation of the data much more powerful and thus automated processing of data in the Semantic Web.

W3C Web architecture



This figure depicts the architecture adopted by the W3C. It standardizes different layers of information processing. The lowest layer is concerned with identification and encoding as basic building blocks for building data models. XML together with the namespace mechanism is the data modeling framework. RDF and its extension OWL enable semantic interpretation and exchange. The levels of proof and trust are still rather hypothetical.

An ontology …

A. helps to separate layout issues from the structural representation of data
B. defines a common syntactic framework to represent standardized domain models
C. can be used as a mediation framework for integrating semantically heterogeneous databases

4.2.4 RDF - Resource Description Framework

How to produce a serialized representation of a graph?

How to describe complex entities?

How to incorporate schema information into a knowledge graph?

How to reason about the knowledge graph?

RDF has been designed by the W3C with a number of different questions in mind. We will now provide a step-wise introduction into the modeling framework.

RDF Model

RDF (Instances)

- *Statements* about *Resources* (addressable by an URI) and literals (XML data)
- Statements are of the form: subject property object
- Like simple natural language sentences
- RDF statements are themselves resources (reasoning about RDF)
- *Properties* define relationships to other resources or atomic values

RDF-Schema

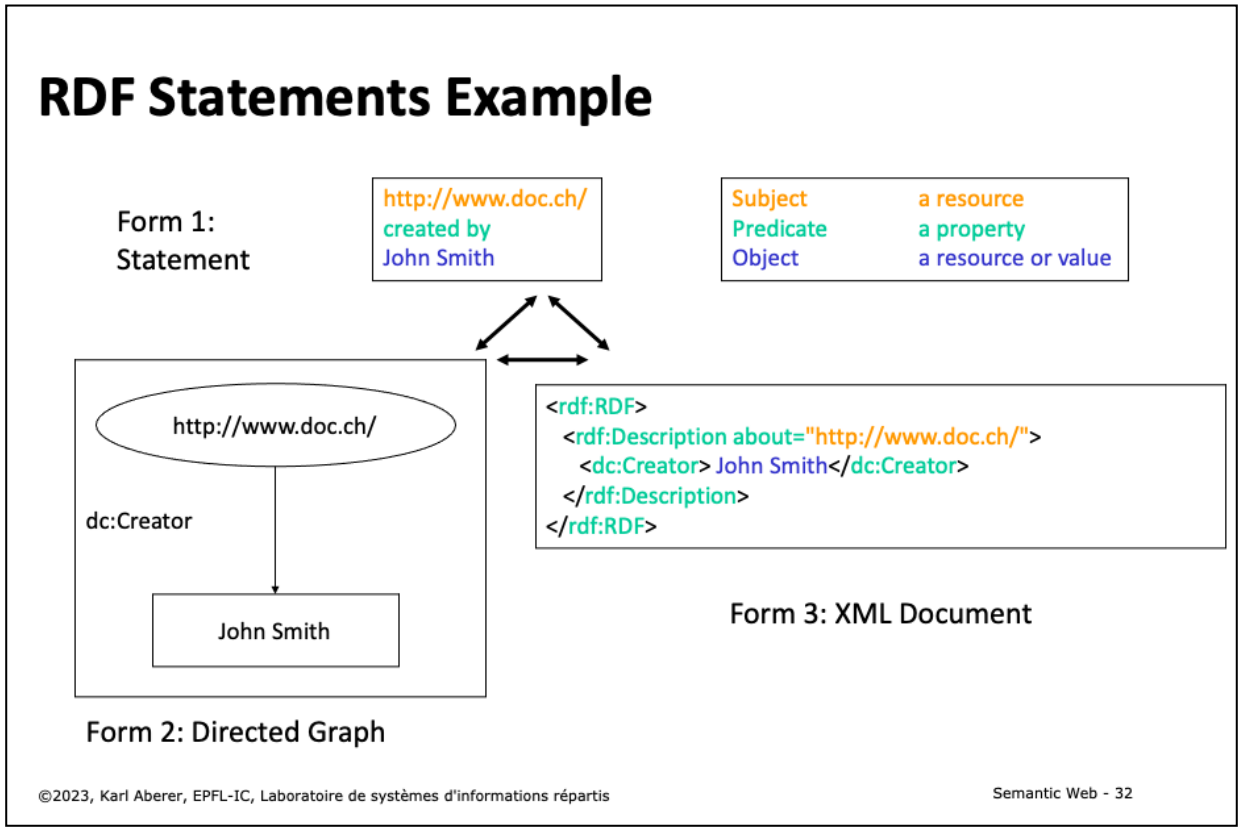
- Data model to specify schemas for RDF instances
- Which properties are applicable for which objects with which subjects
- Defines "grammar" and "vocabulary" for semantic domains (ontology language)

Relation RDF vs. RDFS comparable to well-formed vs. valid XML

RDF is a standard supported by the W3C (http://www.w3.org/RDF/) to represent metadata. It is a graph-oriented data model used to annotate XML documents. RDF consists of two parts: a language for representing metadata instances (RDF), which allows to annotate Web resources with statements. The Web resources are addressed by Universal Resource Identifiers (URI), of which URL's are the most important example. Thus, any Web document or part of it can be annotated. The second part of the RDF standard is a language for specifying schemas for RDF Instances. This language enables specification of the vocabulary and grammar that is used for forming statements for annotation. Since RDF statements can be created also without using RDF schemas, RDF is a semi-structured data model, similar to XML.

The RDF model is like the entity-relationship (ER) model. Entities correspond to resources and relationships correspond to properties. The main difference is that RDF requires that relationships are directed : the resource from which the (directed) relationship emerges is assigned a property with a value to which the relationship points. This reflects the intention to use RDF to associated metadata (the value) with data (the source of the relationship). Sample RDF applications include PICS (annotating documents with information on the suitability of the content for certain groups, e.g., like the movie rating system) and Dublin Core (annotating documents with basic bibliographic information.

It is important to understand that the syntax of RDF and its encoding into XML is well-specified, but that the semantics RDF is only specified in a « semi-formal » manner. This introduces certain ambiguities for applications interpreting RDF statements.

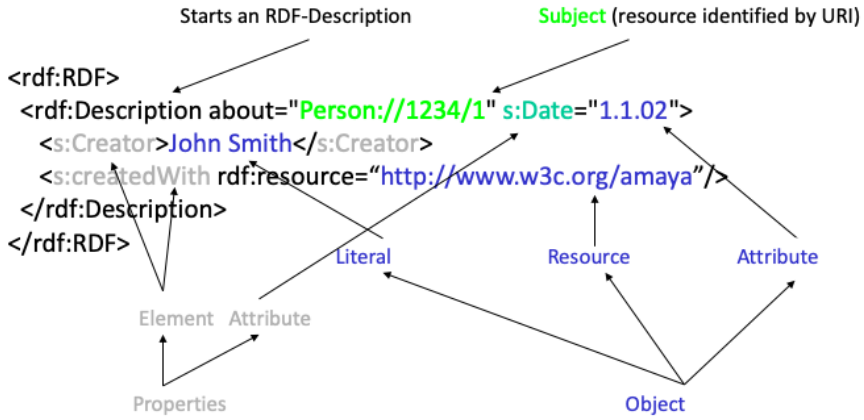


The basic constituent of RDF is an RDF statement. We can view RDF statements in three different ways: (1) like natural language sentences, where the subject is a URI (uniform resource identifier) and the object can be either a URI or a String; (2) as directed graphs, where the subject and object are represented as nodes (an ellipsis is used to represent resources identified by a URI and rectangles to represent literals as atomic XML values) and the predicate is represented as directed link , or (3) as XML documents where the RDF statement is encoded into XML format. The graph representation is suitable for visual presentation and reasoning, whereas the XML representation is used for exchange and storage.

RDF Syntax

Many syntactic varieties possible

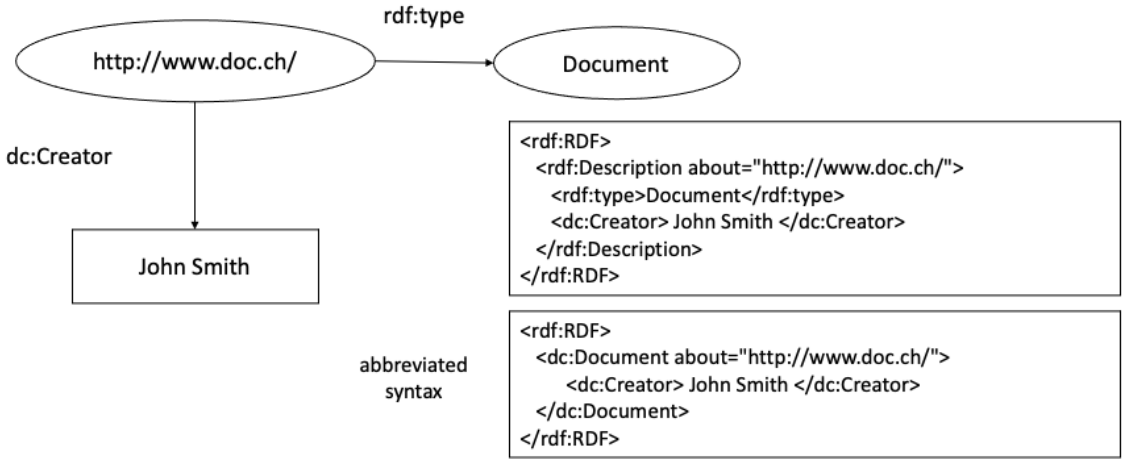
Basic form



For encoding RDF statements into XML there exist several syntactic variations, which can make the reading of RDF documents sometimes rather difficult. Here we summarize the different variants. The basic pattern of encoding is as follows: the subject is represented by an XML element called rdf:Description. This element is the root of the document fragment representing the RDF statement. In the content of the element, one finds one or more predicates, represented by XML elements, e.g., s:Creator. The content of this XML element in turn represents the object of the RDF statement. If the object is not a literal, one can alternatively represent the object as an attribute of the predicate element. Also, both the predicate and the object can be encoded into the element representing the statement, as it is shown for s:Date predicate.

Typing Resources

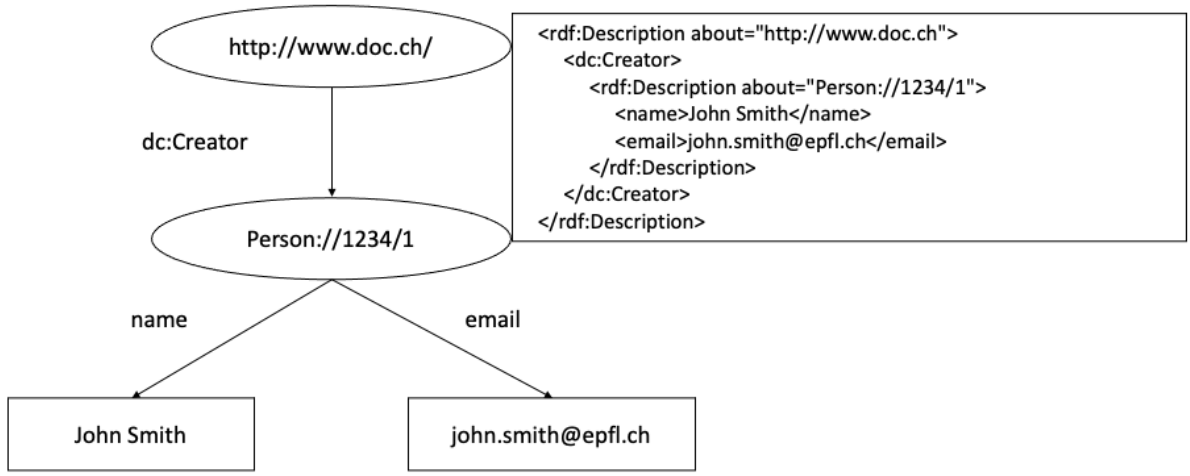
Resources can be associated with a type by using **rdf:type** (a special property)



RDF allows us to categorize resources into different classes (typing). For that purpose, one associates with the resource that should be categorized another resource, that represents the category, using a special RDF property rdf:type. We will see later, when introducing RDF schema, what are possible uses of the type statement. With respect to encoding, the type property can either be represented explicitly like any other property, or one can use a special abbreviated syntax, where the name of the type becomes the element name of the element representing the statement.

RDF Complex Values

Use an intermediate resource



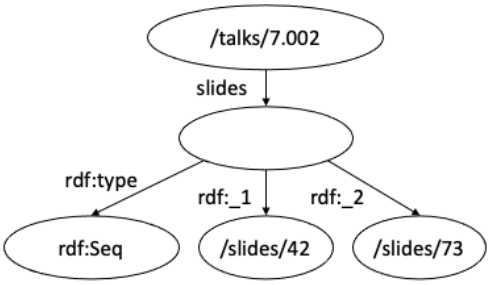
Associating a subject with a property whose value is a simple object is the simplest form of a statement. In general, the value of a property (the object of the statement) may be a complex statement itself, representing a complex data structure. In order to represent such complex object values a new intermediate resource is created - in the example Person://1234/1 - to which different properties are associated. Note that this is different of directly associating these properties with the subject of the overall statement, http://www.doc.ch/ in the example (why?).

In the XML encoding such a complex object value can be represented by directly inlining the complex object into the content of the statement.

RDF Containers

Containers
- Bag (unordered)
- Seq (ordered)
- Alt (alternatives)

Quantifiers
- about: John is author of the talk (consisting of many slides)
- aboutEach: John is author of each slide of the talk



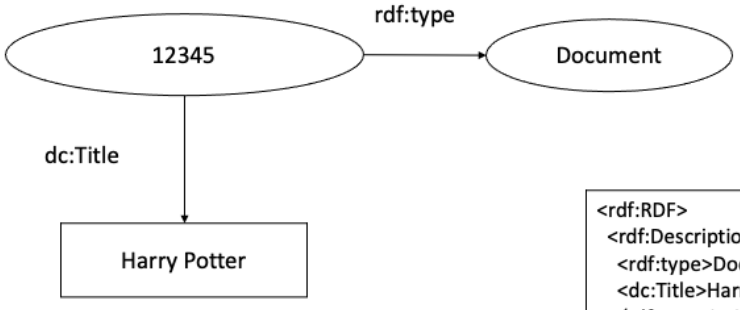

```
<rdf:RDF>
 <rdf:Description about="/talks/7.002">
  <s:slides>
    <rdf:Seq>
      <rdf:_1 resource="/slides/42"/>
      <rdf:_2 resource="/slides/73"/>
    </rdf:Seq>
  </s:slides>
 </rdf:Description>
</rdf:RDF>
```


Statements can be made not only using single resources, but as well using collections of resources. For that purpose, RDF provides the container concept. Containers are special resources of one of three container types that are specified in the RDF standard. A container resource is then associated with a set of other resources. By creating a statement using the container object as "object", one can express statements made about the set of objects. More precisely, one can specify whether the statement is a statement of the set of objects "as a whole" or a statement that applies to each element of the set individually. What the implications of this distinction are is not further specified in the RDF standard, which is an example where the semantics of RDF is not clearly specified.

There exist three different types of containers: bags which are unordered multi-sets (= sets with multiple occurrences of the same resources), sequences which are ordered sets (i.e., lists) of resources and alternatives which is a single resource that is to be chosen out of a given set. The property labels can be used to impose an order on the elements of the set, by using labels _1, _2 etc. If the order is irrelevant one can use rdf:li as element name, instead of rdf:_1, rdf:_2 etc.

Creating New Resources

New RDF resources are created by using **rdf:ID** (a special property)




```
<rdf:RDF>
 <rdf:Description rdf:ID="12345">
  <rdf:type>Document</rdf:type>
  <dc:Title>Harry Potter</dc:Title>
 </rdf:Description>
</rdf:RDF>
```

```
<rdf:RDF>
 <rdf:Description rdf:about="#12345">
  <rdf:type>Document</rdf:type>
  <dc:Title>Harry Potter</dc:Title>
 </rdf:Description>
</rdf:RDF>
```

Alternative syntax

```
<rdf:RDF>
 <dc:Document rdf:ID="12345">
    <dc:Title>Harry Potter</dc:Title>
 </dc:Document>
</rdf:RDF>
```

Abbreviated syntax

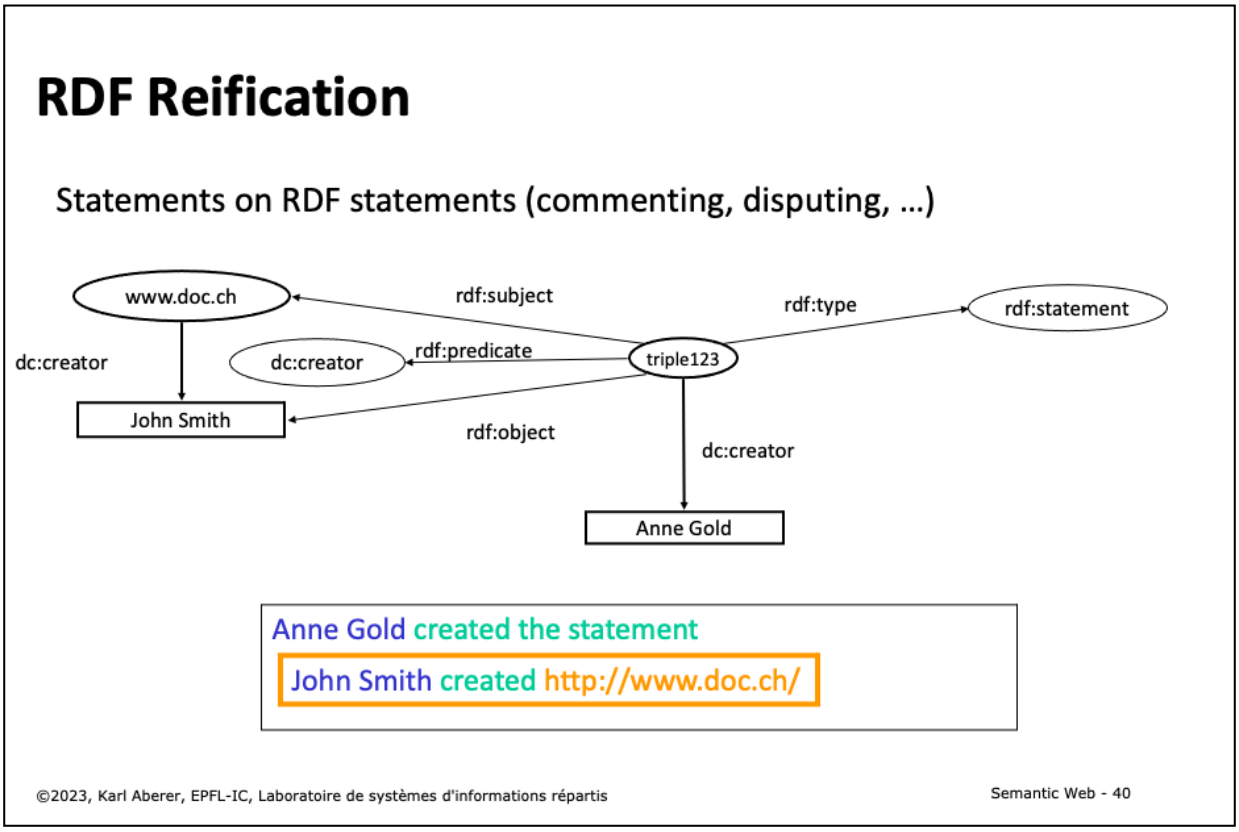
RDF resources are not necessarily Web resources identified by URIs but can also be instantiated within RDF statements. In other words, *new resources* can be defined as part of RDF statements. A resource is in general anything that can be *identified* on the Web. As a consequence, also a newly created resource requires an identifier. For this purpose, RDF provides the rdf:ID attribute to introduce the identifier of the resource. This attribute replaces rdf:about attribute when the statement is about a new resource instead about an existing resource. Using the identifier, this new resource can then be referred to in other RDF statements.

A basic statement in RDF would be expressed in the relational data model by a table …

A. with one attribute
B. with two attributes
C. with three attributes
D. cannot be expressed in the relational data model

The type statement in RDF would be expressed in the relational data model by a table …

A. with one attribute
B. with two attributes
C. with three attributes
D. cannot be expressed in the relational data model



In RDF everything is a resource. In particular, RDF statements are considered as resources and therefore it should be possible to make statements about them. From the viewpoint of the Semantic Web this is in fact extremely important. Since statements do not express some necessarily agreed upon facts, but often different interpretations of data, it is important to provide the possibility of annotating annotations (such as illustrated in the simple example). This allows to comment on annotations, to agree or dispute them etc..

When considering the graph representation of RDF, it is not immediately clear of how to treat a statement as a resource, since a statement consists of three structural elements, the subject, the object, and the predicate. But we can apply the same "trick" as we did already for complex objects and collections. We introduce a new resource which serves as representation for the statement. This resource has as properties all the constituents that make up the statement it represents. This process is called *reification*. It is straightforward to connect the resource representing the statement it to it's subject and object, as they are both resources themselves. Also, the type of the object can be determined through a property rdf:type pointing to the special resource rdf:statement representing the category of statements. For the predicate, a specific new resource representing the predicate is required. We will see later, when introducing RDF schema, that this new resource is indeed part of a schema for RDF statements, expressing that statements using these properties are valid in the given context. The reified RDF statement is connected to its constituents through the special RDF properties rdf:object, rdf:subject and rdf:property. By reifying a statement, one creates a new resource, which can be anonymous, if no identifier is associated with it using rdf:ID, or can be referenced if it has such an identifier.

RDF Reification - Syntax

The statement has an anonymous resource as subject, namely the reified statement which is fully characterized by its properties!

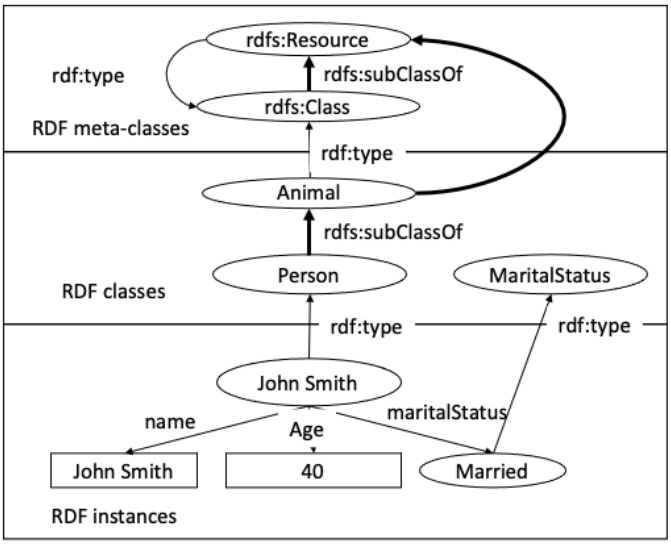
```
<rdf:RDF>
 <rdf:Description about="#triple123">
  <rdf:subject resource="http://www.doc.ch"/>
  <rdf:predicate resource="http://description.org/schema#Creator"/>
  <rdf:object>John Smith</rdf:object>
  <rdf:type resource="http://www.w3.org/TR/WD-rdf-syntax#Statement"/>
  <dc:Creator>Anne Gold</dc:Creator>
 </rdf:Description>
</rdf:RDF>
```


The XML encoding of reified statements follows the principles that have been introduced before. The reified statement is represented as a complex, anonymous object.

RDF Schema - Classification

RDF resources
- anything that can be described

RDF classes
- Categories for subjects and objects
- Classes allow to associate a type with an RDF instance
- Different classes can be in a subClass relationship (be included in each other)
- Classes are themselves RDF instances



Finally, we introduce RDF Schema. RDF Schema provides two basic mechanisms.

1. Categorization RDF resources, into *classes*.
2. Constraints on the possible use of properties, in the form of constraints expressing which classes can participate as subjects and objects in statements using a specific property.

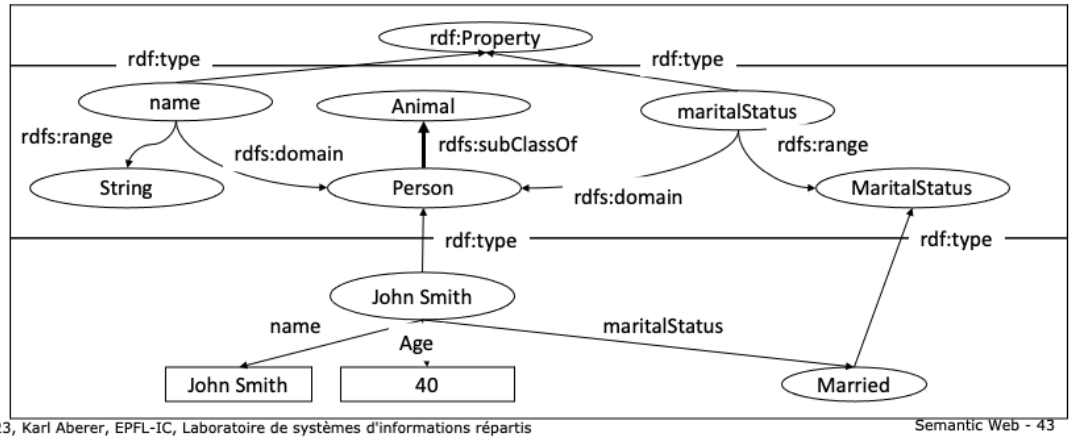
First, we describe the classification mechanism. Classes are represented themselves as resources, which are of type rdfs:Class, a special class in the RDFS specification. The type property rdf:type is used, as in RDF, to indicate the type of a *class resource*. If an application resource is of a specific type, then the resource is connected to the corresponding class resource via the rdf:type property. In the illustration two examples of such a case are included: the resource with ID „John Smith" belongs to the class (or is of type) Person and the resource "Married" is of type MaritalStatus, which is a resource representing a specific predicate type. Between different classes a subclass relationship can be specified, by using the attribute rdfs:subClassOf. The intended semantics is that any resource belonging to the subclass also belongs to the superclass (containment relationship).

An interesting aspect of RDFS is the modeling of the RDF and RDFS concepts within RDFS itself. This is done by introducing a meta-class level that models the modeling constructs and reflects the paradigm that in RDF everything is a resource, including the concepts introduced by RDF and RDFS. In particular, classes are sets of resources, and thus the rdfs:Class resource is a subClass of rdfs:Resource. Application classes, such as "Animal", are sets of resources, and thus also subclass of rdfs:Resource. On the other hand, the type of a resource is indicated by the rdf:type property. Since rdfs:Resource is a class, it is connected via the rdf:Type property to the rdfs:Class resource. This produces the cyclic structure that we can observe within the RDF meta-class schema. In this figure only a small fragment of the RDFS meta-class schema is shown.

RDF Schema - Properties

RDF properties: connect resources

- The RDF instance must have the properties that are declared for the class
- rdfs:domain: classes of which the instances may have a property
- rdfs:range: classes of which the instances may be the value of a property



The second important concept of RDFS is the possibility to constrain the usage of RDF properties that connect resources. As everything in RDF, RDF properties are resources themselves. Thus, they are represented in an RDF schema as resources. For example, maritalStatus is a resource representing a property. In RDF schema it is now possible to constrain the usage of properties as follows: by connecting the property resource through the property rdfs:domain to a class resource, one specifies that the subject when using this property must originate from that class, i.e., be of the type of this class. Similarly for the object, the range can be constrained using rdfs:range. Ranges can also be of atomic type, in that case one connects the property resource to (predefined) resources representing the data type of the atomic type. In the example above this is the atomic type STRING.

The RDFS model bears a lot of similarities with object-oriented model. However, a fundamental difference is that properties (attributes in OO terminology) are defined independently of classes.

RDF Schema - Syntax

```
<rdf:RDF xml:lang="en"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
<rdfs:Class rdf:ID="Person">
        <rdfs:subClassOf rdf:resource="http://www.w3.org/classes#Animal"/>
</rdfs:Class>
<rdf:Property ID="maritalStatus">
         <rdfs:range rdf:resource="#MaritalStatus"/>
         <rdfs:domain rdf:resource="#Person"/>
</rdf:Property>
<rdf:Property ID="name">
         <rdfs:range rdf:resource="http://www.w3.org/classes#String"/>
         <rdfs:domain rdf:resource="#Person"/>
</rdf:Property>
<rdf:Property ID="age">
         <rdfs:range rdf:resource="http://www.w3.org/classes#Integer"/>
         <rdfs:domain rdf:resource="#Person"/>
</rdf:Property>
<rdfs:Class rdf:ID="MaritalStatus"/>
<MaritalStatus rdf:ID="Married"/>
<MaritalStatus rdf:ID="Divorced"/>
<MaritalStatus rdf:ID="Single"/>
<MaritalStatus rdf:ID="Widowed"/>
</rdf:RDF>
```


Since RDF schemas are expressed as RDF statements they can be encoded into XML along the same principles that we have introduced for RDF statements earlier. This example shows the complete encoding of the RDF schema we have used in our example before. Throughout the schema the abbreviated syntax for statements is used, replacing the element name "Description" by the corresponding class name of the subject of the description. Note of how using the ID attribute, in the RDF schema new resources are introduced. They can be referred to from other statements using the newly introduced identifier prefixed with #. The specification of the class with ID "MaritalStatus" includes the specification of the complete extension of the class, enumerating all possible values the members of this class can take, i.e., all possible predicate names that predicates of type MaritalStatus can take. Strictly speaking, the statements creating the instances of class are not part of the schema level but part of the instance level of RDF.

Which is true?

A. Reification is used to produce a more compact representation of complex RDF statements
B. Reification requires to make a statement the subject of another statement
C. Reified statements always make a statement about another statement

Which of the following are part of the RDF schema language?

A. The « type » statement for RDF resources?
B. The « domain » statement for RDF properties?
C. The « subject » statement for RDF statements?

4.2.5 Semantic Web Resources

Examples of Popular Ontologies and Knowledge Bases

- WordNet
- WikiData
- Google Knowledge Graph
- Schema.org
- Linked Open Data

In the following we provide an overview of the common resources and standards used today in the Semantic Web.

WordNet

English dictionary with semantic relationships

Synonymy. Words that have similar meanings, e.g., happy and glad.

Antonymy. The opposite of synonymy, e.g., happy and sad.

Nouns only

Hypernymy. Hierarchical relationship between words, e.g., furniture is a hypernym of chair since every chair is a piece of furniture.

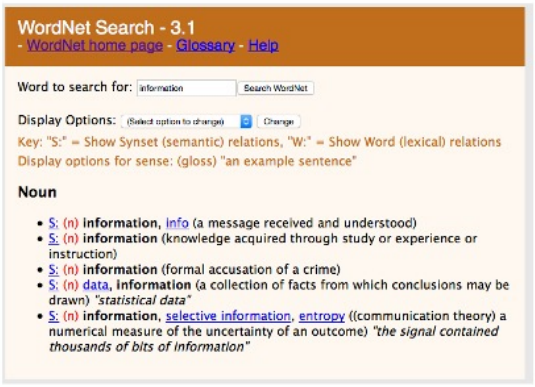
Hyponymy. Opposite of hypernymy. Dog is a hyponym of canine since every dog is a canine.

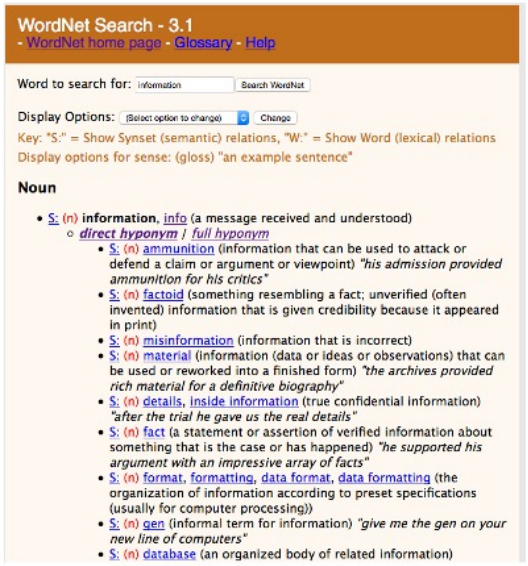
Meronymy. Part-whole relationship. For example, paper is a meronym of book, since paper is a part of a book.

WordNet is a quite old project conducted at Princeton University. The objective of the project is to provide a complete dictionary of the English language, including all semantic relationships among words. In the meantime, similar projects have been started also for other languages.

WordNet Example

http://wordnet.princeton.edu/





The data of WordNet is publicly available and can be accessed online. It has been used in a variety of projects related to problems sich as query expansion in retrieval, word sense disambiguation or or text classification.

WikiData

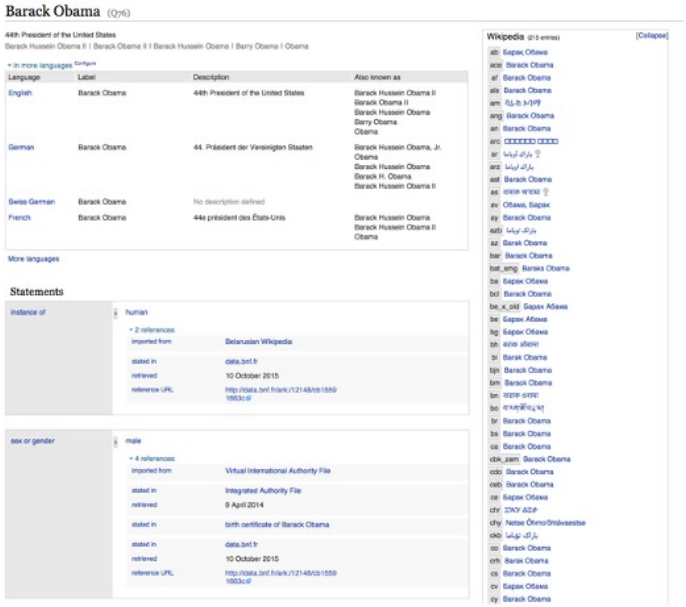
Community project to create an open database of structured data

- Data curation model like Wikipedia
- Intended to support Wikipedia (InfoBoxes)
- 84 million entities (2020)
- Multi-lingual
- Both API access and full databases dumps (JSON, RDF)

WikiData is a compagnion project of WikiPedia. It aims at creating a universal knowledge graph and is based on the same community approach as WikiPedia. It is used by WikiPedia to provide the InfoBoxes.

Example WikiData

www.wikidata.org



The WikiData project uses (a part of) the RDF model to represent the facts. One interesting aspects is that it correlates the representation of the terms describing an entity or concept across languages.

Google Knowledge Graph

Google's internal knowledge base to support the search engine

- Populated from FreeBase and internal data
- Enriched with the support of **schema.org**
- Accessible through API

https://www.google.com/intl/es419/insidesearch/features/search/knowledge.html

Google has popularized knowledge graphs by the development of its knowledge graph. The project actually started from an earlier community effort called FreeBase. It was then enriched with internal data from Google. It uses schema.org as the underlying data model.

Schema.org

Collaborative, community activity to create, maintain, and promote schemas for structured data on the Internet

- Sponsoring companies: Google, Microsoft, Yahoo and Yandex
- Two type hierarchies: textual property values, things that they describe
- Core vocabulary currently consists of 642 Types, 992 Properties, and 219 Enumeration values
- Used by other knowledge bases, e.g., Google Knowledge Graph API, Dbpedia, etc.

Schema.org is a generic data model, or knowledge base schema, that has been adopted and promoted by the large Internet platforms. It consists of two hierarchies, one for descibing types of attributes and one for describing types of entites.

Example

Core plus extension vocabularies

- Thing
  - Action
    - AchieveAction
      - LoseAction
      - TieAction
      - WinAction
    - AssessAction
      - ChooseAction
        - VoteAction
      - IgnoreAction
      - ReactAction
        - AgreeAction
        - DisagreeAction
        - DislikeAction
        - EndorseAction
        - LikeAction
        - WantAction
      - ReviewAction

Action

Thing > Action

An action performed by a direct agent and indirect participants upon a direct object. Optionally happens at a location with the help of an inanimate instrument. The execution of the action may produce a result. Specific action sub-type documentation specifies the exact expectation of each argument/role.

See also blog post and Actions overview document.

Usage: Between 100 and 1000 domains

[more...]

Property	Expected Type	Description
Properties from Action		
actionStatus	ActionStatusType	Indicates the current disposition of the Action.
agent	Organization or Person	The direct performer or driver of the action (animate or inanimate). e.g. "John" wrote a book.
endTime	DateTime	The endTime of something. For a reserved event or service (e.g. FoodEstablishmentReservation), the time that it is expected to end. For actions that span a period of time, when the action was performed. e.g. John wrote a book from January to "December". Note that Event uses startDate/endDate instead of startTime/endTime, even when describing dates with times. This situation may be clarified in future revisions.
error	Thing	For failed actions, more information on the cause of the failure.
instrument	Thing	The object that helped the agent perform the action. e.g. John wrote a book with "a pen".
location	Place or Text or PostalAddress	The location of for example where the event is happening, an organization is located, or where an action takes place.
object	Thing	The object upon the action is carried out, whose state is kept intact or changed. Also known as the semantic roles patient, affected or undergoer (which change their state) or theme (which doesn't). e.g. John read "a book".
participant	Organization or Person	Other co-agents that participated in the action indirectly. e.g. John wrote a book with "Steve".
result	Thing	The result produced in the action. e.g. John wrote "a book".
startTime	DateTime	The startTime of something. For a reserved event or service (e.g. FoodEstablishmentReservation), the time that it is expected to start. For actions that span a period of time, when the action was performed. e.g. John wrote a book from "January" to December. Note that Event uses startDate/endDate instead of startTime/endTime, even when describing dates with times. This situation may be clarified in future revisions.
target	EntryPoint	Indicates a target EntryPoint for an Action.

This is a sample of schema.org. It is easy to see that the model covers very generic concepts that can be applied to a wide range of applications.

Encoding

Different encodings can be used, JSON, RDFa, Microdata

RDFa: microformat for embedding RDF into HTML

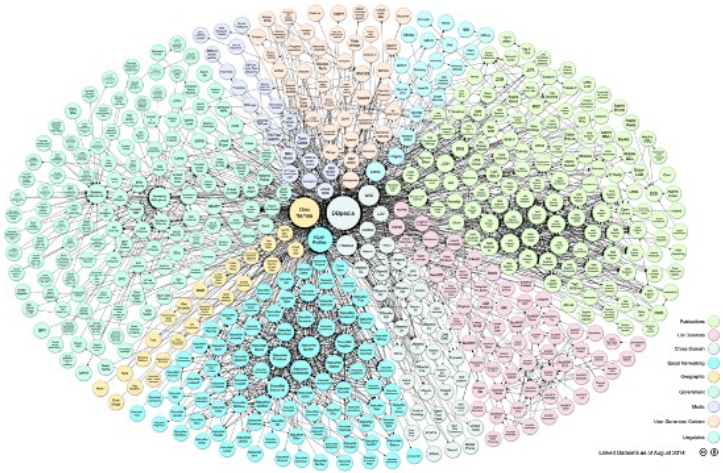
```
<div vocab="http://schema.org/" typeof="Person">
  <span property="name">Jane Doe</span>
  <img src="janedoe.jpg" property="image" alt="Photo of Jane Joe"/>
  <span property="jobTitle">Professor</span>
  <div property="address" typeof="PostalAddress">
    <span property="streetAddress">
      20341 Whitworth Institute
      405 N. Whitworth
    </span>
    <span property="addressLocality">Seattle</span>,
    <span property="addressRegion">WA</span>
    <span property="postalCode">98052</span>
  </div>
  <span property="telephone">(425) 123-4567</span>
  <a href="mailto:jane-doe@xyz.edu" property="email">
    jane-doe@xyz.edu</a>
  Jane's home page:
  <a href="http://www.janedoe.com" property="url">janedoe.com</a>
  Graduate students:
  <a href="http://www.xyz.edu/students/alicejones.html" property="colleague">
    Alice Jones</a>
  <a href="http://www.xyz.edu/students/bobsmith.html" property="colleague">
    Bob Smith</a>
</div>
```


Schema.org can be encoded in any of the existing data exchange standards. One specific type of encoding is RDFa. It is used to embed RDF statements into HTML documents in a standardized approach.

Linked Open Data

Repository of open data and knowledge bases and tools

https://www.w3.org/wiki/DataSetRDFDumps



Given the rapid adoption of knowledge graphs and the growing number of knowledge bases published, Linked Open Data is an effort to provide an common entry point to a large number of knowledge graph. An interesting aspect in this effort is the fact that knowledge bases often have references among each other. Instead of having isolated knowledge bases, this results in a network of knowledge graphs that is captured and display on the Linked Open Data Website.

Linked Open Data Example

DataSetRDFDumps

Linked Data Sets (i.e., with Dereferenceable URIs) available as RDF Dumps

- Please provide the URL for the directory containing the RDF dump files.
- Please try to have one directory or tarball per dump set -- such that we can retrieve and load the entire URL contents, to have a restored snapshot.
- Please include a Publisher/Maintainer URI, for use in constructing attribution triples.

Project	Data Exposed	Size of Dump and Data Set	Archive URL	Publisher / Maintainer URI
Addgene	Addgene catalog (tab delimited file)	1.1 MB	tab-delimited file	
Allen Brain Atlas	Science Commons extract from ABA Web site, on or shortly before 26 Feb 2007	51 MB	dump file	
Airport Data	SPARQL	754,585		Rob Styles
BAMS	BAMS	5.6 MB	bams-from-swanson-98-4-23-07.owl	
BBC John Peel sessions from DBtune.org	holding data released during Hackday, 2007	277,000 triples	peel.tar.gz	[URI?]
BBOP	All OBO ontologies	36 MB	obo-all.tgz	[URI]
BBOP	selected OBO ontologies, downloaded ~21 April 2007, augmented with inferred relations	2.6 MB	obo-in-owl.tgz	[URI]
Billion Triples Challenge Dataset 2008	various dumps	1 billion triples	download page	
Billion Triples Challenge Dataset 2009	crawled Web data	1.14 billion triples	download page	
Billion Triples Challenge Dataset 2010	crawled Web data	3.2 billion triples	download page	
Bio2RDF	various bio- and gene- related datasets	10+ billion triples	download page	
Bizkit	collaborative file describing service	330,026 discrete files, 270MB uncompressed	dump directory	
British Geological Survey (BGS) OpenGeoscience	1:625 000 Geology of the UK (DigMapGB), BGS geochronology and chronostratigraphy, BGS Lexicon of Named Rock Units	Approx 940,000 (19 MB compressed) N-Triples	data_bgs_ac_uk_ALL.zip	British Geological Survey (BGS)
	Chef Moz	290344 restaurants - 104858 reviews - 59243 links to reviews - 2402 editors	size?	URL?
Data-gov Wiki	Datasets containing RDF data converted from datasets published at http://data.gov (and other sources). The datasets are clustered by dc:subject, e.g. government budget, environmental statistics, housing and population statistics, medical cost, energy consumption, public library statistics, and labor statistics.	5+ billion triples	dump directory	Tetherless World Constellation
DBpedia	Data set containing extracted data from Wikipedia. About 2.6 million concepts described by 247 million triples, including abstracts in 14 different languages	247 million triples	dump directory	

LoD provides a common point of access to obtain a large number of public knowledge bases.

Which of the following is NOT an (instance-level) ontology?

A. Wordnet
B. WikiData
C. Schema.org
D. Google Knowledge Graph

References

Relevant articles

- Grigoris Antoniou, Frank van Harmelen, Semantic Web Primer, MIT Press, 2nd edition, 2004.
- Jeen Broekstra, Michel C. A. Klein, Stefan Decker, Dieter Fensel, Frank van Harmelen, Ian Horrocks: Enabling knowledge representation on the Web by extending RDF schema. WWW 2001: 467-478
- Stefan Decker, Sergey Melnik, Frank van Harmelen, Dieter Fensel, Michel C. A. Klein, Jeen Broekstra, Michael Erdmann, Ian Horrocks: The Semantic Web: The Roles of XML and RDF. IEEE Internet Computing 4(5): 63-74 (2000)

WebSites

- XML, XPath, Xquery, RDF: http://www.w3.org/
- OWL: http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.1