Semantic Web Architecture

Andreas Harth, harth@kit.edu, Karlsruhe Institute of Technology
Maciej Janik, janik@uni-koblenz.de, University of Koblenz-Landau
Steffen Staab, staab@uni-koblenz.de, University of Koblenz-Landau

August 31, 2010

Abstract

The Semantic Web extends the existing Web, adding a multitude of language standards and software components to give humans and machines direct access to data. The chapter starts with deriving the architecture of the Semantic Web as a whole from first principles, followed by a presentation of Web standards underpinning the Semantic Web that are used for data publishing, querying and reasoning. Further, the chapter identifies functional software components required to implement capabilities and behaviour in applications that publish and consume Semantic Web content.

Contents

Chapter 1

Semantic Web Architecture

One of the key goals of Semantic Web technologies is to provide machines with a more sapient understanding of data. To this end, an increasing number of Web sites publish data in standards defined by the World Wide Web Consortium (W3C). Given a wider availability of quality data online, applications can leverage a common data access and integration layer for providing elaborate services to users. The chapter derives the architecture of the Semantic Web from first principles, gives an overview of the architecture of Semantic Web applications, and covers building blocks of the Semantic Web in more detail.

The chapter is structured as follows: Section 1.1 introduces a scenario describing an information need which is difficult to satsify using traditional Web technologies based on hypertext, but will be easy to answer using Semantic Web technologies. Section 1.2 presents detailed requirements for an architecture. Section 1.3 derives an architecture from first principles, Section 1.4 covers the individual components deployed on the Semantic Web. Section 1.5 concludes.

1.1 A Semantic Web Scenario from Today

"Which type of music is played in UK radio stations?" and "Which radio station is playing titles by Swedish composers?" are the type of questions that are very hard to answer using existing Web search engines; the upcoming Semantic Web provides a better framework to facilitate answering such queries. The information required to answer these questions is available on the Web. In fact, a large amount of such information already exists in formats amenable to machine processing on the Semantic Web. The reason that Web search engines fail at answering such questions is that they are limited to analysing Web content – mostly documents in natural language – one page at a time, while the Semantic Web allows for

combining data that is distributed across many different sources and described in a machine-interpretable manner.

For example, how may one pursue answering the questions related to playlists of UK radio stations? Playlists of BBC radio shows are published online in Semantic Web formats. A music group such as "Abba" has an identifier (`http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist`) that may be used to relate the music group to information at Musicbrainz (`http://musicbrainz.org/`), a music community portal exposing data on the Semantic Web [66]. MusicBrainz knows about band members such as Benny Andersson and about genre of artists and songs. In addition, MusicBrainz aligns its information with Wikipedia, for example to be able to include the biography of an artist, or to add facts from DBpedia [2], a version of Wikipedia in Semantic Web formats. Information on UK radio stations may be found in lists on web pages such as `http://www.listenlive.eu/uk.html`, which can be translated to a similar Semantic Web representation – descriptions of things and their relationships.

The meaning of such relationships are explained online, too, using a set of ontologies available on the Web, such as Dublin Core for describing general properties of information resources, SKOS for covering taxonomic descriptions, and specialised ontologies covering the music domain. Data at the BBC currently uses at least nine different ontologies, with varying degree of formality (`http://www.bbc.co.uk/ontologies/programmes`).

Given the available data, one may answer questions such as the frequency of certain music genres played on UK radio stations, radio stations playing Swedish composers, and many many more. However, having access to and leveraging such data does not come for free. The outlined scenario and likewise other use cases require generic software components, languages and protocols that must interact in a seamless manner to be able to satisfy such requests. The chapter investigates the construction of the required infrastructure at large, i.e. the Semantic Web architecture, and analyses the requirements that come from the technical need to identify and relate data, and the organisational needs to maintain the Semantic Web as a whole – even if single components shake or break.

1.2 Requirements for a Semantic Web Architecture

The following section develops functional requirements for a Semantic Web architecture from the scenario given above (cf. Section 1.2.2). Achieving such functional capabilities requires an unprecedented growth of openly available data covering a wide range of domains and involving large amounts of people and organisations. Such phenomenal and fast growth is a

non-functional (i.e., not a purely technological) requirement that has only been achieved by the World Wide Web. Thus, the first consideration is the architecture of the Web, to be able to learn from its design considerations and to derive additional non-functional requirements later on.

1.2.1 Web Architecture

The World Wide Web is the largest software system known today. Its explosive growth is tightly associated with its underlying software architecture. Important for the Web's success was that

- many people could set up a Web server easily and independently from each other;
- more people could create documents, put them online and link them to each other;
- even more people could use a browser and access any Web server (and actually some other ones, like FTP or gopher servers) for retrieving these documents.

The fact that each individual system has been coupled only very loosely with the other one and that document creation, document delivery and document browsing could happen in isolation in each of the many individual nodes was of key importance for enabling the fast adoption of the early Web. In this way the Web architecture allowed for a graceful degradation of user experience when

- the network was partially slow ("World Wide Wait"), while other parts might still operate at full speed; or when
- single web servers broke, because others would still work; or when
- hyperlinks were broken, because others would still lead you somewhere.

Thus, the Web's architecture allowed for fast growth and adoption, while other, technically more elaborate systems, such as hypertext systems from the 1980s, lacked the capabilities for mass contribution and gradual degradation. The lesson to be learned is that a state-of-the-art system that produces higher quality output (e.g. no dangling links) may be less useful to its users than a less consistent system, which achieves a separation of concern, letting users do what they are most concerned about, i.e. easily create and access documents. Furthermore, a distributed system without the need for a central coordinator is inherently robust. While there are many potential problems that may affect individual nodes in the World Wide Web, the only problem leading to brittleness of the World Wide Web as a whole is the hierarchical control of the IP addresses and the Domain Name System of the Internet.

1.2.2 Requirements for an Architecture for the Web of Data

Over time, a variety of architectures have been proposed for publishing data on the Web. Many requirements were derived from the design decisions that worked well for the World Wide Web and led to its phenomenal growth, but which had yet to be realised for data and knowledge systems.

In fact, traditional knowledge systems have already exhibited some of the functional requirements sought from the Semantic Web. However, traditional knowledge systems exhibited a lack of flexibility, robustness and scalability. To quite some extent the problem had been a lack of maturity in the face of algorithmic methods with high computational complexity. For instance, description logics systems, which are now the backbone of Web Ontologies, were severely limited in scale, typically capable of handling not more than a few hundred concepts in the mid 1990s (cf. [41]). While such problems could be assuaged in the meanwhile using much increased computational power and better understood and optimised algorithms. However, several bottlenecks remain, which are akin to the problems that the Web architecture solved for the domain of hypertext documents.

Remaining barriers for managing data and semantics revolve around issues concerning the large number of data sources with varying (1) underlying technologies, (2) geographically dispersed locations, (3) authorities, (4) service quality, and (5) adoption rate. These are exactly the dimensions that had and have to be considered for the design of the World Wide Web.

Thus, in analogy to the World Wide Web, the Semantic Web requires a computing mega system with the following five characteristics:

1. *Explicit, Simple Data Representation:* A common data representation should hide the underlying technologies and only capture the gist of the underlying data representations. Here, the analogy may be drawn with HTML documents that have served as simple, yet effective representations of what constitutes a document.

2. *Distributed System:* The system should be fully distributed, comprising of data sources without a centralised instance that controls who owns what type of information. Distributed ownership and control, if done properly, facilitates adoption and scalability, which is in analogy to Web sites and Web pages that are under full control of their producers.

3. *Cross-referencing:* In order to benefit from the network beyond the mere sum of its parts, the data must be cross-linked, allowing for reuse of existing data and existing

data definitions from different authorities, analogous to hyperlinks allowing for the reuse of text in the hypertext space.

4. *Loose Coupling with Common Language Layers:* In a mega system the components have to be only loosely coupled. The loose coupling is achieved by communicating in standardised languages. The standardised languages must come with great flexibility such that they may be customised for specific systems, but the overall communication must not be jeopardised by such specialisation. The requirement should be seen in analogy to the coupling between different Web clients and servers, where dependency is reduced to understanding HTTP as transport protocol and producing and interpreting HTML content.

5. *Ease of Publishing and Consumption:* The mega system should allow for easy publishing and consumption of simple data and for comprehensive publishing and consumption of complex data. The requirement is in analogy to the web page description language HTML that provides a simple means of conveying textual information, but that can be viewed, managed and composed using elaborate browsers and powerful content management systems.

Given these requirements, two points of view for a Semantic Web architecture emerge. One viewpoint is focused on the Semantic Web languages and protocols and is mentioned several times in the above list. Another viewpoint concentrates on the functionalities to be contributed by Semantic Web components.

Requirements for a Semantic Web Language Architecture At high level of abstraction, Semantic Web languages must address the listed requirements. Below, mandatory objectives are presented, accompanied by examples and, in parenthesis, the requirement they refer to.

First, a data model must be able to represent entities such as the group called 'Abba', the person called 'Benny Andersson', their relationship and the concrete data items, such as the string 'Benny Andersson' and the birth-date of Benny Andersson (1).

Second, such a data model must be serialisable in a standardised manner such that data becomes easily exchangeable between different computing nodes (1,2,4). For instance, without a common data serialisation, data from MusicBrainz, BBC and Wikipedia or DBpedia cannot be easily joined. Merge of such datasets may lead into forming interesting connections between playlists, music groups, artists and their origin that span across datasets.

While combining data is possible in conventional systems, such as relational databases, the emphasis on the Semantic Web is on the *ease* of joining such separate pieces of information.

Third, individual entities must be referable in such a data model across borders of ownership or computing systems, thus allowing also for the cross-linking of data (1,2,3,4). Without such cross-linking 'Benny Andersson' from ABBA might be hard to distinguish from the several 'Benny Andersson's now found on MySpace and Facebook, who also might or might not be musicians.

Fourth, the data model should have an expressive, machine-understandable data description language. In a global data space, having an expressive data description language is of major concern, because users and developers cannot longer manually inspect and make use of data descriptions due to the sheer size and heterogeneity of data on the Web (1,5). Furthermore, such a data description language also allows for a refinement of the basic data model, providing levels of specialisation needed in the application domains (4). For instance, the richness of BBC programme descriptions are hard to understand given long chains of data leading from radio stations, over shows, versions of shows, to the songs which are connected to artists.

Fifth, such a data model requires a query and manipulation language allowing for selections of data or aggregations of data such as the number of Swedish composers being broadcast on specific programmes (5).

Sixth, reasoning is desirable to facilitate querying, as it provides shortcuts for complex data situations, e.g. turning the chain of relationships between a programme and a song into a direct relationship using inference rules (5).

Seventh, the transport of data and the transport of queries and results need commonly agreed-upon protocols. While many protocol details are still under discussion, the usage of HTTP is agreed and even refined, e.g., for the transport of queries.

Eighth, such a transport requirement may also include encrypted data requests and data transport. Security of data transmission is typically addressed by encrypting the data transmission channel. On the Web, secure data transfer is achieved by HTTPS, a secure version of HTTP using SSL/TLS, established Internet protocols for encrypted data transmission. Beyond the increased security of transport, further security functionality is required, e.g. for signing data items. Such features call for a completely distributed authentication system to establish the authenticity of a user request and control access to resources.

Additional Requirements for a Semantic Web Components Architecture All requirements for Semantic Web languages imply corresponding requirements on functionality to be delivered by software components. However, some software components are not stan-

dardised (and should not be), but should be customisable to very individual user needs — up to a point where the community may recognise new chores to be done in the stack of software components that should be moved out into a joint language or joint computational model or structure.

Core requirements that are not (yet) included in the language architecture comprise:

First, versatile means for user interaction. A Web of data is not merely a web of documents and understanding cross-linked data is different to reading a document. Hence, early means of tabular interfaces constitute a stepping stone, which however is rather a crutch than a full-fledged solution for making the content of the Web accessible to users. Broad accessibility requires viewing, searching, browsing and querying for data, while at the same time abstracting from the intricacies underlying the distributed origin of the content. The users need the ability to assemble information from a multitude of sources without a-priori knowledge about the domain or structure of data. Such on-the-fly integration of multiple data sources enables new interesting scenarios for searching and interacting with the data, but may require software that is oblivious to the underlying structure.

Likewise, interaction should facilitate data production and publishing. Crucial to the success of the Web of data is that metadata creation and migration of data is made convenient, no matter whether the data originates from the content management systems, relational databases or competing metadata representations, such as microformats.

Second, the issue of provenance and trust is even more important in a Web of data than in a Web of documents. While documents still convey provenance and trust, via indications such as authorship and website ownership, corresponding notions for data are watered down once data is processed and aggregated with other data. Hence, notions of origin, reliability and trustworthiness need to be reconsidered for making them applicable to individual data items and for aggregated data sets. Furthermore, such notions are connected to a faithful authentication working at Semantic Web scale.

Third, in exploring the data and weighing its provenance and trustworthiness, one must be able to align unconnected sets of data. Beyond using identifiers, such as URI/IRIs, interlinking implies the capability to suggest alignments between identifiers or concepts from different sets of data. Only with such an alignment the full picture of a Web of data may emerge.

1.3 First Principles for a Semantic Web Architecture

A software architecture describes the structure of a software system and serves the purpose of dividing the software into components that may be developed and maintained — and maybe used — independently of each other. Thus, a software architecture gives developers, maintainers and users the possibility to care about their part of the system, while being able to communicate system properties with other people.

Unlike traditional information systems design, both the Web and the Web of data do not emphasise the specific application, but rather generic needs that can be realised in many different components, which only agree on a core set of standards. Thus, the development of the architecture of the Semantic Web first focused on protocols and languages, whereby HTTP (Hypertext Transfer Protocol) was mostly accepted as given for the Semantic Web in order to be compatible with the Web, but where the language layer was considered pivotal to the flexibility of creating and exploiting a Web of data.

Therefore, the first architectural viewpoint presented is the famous Semantic Web layer cake which is actually a Semantic Web protocols and languages layer cake. Most of the parts in the layer cake are related to data publishing and exchange.

The second architectural viewpoint considered is a functional view; thus, in the following a description of software components that provide certain functionality, such as supporting language standards or user interaction is followed by a discussion of alternative architectural choices for Semantic Web applications.

1.3.1 Protocols and Languages

The Semantic Web is rooted in a set of language specifications which represent a common infrastructure upon which applications can be built. Figure 1.1 illustrates the language standards underpinning the Semantic Web. Unlike most previous variations of the Tim Berners-Lee's Layer Cake (`http://www.w3.org/2007/03/layerCake.png`), the focus here is on the established languages that have been developed in different W3C standardisation committees (cf. Fig. 1.1) and for which stable recommendations are available.

The remainder of the section first briefly introduces each language component, starting at the bottom and progressing to the top; more detailed discussions follows later on.

Given the decentralised nature of the Semantic Web, data publishers require a way to refer to resources unambiguously. Resources on the Internet are identified with Uniform Resource Identifiers (URIs)[5]. URIs on both the Web and the Semantic Web typically use identifiers based on HTTP, which allows for piggybacking on the Domain Name System (DNS) to ensure

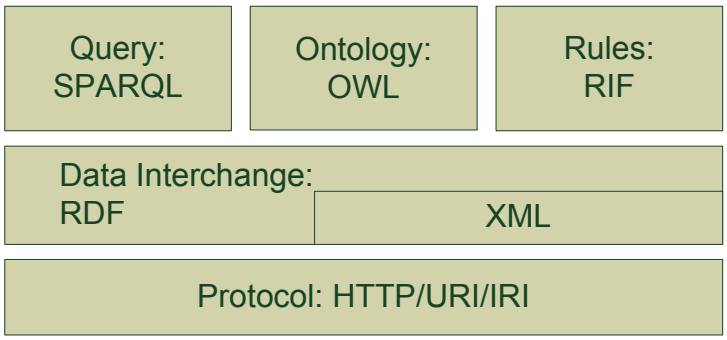

Figure 1.1: Semantic Web Protocol and Language Standards.

the global uniqueness of domain names and hence URIs. In the example, the URI `http://www.bbc.co.uk/music/artists/2f031686-3f01-4f33-a4fc-fb3944532efa#artist` denotes Benny Andersson of ABBA. Internationalized Resource Identifiers (IRI)[28] complement URIs and allow for use of characters from a large range of writing systems in identifiers.

Implicit in the use of URIs is a mechanism for retrieving content; assuming an HTTP URI denoting ABBA, a user has the ability to dereference the URI, that is, perform a lookup using HTTP. Given there is content hosted at the URI, the user retrieves content which is associated with the URI.

The ability to point to resources unambiguously and dereference them is a first step. Next, required is a language to exchange description of resources. The Extensible Markup Language (XML) is used for encoding documents and provides means for specifying and serialising structured documents which can be parsed across operating systems.

Building on a referencing and a document exchange mechanism, means to encode description about resources are required. Given that the data on the Semantic Web is highly distributed, the description of resources should be encoded in a way that facilitates integration from a large number of sources. A graph-structured data format [62] achieves the easy integration of data from multiple sources. The W3C standard for encoding such data is the Resource Description Framework (RDF). RDF graphs can be serialised in multiple ways; one of the most commonly used is the XML serialisation.

Having integrated data, mechanisms for querying the integrated graphs are necessary. SPARQL (a recursive acronym for SPARQL Protocol and RDF Query Language) is a declarative query language, similar to SQL, which allows for specifying queries against data in RDF.

Encoding data as graph covers only parts of the meaning of the data. Often, constructs to model class or property hierarchies provide machines and subsequently humans a more sapient understanding of data. To more comprehensively model a domain of interest, so-

called ontology languages can be employed. RDF Schema (RDFS) is a language which can be used to express for example class and property hierarchies as well as domain and range of properties. Since data originates from multiple sources and is thus highly heterogeneous, means to reconcile data from different sources and to check consistency of combined data are required. The Web Ontology Language (OWL) allows for specifying equality of resources or cardinality constraints of properties, for example. Ontology languages allow for automated inferences, i.e. drawing conclusions based on existing facts.

An alternative way to specifying logical inferences is via rules. Often, users require means to express logical rules which transform data or enrich data with additional specifications. The Rules Interchange Format (RIF) allows for encoding and exchanging such logical rules.

1.3.2 Software Components

One way of viewing the Semantic Web architecture is via the standards and languages used. Another view is via software applications and components that implement functionality based on the standards and languages. Figure 1.2 represents the components typically used as Semantic Web infrastructure when implementing functionality in applications. For a survey of Semantic Web applications described in the literature and the components they implement see [42].

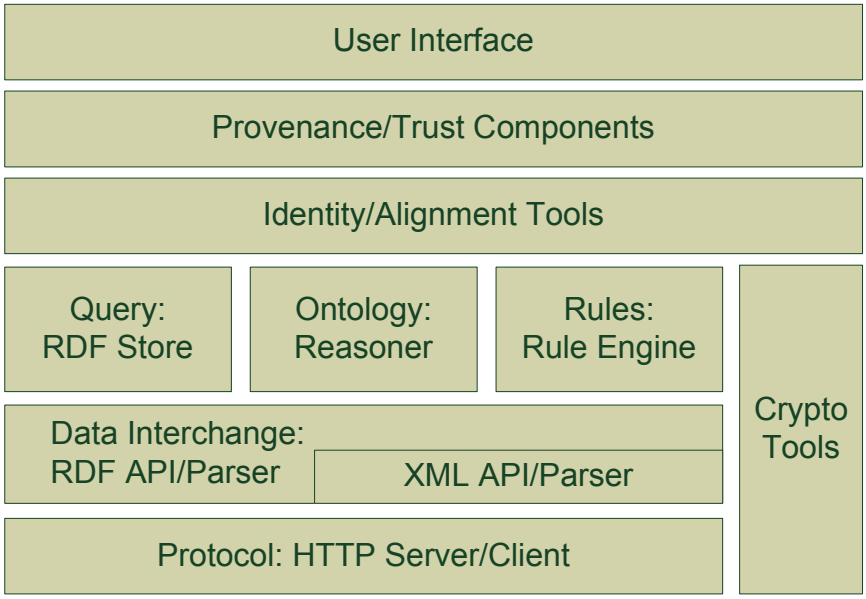

Figure 1.2: Semantic Web Components Architecture.

The basic infrastructure components pertain to publishing RDF and dereferencing content. URIs serve as unique identifiers, however, per convention URIs often can be also used

to retrieve content pertaining to the resources they identify. Web servers provide the infrastructure for serving content via HTTP, and HTTP clients provide lookup functionality. The Semantic Web inherits the basic infrastructure for referencing and lookup from the Web.

Content returned as the result of HTTP lookups is either processed using XML parsers and APIs (in case of non-RDF content) or RDF parsers and APIs. While the W3C developed a specification for manipulating XML documents (the Document Object Model), no such standardised specification exists for processing RDF content. However, there are several open source implementations of APIs for RDF, which are covered in 1.4.2.

Having gathered RDF data, typical applications use RDF repositories for storing and indexing RDF data and processing SPARQL queries. Ontology languages such as RDFS and OWL add more expressivity to RDF data, and so-called reasoners are able to process ontologies and draw specified inferences. Similarly, rule engines allow for processing rules exchanged in RIF.

A crucial point for transmitting sensitive (personal) data is to ensure the that data transmissions cannot be intercepted, read or altered. Crypto tools cover encryption and authentication technologies to ensure the secure exchange of data. Crypto modules such as SSL processors verify digital certificates and provide cryptographic privacy and authentication.

Given that content aggregated from a large number of sources often use multiple identifiers to denote the same real-world object, an integration and alignment layer provides for consolidation and tighter integration of data. The provenance and trust layer analyses the data in conjunction with additional information to provide the user with a notion of trust associated to individual data items.

Finally, the user interface enables users to interact with Semantic Web data. From a functionality viewpoint, some user interfaces are generic and operate on the graph structure of the data, while others are tailored to a certain domain and ontology.

1.3.3 System Architecture Styles

In general, one can distinguish between two basic architectural styles for applications that consume Semantic Web data [37]: i) gather and pre-process data a priori, similar to web search engines [14] and data warehouses, or ii) integrate data on demand at query time, similar to database approaches [55].

In the pre-processing approach, web crawlers or scripts collect large amounts of data and indexers prepare the entire corpus for fast lookups. The architecture of web crawlers has been adapted to the intricacies of Semantic Web data (e.g. [38]); also, tools exist that allow for converting metadata embedded in a variety of formats such as Microformats to RDF. In some

of the warehousing systems, reasoning modules ensure that all inferences on the gathered data are computed a priori and then the queries are answered against the materialised version of the dataset. Typical large-scale Semantic Web search engines [24, 44, 22, 20, 59] use the pre-computed approach where all data is collected and preprocessed.

The on-demand query model, on the other hand, is used in meta-search engines [73] and distributed query processing systems, which start with the query (or goal) and iteratively collect the required data from a set of distributed sources. Meta-search engines collect and integrate results from several search engines during query execution time. Distributed query processing systems [40, 50] use source descriptions to decide which source can answer parts of a given query, and delegate (parts of) the initial query to the appropriate source. In a similar vein, reasoning systems that employ so-called backward chaining (used in e.g. XSB [68]) start with a goal and iteratively find rules or data that contribute answers. Semantic Web applications that use the on demand query model are SemaPlorer [69] and DARQ [65].

Deciding on which architecture to select for applications depends on the requirements of the use case. The warehousing model provides fast query response times due to the large amount of preprocessing involved, but suffers a number of drawbacks. First, the aggregated data is never current as the process of collecting and indexing vast amounts of data is time-consuming. Second, from the viewpoint of a single requester with a particular query, there is a large amount of unnecessary data gathering, processing, and storage involved since a large portion of the data might not be used for answering that particular query. Furthermore, due to the replicated data storage, the data providers have to give up their sole sovereignty on their data (e.g., they cannot restrict or log access any more since queries are answered against a copy of the data).

On-demand query processing offers several advantages: the system is more dynamic with up-to-date data and new sources can be added easily without time lag for indexing and integrating the data, and the system requires less storage and processing resources at the query issuing site. The drawback, however, is that such systems cannot give guarantees about query performance since the integration system relies on a large number of potentially unreliable sources, and that the same work has to be done repeatedly.

1.4 Building Blocks

The following section covers the functional software components – some of them implementing W3C recommendations – which are used for data publishing and consumption.

1.4.1 Referencing, Transport Protocol and Linked Data Principles

Data access is a vital component of the architecture of the Semantic Web. The successful traditional client-server model has been adapted to the Web at a grand scale where clients can easily connect to and transfer data from a multitude of servers. Documents on servers are interlinked with each other in a decentralised manner, and clients use these links to navigate from server to server. The model has been condensed to the necessary functionality, and loose coupling between one server and another is achieved by (unilaterally) linking from a document on one server to a document on another server. Actually, whether the referenced document is located on the same server or a server on a different continent does not make a fundamental difference in the process of linking.

URI/IRI and HTTP are the core specifications for both the Web and the Semantic Web, and enable decentralised referencing and transport of content. Uniform Resource Identifiers (URI) are used to identify resources on the Web, for example, `http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist` can be used to denote the music group ABBA. International Resource Identifier (IRI) extend URIs with Unicode support.

For URIs on the Semantic Web, there exists a distinction between information resources (i.e. those URIs which are webpages such as the page describing ABBA) and non-information resources (i.e. denoting ABBA the music group, or an abstract concept such as "the integers") [35]. It is important to not confuse identifiers of documents with identifiers of things, e.g. `http://en.wikipedia.org/wiki/ABBA` is the identifier of the document containing a textual description of ABBA while `http://dbpedia.org/resource/ABBA` represents the music group ABBA itself.

Having the ability to reference things via URIs forms the basis for identity on the Web. Sources other than the authoritative source (i.e. the source which is associated to a URI via syntactic or protocol means) can reuse external URIs, which helps to construct a Web of data. For example, the content about ABBA at the BBC may reuse DBpedia's URI for ABBA, thus establishing an association between its own notion of ABBA and the notion of ABBA from DBpedia.

A URI starts with a scheme name (e.g. `http`, `ftp`, `tel`, `mailto`) followed by additional information. The example in this chapter only uses URIs with the Hypertext Transfer Protocol (HTTP)[31] scheme. HTTP URIs have the benefit that when dereferenced – i.e. when a HTTP request on them is performed – they return some form of content. In the Semantic Web, the content returned is typically in RDF. HTTP allows for mechanisms to specify additional parameters when requesting content. For example, clients can ask for

content in certain formats to be returned from the web server using the `Accept` header, which specifies the preference for certain formats.

Data in the music scenario is published by the BBC, DBpedia, and MusicBrainz. All these data publishers use a subset of the protocols and languages shown in Figure 1.1. Data is freely interconnectable, meaning that groups can publish data and interlink their data with others' without direct coordination. For example, the BBC links their ABBA URI to the DBpedia equivalent without the need for DBpedia to permit the link or configure anything on their side.

Data publishers on the Semantic Web typically use Linked Data principles [6]:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs, so that they can discover more things.

Data publishing adhering to these simple principles leads to a unified system for data access and interlinkage. Assuming that all the data publishers use these standards, data consumers can use those standards (as clients) to access the data and further process the data. The so-called Linked Data cloud of Semantic Web data [21] has been growing considerably in the past years and provides a foundation upon which applications leveraging that data can be built. Wrappers (a special case of mediators [80]) allow for accessing legacy data as Linked Data. For example, D2R [10] provides a mechanism for exposing data stored in relational databases as Linked Data. Custom-build wrapper software often provides Linked Data access to Web sources returning JSON or XML.

HTTP is based on the representational state transfer (REST) [30] request/response model, which can be extended to query/answer for data. This model decouples data providers from data consumers allowing clients to use uniform interface for accessing and querying data stored on remote servers. In the RESTful approach information about resources identified by URIs can be accessed directly using simple HTTP GET method, or if site implements SPARQL endpoint, clients can post powerful SPARQL queries against the published data. The Linked Data and data access models are presented in more details in Chapter 6 "Semantic Annotation and Retrieval: Web of Data."

1.4.2 Data Interchange

Given that data on the Semantic Web is created by people across the planet who do not necessarily coordinate when creating their data, the architecture needs to allow for the distributed creation of content while allowing for integration of and interoperation between the created data. Graph-structured data models (e.g. [62]) have been identified as candidates for integration of data from disparate sources. RDF is such a graph-structured data model that uses URIs as identifying mechanism, and provides the means for the separate modelling and publishing of data which can be interlinked and integrated in subsequent steps. RDF is a data format based on directed labelled graphs, which contains data items consisting of (subject, predicate, object) triples. In such triples, subject, predicate and object can be either URIs or, depending on which position an element is located at, literals (string such as "ABBA" or numbers or dates) or blank nodes – identifiers which are local to a given graph and cannot be referenced from outside. The RDF specification includes a special property `rdf:type` which is used to model instances (for example, ABBA) and classes (for example, the class MusicGroup). A graph encoding a description of ABBA is shown in Figure 1.3.

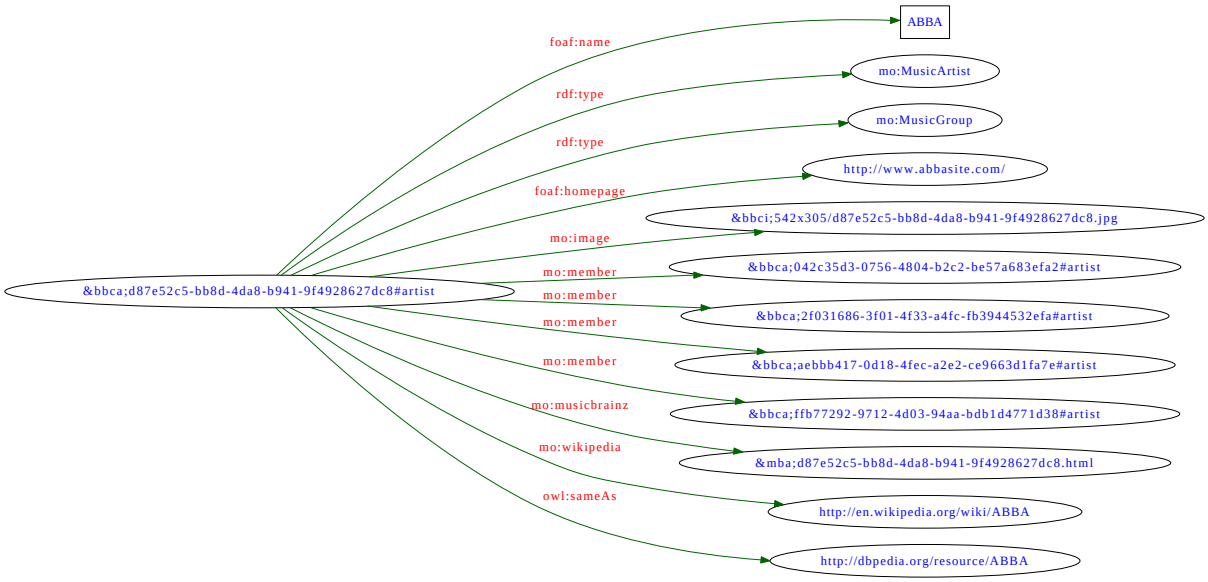

Figure 1.3: RDF Graph describing ABBA (partial content of `http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8.rdf` as of December 10, 2009). URIs are abbreviated using RDF/XML syntax from Figure 1.4.

RDF is an abstract data format which can be written in multiple serialisations. One of the serialisations is XML, the Extended Markup Language, which is shown in Figure 1.4. Other serialisations include Turtle, a format used in Figure 1.5 which encodes RDF in a triple language and allows for shortcuts such as "`;`" for repeating subjects or "`,`" for repeating

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
<!ENTITY bbca "http://www.bbc.co.uk/music/artists/">
<!ENTITY bbci "http://www.bbc.co.uk/music/images/artists/">
<!ENTITY mba "http://musicbrainz.org/artist/">
]>

<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:owl="http://www.w3.org/2002/07/owl#"
 xmlns:foaf="http://xmlns.com/foaf/0.1/"
 xmlns:mo="http://purl.org/ontology/mo/">

 <mo:MusicArtist rdf:about="&bbca;d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist">
  <rdf:type rdf:resource="http://purl.org/ontology/mo/MusicGroup"/>
  <foaf:name>ABBA</foaf:name>
  <foaf:homepage rdf:resource="http://www.abbasite.com/"/>
  <mo:image rdf:resource="&bbci;542x305/d87e52c5-bb8d-4da8-b941-9f4928627dc8.jpg"/>
  <mo:member rdf:resource="&bbca;042c35d3-0756-4804-b2c2-be57a683efa2#artist"/>
  <mo:member rdf:resource="&bbca;2f031686-3f01-4f33-a4fc-fb3944532efa#artist"/>
  <mo:member rdf:resource="&bbca;aebbb417-0d18-4fec-a2e2-ce9663d1fa7e#artist"/>
  <mo:member rdf:resource="&bbca;ffb77292-9712-4d03-94aa-bdb1d4771d38#artist"/>
  <mo:musicbrainz rdf:resource="&mba;d87e52c5-bb8d-4da8-b941-9f4928627dc8.html"/>
  <mo:wikipedia rdf:resource="http://en.wikipedia.org/wiki/ABBA"/>
  <owl:sameAs rdf:resource="http://dbpedia.org/resource/ABBA"/>
 </mo:MusicArtist>
</rdf:RDF>
```

Figure 1.4: RDF describing ABBA serialised in RDF/XML

subject/predicate pairs. A recent development is RDFa which allows for embedding RDF statements directly into (X)HTML documents [1]. To be able to process RDF in application programs, parsers such as ARP from the Jena project [53] or parsers part of the Redland RDF library [4] can be used. These libraries also contain repositories which allow for storing and retrieving RDF. RDF is further described in Chapter 4 "Semantic Annotation and Retrieval: RDF."

1.4.3 Querying, Updating and Views

Data access is an important component in the architecture of the Semantic Web. The Linked Data principles (c.f. 1.4.1) allow for publishing and accessing simple facts, however they do not support more complex queries, because data published using the Linked Data paradigm does not have to be accompanied by more sophisticated query mechanisms. Consider the example from Musicbrainz with a query for information about ABBA. Now, consider a query asking for singers from ABBA that were also members of other music bands, as such a situation not unusual among musicians. Although a software program could navigate from the URI describing ABBA to each of its members, and later to all bands she or he was a member of, iteratively dereferencing individual URIs could be too time-consuming for certain use cases. Furthermore, if some of the URIs do not point to real-world web addresses, it is

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix mo: <http://purl.org/ontology/mo/> .

<http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist>
    rdf:type mo:MusicArtist, mo:MusicGroup ;
    foaf:name "ABBA" ;
    foaf:homepage <http://www.abbasite.com/> ;
    mo:image <http://www.bbc.co.uk/music/images/artists/542x305/d87e52c5-bb8d-4da8-b941-9f4928627dc8.jpg> ;
    mo:member <http://www.bbc.co.uk/music/artists/042c35d3-0756-4804-b2c2-be57a683efa2#artist>,
              <http://www.bbc.co.uk/music/artists/2f031686-3f01-4f33-a4fc-fb3944532efa#artist>,
              <http://www.bbc.co.uk/music/artists/aebbb417-0d18-4fec-a2e2-ce9663d1fa7e#artist>,
              <http://www.bbc.co.uk/music/artists/ffb77292-9712-4d03-94aa-bdb1d4771d38#artist> ;
    mo:musicbrainz <http://musicbrainz.org/artist/d87e52c5-bb8d-4da8-b941-9f4928627dc8.html> ;
    mo:wikipedia <http://en.wikipedia.org/wiki/ABBA> ;
    owl:sameAs <http://dbpedia.org/resource/ABBA> .
```

Figure 1.5: RDF describing ABBA serialised in Turtle

not even possible to find an answer for such a query.

The SPARQL Query Language for RDF [74] is designed for evaluating queries against RDF datasets and designed to handle complex structure queries, typically over data stored in RDF repositories. The repository that supports SPARQL must implement querying of the underlying data using a specific syntax and protocol. With RDF repositories, access to the information is no longer realised by dereferencing individual URIs for RDF files, but by posing queries to SPARQL endpoints. SPARQL allows a user to specify arbitrary URIs (even those not accessible on the Web) and a graph pattern that should be matched against the knowledge base together with additional constraints. The example graph pattern for asking about different bands that musicians from ABBA were singing in is presented in Figure 1.6.

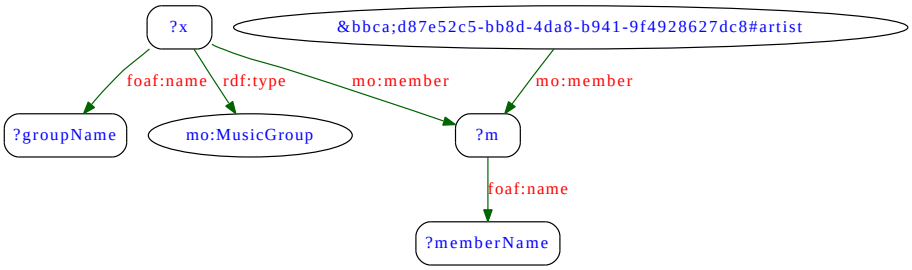

Figure 1.6: Graphical representation of the WHERE clause of the query for music groups that members of ABBA sing in

The query can be written in SPARQL, as presented in Figure 1.7. A SPARQL query consists of sections that define different aspects of the query. PREFIX is used to abbreviate URIs, mostly for clarity and to improve readability of the graph pattern. In the SELECT section, users can specify the exact information they are interested in. There is no longer a

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?memberName ?groupName
WHERE  { <http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist> mo:member ?m .
         ?x mo:member ?m .
         ?x rdf:type mo:MusicGroup .
         ?m foaf:name ?memberName .
         ?x foaf:name ?groupName }
FILTER (?groupName <>  "ABBA")
```

Figure 1.7: SPARQL query for music groups that members of ABBA sing in

limitation to one resource and its description as with Linked Data lookups on URIs. Please note that the example query shall return only in the names of musicians and bands, not the URIs describing them.

The core of SPARQL is contained in the WHERE clause. Here, users define the exact graph pattern that has to be matched against knowledge base. A basic graph pattern consists of individual (subject, predicate, object) patterns which are joined by variables, forming a template that will be filled during the matching process. In the example joining resources include the unknown band and members of ABBA. The WHERE clause is used for matching the structure of the graph. Optionally, the WHERE clause is followed by a FILTER expression that narrows the returned results only to the structures that fulfill specific criteria. The example query filters the names of music bands other than ABBA.

SPARQL can be implemented over several kinds of graph repositories. Popular repositories include Sesame [16], Jena [53], Virtuoso [78], BigData [9], OWLIM [61] or RDF-3X [56]. Some of the repositories do not only provide storage and query access for RDF graphs, but also support querying with use of inference and rules. SPARQL lacks a a syntax for expressing inference. It is the role of the underlying repository to support a corresponding inference model.

Since data is often stored in relational databases, wrappers are used to provide SPARQL access to data stored in relational databases or accessible via an API. Examples include D2R [10] and Triplify [3]. Using such bridges, legacy relational datasets can be exposed and queried as semantic graphs. Such wrappers are of high importance, as they enable smooth transition from the relational model to graph-based processing.

With the recently proposed version of SPARQL 1.1 [48], new features are introduced in the language, which include aggregate functions (such as minimum, maximum, sum), sub-queries (nesting of SPARQL queries), negation and project expressions (e.g. providing a result computed by multiplication of values). Further information about query-specific extensions accompanied by examples can be found at `http://www.w3.org/TR/sparql11-query`.

In addition to the query language, SPARQL defines the access protocol and interoperability data formats. Datasets can be exposed via a SPARQL endpoint to the outside world, accessible via HTTP requests. The access protocols enable remote access to the data and frees the data from closed data-silos. In the example, MusicBrainz does not only publish the information as Linked Data, but also provides a query endpoint accessible via HTTP to post SPARQL queries (`http://dbtune.org/musicbrainz/snorql`), which facilitates the use of remote repositories and releases the user from the burden of downloading the dataset locally to be able to pose queries.

Currently, the SPARQL standard provides solution for querying a single knowledge base. It is possible to remotely query MusicBrainz for information about ABBA and later send a query to DBpedia for the biography of each of the band members. Yet to do this, two separate queries have to be asked and the user is in charge of merging the results. New language constructs are proposed in SPARQL to handle the querying of multiple (remote) repositories, but the federated query feature is not defined yet in the standard.

Before the new standard is settled, some solutions to federated queries have been already proposed to address the need of querying multiple repositories. The foundations for querying distributed RDF repositories were stated by Stuckenschmidt et. al. [77]. There, the authors proposed a mediation architecture, index structures and algorithms for executing distributed path queries. The approach was further refined and extended in Networked Graphs [70]. Networked Graphs do not only allow for querying of remote repositories in a unified manner and building dynamic views of the remote graphs, but also for joining them together and using recursive views (defined as CONSTRUCT queries), and for applying rules. The requirement is that each of the graphs in the query has to be accessible via a SPARQL endpoint. The SPARQL query that uses the Networked Graphs framework for the extended Wikipedia pages of artists singing in ABBA is presented in Figure 1.8.

```
PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
CONSTRUCT { ?member mo:wikipedia ?biography . ?member foaf:name ?name}
FROM NAMED :Musicbrainz   FROM NAMED :DBpedia
WHERE {
    GRAPH :Musicbrainz {
        <http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist> mo:member ?member .
        ?member foaf:name ?name }
    GRAPH :DBpedia {
        ?member mo:wikipedia ?biography }
}
```

Figure 1.8: SPARQL query using Named Graphs for Wikipedia pages of ABBA members

The query accesses two graphs and presents joined results to the user. CONSTRUCT queries produce new RDF graphs that either can be presented to the user or fed as data source

to the next query. The implementation of Networked Graphs takes care of distributing proper parts of a query to specific remote SPARQL endpoints (depending on a configuration) and joining the final result. From the user perspective, it is executed as a single query extracting data from two named RDF graphs, hiding the complexity of accessing remote repositories and creating a combined result.

The result of such CONSTRUCT query can be treated as a dynamic, customisable view of the underlying data and allows for presenting data from the connected repositories in a specified, even restricted form. With mechanisms such as Named Graphs it is possible to hide the heterogeneity of data sources and schemas, restrict access to specific resources, or redefine types of relationships between entities. It can serve the same purpose as SQL view in relational databases. As the view mechanism produces RDF graphs as result, it can be later used as a single data source for constructing another view.

In addition to accessing data, the newly proposed version of SPARQL Update [72] introduces methods for data and graph manipulation in the form of *inserts* and *deletes*. This language accompanies SPARQL, using its syntax and protocol to express updates to RDF repository. Until now all updates to data in the repository had to be performed using storage-specific tools. With the SPARQL Update standardisation, changes to the semantic data in repositories do no longer require use of third-party applications. Statements can be added or removed from the repository using the SPARQL language. Chapter 8 "Querying the Semantic Web: SPARQL" contains more detailed description of the SPARQL query language and protocol.

1.4.4 Ontology and Reasoning

Semantics can emerge in two ways: the first is a social one where meaning arises via a common understanding in a group of people using shared identifiers (covered in Section 1.4.7). The second way for encoding meaning is by the use of logical constructs. The standards enabling reasoning on the Semantic Web are RDF Schema [15], the ontology language, OWL (Web Ontology Language)[34], and RIF (Rule Interchange Format) [12], covered in the next section.

To illustrate the use of reasoning, i.e. drawing conclusions from existing facts in the Semantic Web, a selection of ontology constructs are explained based on our example – namely `rdfs:subClassOf`, `owl:sameAs` and property chains.

The `rdfs:subClassOf` construct can be used to model class hierarchies. Consider, for example, the Music Ontology which specifies two classes, `mo:MusicGroup` and `mo:MusicArtist` and contains an axiom specifying that `mo:MusicGroup` is a subclass of `mo:MusicArtist` via

the rdfs:subClassOf property. Such a construct allows a reasoner to deduce that instances of mo:MusicGroup are also of type mo:MusicArtist. In the example, given the axiom and a fact stating that ABBA is a mo:MusicGroup, a reasoner can draw the conclusion that ABBA is also of type mo:MusicArtist and applications that query for all mo:MusicArtists also get ABBA as a query result even if the instance does not explicitly contain that type assertion.

Another construct is that of owl:sameAs which can be used to specify that two resources are identical. For example, one could state that http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist is the same as http://dbpedia.org/resource/ABBA, which allows a reasoner to consolidate information about ABBA from multiple sources. Using owl:sameAs instead of reusing the same URI across sources allows for sources being connected after they coined the URIs for the respective thing. In addition, by coining separate URIs sources one may provide information about the resource via HTTP lookups on the URI.

Property chains, which have been added to the recent OWL 2 specification, are useful to collapse property paths. To find out which show played which songs, on the original BBC data one must have access to the timeline of the show, which requires traversing a path of four properties. Using an axiom as shown in Figure 1.9, a reasoner can collapse that lengthy path into a single predicate, facilitating access to data via the specified shortcut.

```
<rdf:Description rdf:about="#hasTimeline">
   <owl:propertyChainAxiom rdf:parseType="Collection">
       <owl:ObjectProperty rdf:about="&dc;relation"/>
       <owl:ObjectProperty rdf:about="&po;version"/>
       <owl:ObjectProperty rdf:about="&po;time"/>
       <owl:ObjectProperty rdf:about="&timeline;timeline"/>
   </owl:propertyChainAxiom>
 </rdf:Description>
```

Figure 1.9: OWL2 Axiom encoding a property chain

OWL enables also more complex modelling constructs, e.g. cardinality restrictions (a predicate can only have a certain number of objects) or disjoint classes (the set of instances of two classes are disjoint). Specialised software systems, so-called reasoners, are able to take OWL knowledge bases and i) check for the consistency of the knowledge base and ii) infer new statements based on existing ones. More details on ontology reasoning can be found in Chapter 9 "Querying the Semantic Web: SPARQL".

1.4.5 Rules and Rule Engines

Another mechanism for drawing conclusions from existing knowledge are logical rules (e.g. know from Prolog). Rules consist of two parts, antecedent and consequent: if the statement in the antecedent is true, the statement in the consequent follows. RIF is the W3C recommendation for exchanging rule sets between rule engines. The common subset of a large number of rules systems are standardised in RIF Core [12] which forms the basis for more expressive languages, notably Prolog-style and Production Rule languages. Figure 1.10 shows two mapping rules relating predicates in the FOAF vocabulary [39]. Rules are presented in more detail in Chapter 10 "KR and Reasoning on the SW: RIF."

```
if { ?x foaf:firstName ?first;
    foaf:surname ?last }
then
   { ?x foaf:family_name ?last;
     foaf:givenname ?first;
     foaf:name func:string-join(?first " " ?last)
   }

if { ?x foaf:name ?name } and
     pred:contains(?name, " ")
   then
   { ?x foaf:firstName func:string-before(?name, " ");
     foaf:surname func:string-after(?name, " ")
   }
```

Figure 1.10: Two RIF rules for mapping FOAF predicates.

1.4.6 Security and Encryption

In open systems such as the Internet, where data is transfered across unsecured links involving infrastructure maintained by a large number of organisations, mechanisms for secure exchange of data have to be established. In addition, technology has to be in place to ensure the authenticity of documents via digital signatures. Further, to establish the identity of users, authentication mechanisms are required. These issues are addressed in the architecture of the (Semantic) Web in the crypto layer.

To make sure data is not altered during transmission, HTTPS, a secure version of HTTP, has been developed [67], using a different server port compared to HTTP and an encryption protocol to counteract man-in-the-middle attacks and eavesdropping of connections on the level of the transport layer.

Digital signing of RDF graphs ensures the authenticity of content - that the content at hand has been created by a known source and that the content has not been altered. Signing

RDF documents consists of a normalisation step which returns a canonical representation of RDF triples which then are signed using standard digital signature methods [17].

For establishing the identity of a user when logging into a web site or service, two similar mechanisms are currently in use. With OpenID [58], users are redirected from a service provider to an identity provider which authenticates the user and returns the credentials of the user to the service provider. In contrast to OpenID, FOAF+SSL [76] is a distributed mechanism based on digital certificates and RDF which allows for authentication using information from published and signed FOAF profiles. In this model, FOAF friendship network backed with the cryptographic keys and signatures provides information for successful authentication of a user identified by his or her FOAF URI.

1.4.7 Identity Management and Alignment

Identity is related to the social meaning of semantics and concerns the question of what identifiers mean and how to ensure the authenticity of an identifier (i.e. how to know an identifier is "the right one" to stand for a real-world entity). The issue of identity is substantial in the context of the Semantic Web, given that the system is decentralised and people may mint own identifiers for resources, which leads to a plethora of identifiers for the same real-world entity. Reusing identifiers across sources allows for discovery and navigation in such a decentralised environment. Using instance URIs or class URIs coined by other sources helps to weave a web where distributed discovery is possible. For example, the BBC linking to `http://dbpedia.org/resource/ABBA` establishes an association with BBC's description of ABBA and the one at DBpedia.

Indications for identity in the sense of object equality can be explicitely stated via reusing identifiers across sources, either directly or via connecting an own resource with an already existing one (e.g. via `owl:sameAs` properties on the instance level or `rdfs:subClassOf` on the class level).

On the Semantic Web, anybody can use RDF descriptions to attach certain pieces of information to newly minted or existing URIs. To be able to consolidate information from multiple sources the identity of URIs has to be established. In the simplest case, when two sources attach RDF descriptions to the same URI, a syntactic check can be used to merge the data. At data creation time the data publishers have to be aware of the existence of a URI (and what entity it refers to) and therefore decide that they mean the same thing, reuse the URI and thus subscribe to the meaning of the URI. The data publishers can then attach new descriptions to the URI. For example, to add a new photo of ABBA one could simply publish a triple as shown in Figure 1.11. Systems such as `http://sameas.org/` or

the Okkam Entity Server [13] provide services for determining equivalent identifiers. For a given identifier, if it is known to the system, they provide a list of co-referant URIs. It makes possible to locate the same entity in different systems, although it can be referred by multiple different URIs.

```
<http://dbpedia.org/resource/ABBA> foaf:depiction
          <http://farm3.static.flickr.com/2137/2203448820_c66459d45a_o_d.jpg> .
```

Figure 1.11: Annotating an existing URI

Data publishers can also relate their identifiers to already existing ones, which is in-line with Linked Data principles. Interlinking data and reusing concepts already modelled in existing ontologies facilitates data integration and reasoning. The BBC publishing a triple linking their concept of the instance ABBA to the one at DBpedia via the `owl:sameAs` property is such an example. A case of relating own identifiers to already existing ones on the data description level is the Music Ontology stating that `mo:MusicArtist` is an `rdfs:subClassOf foaf:Agent`. By that, the creators of the Music Ontology relate their concept of MusicArtist to the more general and established concept of `foaf:Agent`.

Another way to establish sameness of identifiers is via reasoning. For example, OWL allows for specifying so-called inverse functional properties which uniquely identify a thing, such as an ISBN for books or passport numbers for people. Reasoners can be used to establish the sameness of two URIs if they share the same value for a property which is defined as inverse functional.

While it is desirable to have associations between data sources available, disparate sources often lack links between them. To allow for querying and processing data from these sources in an integrated manner, mappings between identifiers have to be found. The Silk [79] linking framework allows for a declarative specification of how resources relate to each other and thus provides a semi-automatic way of matching identifiers across sources. Other approaches, such as [18], allow for automated instance matching based on defined similarity metrics.

Work on ontology matching has mostly focused on aligning class- and property-level identifiers. For example, [27] presents algorithms to align taxonomies with each other. For a comprehensive review of state-of-the-art methods for ontology matching see [29]. The Ontology Alignment Evaluation Initiative [57] holds annual events to evaluate systems and publish matching accuracy of participating systems.

The graph-structured nature of RDF allows for the amalgamation of data from disparate sources. If the data sources do not share identifiers between each other, instance and ontology matching can provide the tight integration that more elaborate application scenarios require.

1.4.8 Provenance and Trust

On the Semantic Web, not all data is created equal and in order to judge the value of the data items and how to use data appropriately one has to know the origin of data. Data provenance can be traced back in various ways. Along formal chains of information processing, such as queries, one may trace the data lineage (as data provenance is called under these specific circumstances) using a formal machinery [25, 32]. These models can provide explanation to questions, such as: which pieces of data were chosen and composed together, where do they come from, who created them, or which inference rules were used to create implicit statements.

Along chains of data processing, where the semantics of the processing are not fully specified, one needs more abstract models to trace data provenance. Such a more abstract model is given by workflow provenance models such as the Open Provenance Model (OPM) [54]. In these models, stages of data processing can also be black boxes (unlike in data lineage tracing) and provenance includes also information about processes constraints, interactions and dependencies. Such meta-information is crucial for understanding what happened in the whole chain of data manipulation, which process produced certain results, or who and when initiated them. An example of a workflow provenance with inference (dotted lines) in the Open Provenance Model is shown in Figure 1.12.

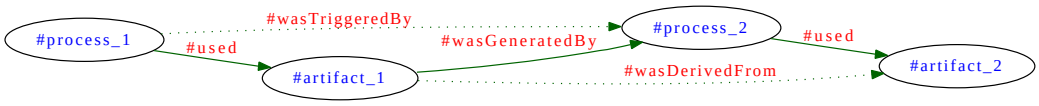

Figure 1.12: Inference in the Open Provenance Model

Based on provenance one may also attribute other properties, such as trust on data. There are multiple approaches to modelling trust and trust propagation. In data lineage, trust is related to information like data source, authorship, certainty and other dimensions of meta-knowledge. Queries that combine data of different certainty or from multiple data sources need a specific algebra to calculate the overall trust of a result. Such a comprehensive algebra for querying and reasoning with different dimensions of meta-knowledge is described in [26]. Using the proposed algebra, certainty can be calculated for multiple joined statements and graphs, providing a trust value for larger structures.

In peer-to-peer networks peers must decide if a node is reliable or malicious. Trust is related to the reputation of the node, and modelling requires both a notion of trust and distrust. Trust to the specific node (reputation) reflects the experiences of all peers in the network interacting with the node and each peer gains a view of the network that is wider

than its own experience [46].

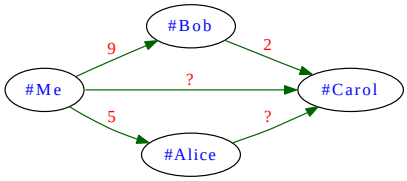

Figure 1.13: Example of friendship relations in a user network with assigned trust values

In social networks, trust follows the friendship relationship. Users define their level of trust to the direct neighbours. An example of Me trusting Alice and Bob, and Alice and Bob trusting Carol is shown in Figure 1.13. Trust to an unknown person (Carol) is based on trust defined in own friendship network, and further on trust of friends of their friends [33].

Provenance and trust provides the way for verification of data and information. In an open environment, such as Semantic Web, where everybody can publish any data without limitation it helps to distinguish fake information from the genuine data.

1.4.9 User Interaction and Visualisation

User interfaces over Semantic Web data are still in their infancy, given that sizable amounts of quality RDF data have only been recently made available online. Current challenges for building user interfaces over Semantic Web data are: interacting in new ways, heterogeneous data, and data creation [71]. Dealing with data rather than documents (as in tradtional Web search engines) poses several new challenges for interaction design. Data collected from thousands of sources on the Web exhibits far greater variance in inter-linkage and vocabularies used than relational or single-source datasets. In addition, a Semantic Web user interface should allow users to change and create new data rather than just allow for viewing and browsing. In the following each challenge is discussed in detail.

The basic paradigm for interacting with data is query/answer: users construct queries and the system returns answers. Thus, a system enabling users to interact with data has to provide means for posing queries and displaying the results that the system returns. General-purpose user interfaces can offer access to data using keywords, natural language, iterative menu-guided mechanisms, or graphical query languages [47]. Iterative menu-guided systems are used in the majority of current user interfaces over Semantic Web data. In such systems, users can build queries of different expressiveness. While systems such as the Disco Hyperdata Browser [11] allow for traversing the RDF graph one resource at a time, other systems implement more expressive query functionality such as faceted browsing [81, 60, 49, 36, 75].

Since the answer that these systems return comprises of data – typically a subgraph in the case of Semantic Web data – the results can be displayed using appropriate visualisations. Vispedia [19] is a Web-based visualisation system operating over data from DBpedia in an interative, interactive data exploration process, which enables a broad class of non-expert users to use data extracted from Wikipedia. Figure 1.14 shows a map visualisation of the North American cities hosting a SIGMOD conference.

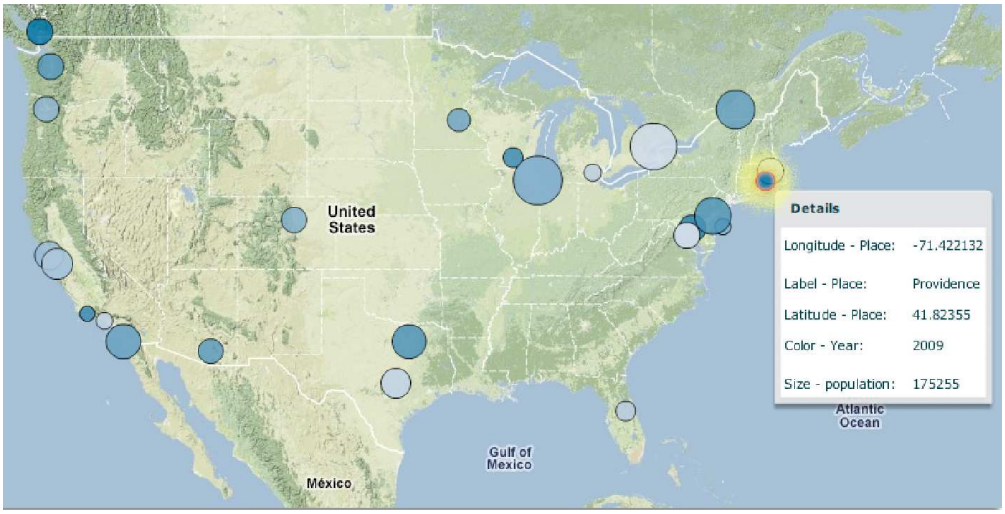

Figure 1.14: A map visualisation of data covering location of SIGMOD host cities in North America and their populations (from [19]).

User interfaces over Semantic Web data can be organised along a continuum from systems that are coded against a pre-defined ontology (e.g. SEAL [52]) to generic user interfaces which assume graph-structured data as underlying data model, with minimal knowledge about the kind of data on which they operate (e.g. Tabulator [7]). Since the Semantic Web data may include ontologies which specify the meaning of data, and the data model is universal [23], there is the possibility for creating a single type of client which aims to provide seamless access to arbitrary content. For dealing with content heterogeneity, general-purpose user interfaces have to offer generic interaction and visualisation functionality, while being extensible and offer specialised capabilities for data from domains which were not anticipated during design-time of the interface [64].

Systems may use a declarative language to specify how RDF should be rendered, similar to using XML style sheets to render XML documents. Fresnel [63] allows for the specifation of so-called lenses which describe how a given piece of RDF should be rendered. Fresnel lenses are used to select and order parts of an RDF graph, and Frensel formats add content formatting and hooks to CSS styling instructions. RDF graphs often encode complex data which require dedicated visualisation widgets for interaction and presentation. An example

of the display of complex data is the SIMILE Timeline widget [45] which can be used to visualise temporal information encoded in an RDF graph.

A major task in building domain-independent user interfaces over Web data is to decide on how to order query results. As in Web search, ranking plays an important role here and offers the software to decide on orderings in lieu of a fixed schema which is typically used to decide on ordering data items. These systems typically employ ranking to prioritise and order items for display.

Finally, Semantic Web user interfaces should allow for the creation and editing of content. The Semantic MediaWiki [51] software extends the wiki system powering Wikipedia with a syntax for adding RDF to wiki pages. Tabulator Redux [8] adds editing capabilities to the generic data browser.

The Semantic Web brings a new set of challenges to user interface design. Easy-to-use interfaces are crucial to achieving the scenario outlined in the beginning of the chapter and let users get answers to question such as "Which type of music is played in UK radio stations?" and "Which radio station is playing titles by Swedish composers?".

1.5 Future Issues

This chapter presented an overview of the architecture of the Semantic Web, starting from the basic principles and requirements, further describing its basic building components with their functionality, ending with approaches to senantic user interfaces. The Semantic Web extends the existing Web and makes online content easier to process by computer programs. Information is represented using explicit statements (facts), with specific semantics that links different pieces of information together. The Semantic Web brings additional meaning to the information on the Web and allows for complex applications operating over collaboratively-edited data.

There are multiple layers and services defined within the architecture of the Semantic Web. The lower layers specify transport and serialisation principles using XML and RDF. On top of them, languages such as RDFS and OWL allow expressing additional semantics and relationships within the data. They introduce schema, distinction between objects and concepts, restrictions, entailment, and reasoning rules. Using these ontology languages, it is possible not only to state simple facts, but also check correctness and consistency of the defined ontology or infer additional facts.

In parallel, the Semantic Web model defines data access principles based on the notion of Linked Data, in addition to a query language, SPARQL, which allows for accessing and

querying knowledge bases in a unified manner, abstracting from the underlying complexity or reasoning mechanisms. W3C recommendations represent a stable technical foundation for data publishing. Although their adoption rate is growing, a direct and comprehensive reuse of published data remains an open issue [43]. Other currently open research questions relate to the use of cryptographic methods for authentication and encryption and the tracking of provenance of data.

Higher layers and services provide extended semantics. Several provenance models are used to provide origin and history of the data, while trust defines trustworthiness of simple, compound and derived information. Rules can be used to model additional dependencies and define inference beyond the axioms defined in ontology languages. Such layers extend the standard Semantic Web and make it more attractive for numerous applications.

Finally, information from the Semantic Web has to be exchanged in a secure manner and presented in a human understandable form to the user. Mechanisms such as HTTPS and FOAL+SSL ensure the security of data transportation and an appropriate access mechanism. A variety of applications provide specialised graphical interfaces for searching and navigating the Semantic Web data in a user-friendly way.

Layers and services overviewed in this chapter characterise different elements of the Semantic Web and define a composite system that covers multiple aspects: from data representation via integration and inference to visualisation. The components introduced in this chapter are explained in more detail in further chapters of this book.

1.6 Related Resources

1. T. Berners-Lee. "Semantic Web." Presentation at XML 2000 Conference, Washington, DC, 2000. `http://www.w3.org/2000/Talks/1206-xml2k-tbl/`

   This presentation gives an overview of the Semantic Web, its philosphy, planned architecture and the basic building blocks. It introduces the layers of the semantic web (semantic web layer cake), highlights the differences bewteen web and semantic web, and explains core architecture layers. Additionally it shows directions for practical deployment of the Semantic Web.

2. J. Handler. "My Take on the Semantic Web Layercake." Presentation at Dagstuhl Semantic Web conference, Germany 2009.
   `http://www.cs.rpi.edu/~hendler/presentations/LayercakeDagstuhl-share.pdf`

   Funny dinner-time presentation of the semantic layer care in rhymes from the Dagstuhl conference. Jim Handler presents the semantic layer cake, how and from where it evolved, and provides some details on different layers - all written as funny, rhymed story. It provides an overview of the core semantic layers placing them in the contexts of previously defined and currently existing web layers or stacks.

3. P. Hitzler, M. Krötzsch, S. Rudolph. "Foundations of Semantic Web Technologies." Chapman & Hall/CRC Textbooks in Computing, 2009.

   Authors present comprehensive description of foundations for semantic web. Major foucus is on ontology languages and representation, formal semantics, logic, rules and reasoning. In addition to detailed chapters on foundation issues, authors present highlights of ontology engineering and example applications. Book illustrated with examples provides strong basics for understanding each of the presented builing blocks of semantic web.

4. S. Staab and R. Studer, editors. "Handbook on Ontologies." Springer, 2009.

   Handbook on Ontologies is a collection of papers by several authors that covers different aspects of ontology engineering. The Handbook begins with the ontology basics like formal languages, logics and reasoning, then presents methodology and different approaches for ontology engineering. Further it concentrates on the use of wide range of existing ontologies, from the top-level and domain ontologies to task or application specific. Finally, it describes infrastructure that enables efficient use of ontologies, reasoning, their storage and retrieval, and present multiple applications and methods for use of ontologies in different tasks and applications.

5. D. Artz and Y. Gil. "A Survey of Trust in Computer Science and the Semantic Web." Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 2007.

   This survey presents different approaches to modelling trust and reasoning about trust. It identifies four major areas of research in trust: policy-based, reputation-based, general models and trust in information resources. Each of the identified areas is described in more details, to give the reader an overview of its most important methods and approaches. Authors provide an extensive list of thematically organized references to publications on trust over the last decade.

References

[1] B. Adida and M. Birbeck. Rdfa primer. W3c working group note, W3C, October 2008.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *Proceedings of 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference (ISWC+ASWC 2007)*, pages 722–735, November 2008.

[3] S. Auer, S. Dietzold, J. Lehmann, S. Hellmann, and D. Aumueller. Triplify: light-weight linked data publication from relational databases. In J. Quemada, G. León, Y. S. Maarek, and W. Nejdl, editors, *WWW*, pages 621–630. ACM, 2009.

[4] D. Beckett. The design and implementation of the redland rdf application framework. *Computer Networks*, 39(5):577 – 588, 2002.

[5] T. Berners-Lee. Universal resource identifiers in WWW: a unifying syntax for the expression of names and addresses of objects on the network as used in the World-Wide web. RFC 1630, Internet Engineering Task Force, June 1994.

[6] T. Berners-Lee. Linked data - design issues, 2006. http://www.w3.org/DesignIssues/LinkedData.

[7] T. Berners-lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *In Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006.

[8] T. Berners-Lee, J. Hollenbach, K. Lu, J. Presbrey, E. Prud'ommeaux, and M. Schraefel. Tabulator redux: Browsing and writing linked data. In *Linked Data on the Web (LDOW2008)*, 2008.

[9] BigData. Bigdata rdf database. http://www.bigdata.com/.

[10] C. Bizer and R. Cyganiak. D2rq - lessons learned. In *W3C Workshop on RDF Access to Relational Databases*, 2007. http://www.w3.org/2007/03/RdfRDB/papers/d2rq-positionpaper/.

[11] C. Bizer and T. Gau. Disco - hyperdata browser. http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/.

[12] H. Boley, G. Hallmark, M. Kifer, A. Paschke, A. Polleres, and D. Reynolds. Rif core dialect. W3c candidate recommendation, W3C, October 2009.

[13] P. Bouquet, H. Stoermer, M. Mancioppi, and D. Giacomuzzi. Okkam: Towards a solution to the "identity crisis" on the semantic web. In *SWAP 2006 - Semantic Web Applications and Perspectives, Proceedings of the 3rd Italian Semantic Web Workshop, Scuola Normale Superiore, Pisa, Italy*, 2006.

[14] E. A. Brewer. Combining Systems and Databases: A Search Engine Retrospective . *Readings in Database Systems, 4th. Edition*, 1998.

[15] D. Brickley and R. Guha. Rdf vocabulary description language 1.0: Rdf schema. W3c recommendation, W3C, February 2004.

[16] J. Broekstra, A. Kampman, and F. V. Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF schema. In *International Semantic Web Conference*, pages 54–68. Springer, 2002.

[17] J. J. Carroll. Signing rdf graphs. In *Second International Semantic Web Conference*, pages 369–384, 2003.

[18] S. Castano, A. Ferrara, S. Montanelli, and G. Varese. Matching semantic web resources. *Database and Expert Systems Applications, International Workshop on*, 0:84–88, 2009.

[19] B. Chan, J. Talbot, L. Wu, N. Sakunkoo, M. Cammarano, and P. Hanrahan. Vispedia: on-demand data integration for interactive visualization and exploration. In *SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data*, pages 1139–1142. ACM, 2009.

[20] G. Cheng and Y. Qu. Searching linked objects with falcons: Approach, implementation and evaluation. *JSWIS*, 5(3):49–70, 2009.

[21] R. Cyganiak. About the linking open data dataset cloud. http://richard.cyganiak.de/2007/10/lod/.

[22] M. d'Aquin, C. Baldassarre, L. Gridinoc, S. Angeletou, M. Sabou, and E. Motta. Characterizing knowledge on the semantic web with watson. In *EON'07*, pages 1–10, 2007.

[23] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. C. A. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The semantic web: The roles of xml and rdf. *IEEE Internet Computing*, 4(5):63–74, 2000.

[24] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. C. Doshi, , and J. Sachs. Swoogle: A Search and Metadata Engine for the Semantic Web. In *Proceedings of the 13th ACM Conference on Information and Knowledge Management*, Nov. 2004.

[25] R. Q. Dividino, S. Schenk, S. Sizov, and S. Staab. Provenance, trust, explanations - and all that other meta knowledge. *KI*, 23(2):24–30, 2009.

[26] R. Q. Dividino, S. Sizov, S. Staab, and B. Schueler. Querying for provenance, trust, uncertainty and other meta knowledge in rdf. *J. Web Sem.*, 7(3):204–219, 2009.

[27] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to match ontologies on the semantic web. *The VLDB Journal*, 12(4):303–319, 2003.

[28] M. Duerst and M. Suignard. Internationalized resource identifiers (IRIs). RFC 3987, Internet Engineering Task Force, Jan. 2005.

[29] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg, 2007.

[30] R. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[31] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force, June 1999.

[32] G. Flouris, I. Fundulaki, P. Pediaditis, Y. Theoharis, and V. Christophides. Coloring rdf triples to capture provenance. In A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, editors, *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 196–212. Springer, 2009.

[33] J. Golbeck. Computing and applying trust in web-based social networks. In *Ph.D. thesis*, 2005.

[34] W. O. W. Group. Owl 2 web ontology language document overview. W3c recommendation, W3C, October 2009.

[35] H. Halpin and V. Presutti. An ontology of resources for linked data. In *Linked Data on the Web (LDOW2009)*, 2009.

[36] A. Harth. Visinav: Visual web data search and navigation. In *Database and Expert Systems Applications, 20th International Conference, DEXA 2009, Linz, Austria, August 31 - September 4, 2009. Proceedings*, pages 214–228, 2009.

[37] A. Harth, K. Hose, M. Karnstedt, A. Polleres, K.-U. Sattler, and J. Umbrich. Data summaries for on-demand queries over linked data. In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 411–420, New York, NY, USA, 2010. ACM.

[38] A. Harth, J. Umbrich, and S. Decker. Multicrawler: A pipelined architecture for crawling and indexing semantic web data. In *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 258–271. Springer, 2006.

[39] S. Hawke. Rule interchange format (rif) hcls 2009 - use case: Vocabulary mapping, 2009. http://www.w3.org/2009/Talks/0903-rif/Overview.html.

[40] D. Heimbigner and D. McLeod. A federated architecture for information management. *ACM Trans. Inf. Syst.*, 3(3):253–278, 1985.

[41] J. Heinsohn, D. Kudenko, B. Nebel, and H.-J. Profitlich. An empirical analysis of terminological representation systems. *Artif. Intell.*, 68(2):367–397, 1994.

[42] B. Heitmann, S. Kinsella, C. Hayes, and S. Decker. Implementing semantic web applications: reference architecture and challenges. In *Proceedings of 5th International Workshop on Semantic Web Enabled Software Engineering, co-located with ISWC 2009*, 2010.

[43] A. Hogan, A. Harth, A. Passant, S. Decker, and A. Polleres. Weaving the pedantic web. In *Proceedings of the Linked Data on the Web WWW2010 Workshop (LDOW 2010)*, 2010.

[44] A. Hogan, A. Harth, J. Umbrich, and S. Decker. Towards a scalable search and query engine for the web. In *WWW'07*, pages 1301–1302, 2007.

[45] D. F. Huynh, D. R. Karger, and R. C. Miller. Exhibit: lightweight structured data publishing. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 737–746. ACM, 2007.

[46] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *International World Wide Web Conference (WWW2003)*, 2003.

[47] E. Kaufmann and A. Bernstein. How Useful are Natural Language Interfaces to the Semantic Web for Casual End-users? In *6th International Semantic Web Conference (ISWC 2007)*, pages 281–294, March 2007.

[48] K. Kjernsmo and A. Passant. Sparql new features and rationale - w3c working draft 2 july 2009, 2009. http://www.w3.org/TR/sparql-features/.

[49] G. Kobilarov and I. Dickinson. Humboldt: Exploring linked data. In *Linked Data on the Web (LDOW2008)*, 2008.

[50] D. Kossmann. The state of the art in distributed query processing. *ACM Computing Surveys*, 32(4):422–469, Dec. 2000.

[51] M. Krötzsch, D. Vrandecic, and M. Völkel. Semantic mediawiki. In *International Semantic Web Conference*, pages 935–942, 2006.

[52] A. Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure. Semantic portal - the seal approach. In *Spinning the Semantic Web*, pages 317–359, 2003.

[53] B. McBride. Jena: Implementing the rdf model and syntax specification. In *Proceedings of the Second International Workshop on the Semantic Web*, 2001.

[54] L. Moreau, J. Freire, J. Futrelle, R. E. McGrath, J. Myers, and P. Paulson. The open provenance model: An overview. In J. Freire, D. Koop, and L. Moreau, editors, *IPAW*, volume 5272 of *Lecture Notes in Computer Science*, pages 323–326. Springer, 2008.

[55] A. Motro. Superviews: Virtual integration of multiple databases. *Software Engineering, IEEE Transactions on*, SE-13(7):785–798, July 1987.

[56] T. Neumann and G. Weikum. The rdf-3x engine for scalable management of rdf data. *VLDB J.*, 19(1):91–113, 2010.

[57] OAEI. Ontology alignment evaluation initiative. http://oaei.ontologymatching.org/.

[58] OpenID. Openid foundation website, 2010. http://openid.net/.

[59] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello. Sindice.com: A document-oriented lookup index for open linked data. *JMSO*, 3(1), 2008.

[60] E. Oren, R. Delbru, and S. Decker. Extending faceted navigation for rdf data. In *Proceedings of the 5th International Semantic Web Conference*, pages 559–572, 2006.

[61] OWLIM. Owlim semantic repository. http://www.ontotext.com/owlim.

[62] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *ICDE '95: Proceedings of the Eleventh International Conference on Data Engineering*, pages 251–260, Washington, DC, USA, 1995. IEEE Computer Society.

[63] E. Pietriga, C. Bizer, D. R. Karger, and R. Lee. Fresnel: A browser-independent presentation vocabulary for rdf. In *International Semantic Web Conference*, pages 158–171, 2006.

[64] D. A. Quan and R. Karger. How to make a semantic web browser. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 255–265, New York, NY, USA, 2004. ACM.

[65] B. Quilitz and U. Leser. Querying distributed rdf data sources with sparql. In *5th European Semantic Web Conference*, pages 524–538, 2008.

[66] Y. Raimond, C. Sutton, and M. Sandler. Interlinking music-related data on the web. *IEEE MultiMedia*, 16:52–63, 2009.

[67] E. Rescorla. HTTP over TLS. RFC 2818, Internet Engineering Task Force, May 2000.

[68] K. Sagonas, T. Swift, and D. S. Warren. Xsb as an efficient deductive database engine. *SIGMOD Rec.*, 23(2):442–453, 1994.

[69] S. Schenk, C. Saathoff, S. Staab, and A. Scherp. Semaplorer - interactive semantic exploration of data and media based on a federated cloud infrastructure. *J. Web Sem.*, 7(4):298–304, 2009.

[70] S. Schenk and S. Staab. Networked graphs: a declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the web. In *World Wide Web*, pages 585–594. ACM, Apr. 21-25, 2008.

[71] M. Schraefel, J. Golbeck, D. Degler, A. Bernstein, and L. Rutledge. Semantic web user interactions: exploring hci challenges. In *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*, pages 3929–3932, New York, NY, USA, 2008. ACM.

[72] A. Seaborne, G. Manjunath, C. Bizer, J. Breslin, S. Das, I. Davis, S. Harris, K. Idehen, O. Corby, K. Kjernsmo, and B. Nowack. Sparql update - a language for updating rdf graphs (w3c member submission 15 july 2008), 2008. http://www.w3.org/Submission/SPARQL-Update/.

[73] E. Selberg and O. Etzioni. Multi-service search and comparison using the metacrawler. In *Proceedings of the 4th International World Wide Web Conference*, pages 195–208, 1995.

[74] SPARQL. Sparql query language for rdf - w3c recommendation 15 january 2008, 2008. http://www.w3.org/TR/rdf-sparql-query/.

[75] M. Stefaner, S. Ferr, S. Perugini, J. Koren, and Y. Zhang. *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience*, volume 25 of *The Information Retrieval Series*, chapter 4 - User Interface Design, pages 75–112. Springer, 2009.

[76] H. Story, B. Harbulot, I. Jacobi, and M. Jones. FOAF+TLS: RESTful Authentication for the Social Web. In *First Workshop on Trust and Privacy on the Social and Semantic Web (SPOT2009)*, Jun 2009.

[77] H. Stuckenschmidt, R. Vdovjak, J. Broekstra, and G.-J. Houben. Towards distributed processing of RDF path queries. *Int. J. Web Eng. Technol.*, 2(2/3):207–230, 2005.

[78] Virtuoso. Virtuoso universal server. http://virtuoso.openlinksw.com/.

[79] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the web of data. In *8th International Semantic Web Conference (ISWC2009)*, October 2009.

[80] G. Wiederhold. Mediators in the architecture of future information systems. *Computer*, 25(3):38–49, 1992.

[81] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, USA, 2003. ACM.