4 Semantic Technologies

As discussed in section 2.1 of this thesis we regard semantic technologies “as technical approaches that facilitate or make use of the interpretation of meaning by machines”. Ontologies are one of the core elements of semantic solutions. In the following, we review the definition of ontologies and briefly describe their general characteristics. Moreover, we discuss important concepts for ontology and knowledge representation within the Semantic Web. After that, we explain ways to process knowledge representations, such as reasoning, inferencing, and querying. Due to the focus of this thesis, we finally describe how relational databases and ontologies are related.

4.1 Characteristics of an Ontology

In section 2.1, we derived the following definition for ontologies: Ontologies are “a formal and sharable means to explicitly model some real-world phenomenon for machine-readable knowledge representation”. According to this definition, ontologies have at least five important characteristics, namely “formality, explicitness, being shared, conceptuality and domain-specificity” (Grimm et al., 2007, p. 69f.). In the following, we will explain the term “ontology” along these five characteristics.

Formality: With ontologies, real-world phenomena and their relationships among each other can be described in a machine-readable way by using formal elements, i.e. concepts, relationships, instances, and axioms (cf. Grimm et al., 2007, p. 88). Ontologies are therefore used to structure and store knowledge about a domain of interest. The degree of formality of ontologies and their expressiveness to represent real-world elements varies from natural language descriptions to highly formal axioms (cf. Smith & Welty, 2001, p. 6f.; Uschold & Gruninger, 1996, p. 98). In fact, there are several different knowledge representation languages that offer modeling constructs to represent different levels of formality. The degree of formality thereby influences the ability of machine-interpretation of the represented knowledge. With increasing formality, the machine interpretation capabilities rise, but also the complexity of ontology development and maintenance increases.

Explicitness: While much knowledge usually relies in people’s minds, the development of a materialized ontology documents expert knowledge in an explicit

C. Fürber, *Data Quality Management with Semantic Technologies*,
DOI 10.1007/978-3-658-12225-6_4, © Springer Fachmedien Wiesbaden 2016

way. Moreover, the design of formal ontologies for machine interpretation promotes the rigorous explicit representation of knowledge within the ontology and the automated identification of misconceptions, i.e. inconsistencies within the ontology / understanding of a domain (cf. Grimm et al., 2007, p. 70; Hepp, 2008b, p. 16).

Being shared: Ontologies are usually developed for a certain community, e.g. to capture the knowledge of domain experts. For its successful adaptation it is, therefore, necessary to achieve agreement about the ontology among large parts of the community (cf. Grimm et al., 2007, p. 70). Once an agreement can be established, the chance for widespread adoption of the ontology as a standardized means to represent knowledge rises. Thereby ontologies may help to improve communication, enable reuse of shared knowledge, and facilitate interoperability while keeping schematic heterogeneity at a minimal level (cf. Gasevic et al., 2006, p. 48).

Domain specificity: Due to the complexity of representing concise knowledge and achieving agreement, ontologies are usually limited to a certain domain (cf. Grimm et al., 2007, p. 70). Despite domain specificity, ontologies can be combined with other ontologies to represent knowledge of multiple domains.

Conceptuality: The represented knowledge within ontologies is organized into concepts and relationships. The concepts and relationships can also be represented in hierarchies so that different levels of abstraction may be represented while being connected to each other. Instead of explaining individual phenomena, ontologies provide a framework for as many tasks as necessary within the domain of interest (Grimm et al., 2007, p. 70).

In summary, the use of ontologies for the representation of domain knowledge promises the following benefits (cf. Hepp, 2008b):

- Reduction of ambiguity through the formal and explicit representation of knowledge,
- conservation of implicit knowledge through explicit representation,
- knowledge sharing and reuse through the provision of a common vocabulary / ontology,
- reduction of manual work through the reuse of shared knowledge,
- reduction of manual work through a formal, machine-interpretable knowledge representation,

- automated inference of implicit facts through the formal representation of knowledge,
- automated identification of misconceptions through the formal, explicit representation of knowledge, and
- improved interoperability through the use of a common vocabulary / ontology.

Collections of actual instances that use the elements of ontologies to represent knowledge are known as knowledge bases and should not be confused with ontologies that provide the vocabulary to express knowledge (cf. Hepp, 2008b, p. 6). In the following, we use the term “ontology” to name the schema of knowledge and the term “knowledge base” to refer to an ontology-based representation of knowledge instances.

4.2 Knowledge Representation in the Semantic Web

Ontologies and knowledge bases in Semantic Web architectures are typically represented by using and combining elements of the “Resource Description Framework” (RDF)[10], “RDF Vocabulary Description Language” (which is also known as “RDF Schema” (RDFS)[11]), and the “Web Ontology Language” (OWL)[12]. The following subsections will give a brief overview about the most important language constructs of the Semantic Web, namely resources and Uniform Resource Identifiers (URI), the core RDF Syntax, and important vocabulary elements of RDF, RDFS, and OWL related to the topics of this thesis.

4.2.1 Resources and Uniform Resource Identifiers (URIs)

Semantic Web languages describe resources and relationships among resources. The term “resource” has thereby a very generic meaning which is not constrained to any subset of concepts. A resource can be a Web site, a product, a document, a service, a plan, a person, or anything else (cf. Berners-Lee et al., 2005). Resources are

[10] Resource Description Framework (RDF), http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/
[11] RDF Schema (RDFS), http://www.w3.org/TR/2004/REC-rdf-schema-20040210/
[12] Web Ontology Language, http://www.w3.org/TR/2004/REC-owl-guide-20040210/, recently updated to OWL 2, http://www.w3.org/TR/2009/REC-owl2-overview-20091027/

identified by Uniform Resource Identifiers (URIs) (Sauermann & Cyganiak, 2008). Web addresses like "`http://www.google.com`" are a special kind of URI, namely a Uniform Resource Locator (URL) which not only identifies a resource, but also locates it (Berners-Lee et al., 2005). A major advantage of URIs on the World Wide Web (WWW) is their global uniqueness. Therefore, URIs facilitate the unambiguous identification of resources. However, there are several limitations on the WWW that may disturb the unambiguous identification of a resource via its URI. The resource which is identified by the URI may over time disappear or its meaning may change. Moreover, it is possible that the URL of one resource is redirected to the URL of another resource. In order to avoid changes, URIs should be designed carefully so that they can be held stable and lasting (cf. Berners-Lee, 1998a).

4.2.2 Core RDF Syntax: Triples, Literal Triples, and RDF Links

The core structure of RDF are so called triples. Triples allow the definition of statements in a subject, predicate, object format as illustrated in figure 23 (cf. Klyne & Carroll, 2004). With the triple structure, it is possible to draw relationships (predicates) between two entities or between an entity and the state of a property (subject, object). Therefore, the predicate position of a triple is always reserved for a property "that denotes a relationship" (Klyne & Carroll, 2004). Properties are always identified via URIs. Combinations of multiple triples form a graph (cf. Grimm et al., 2007, p. 84).

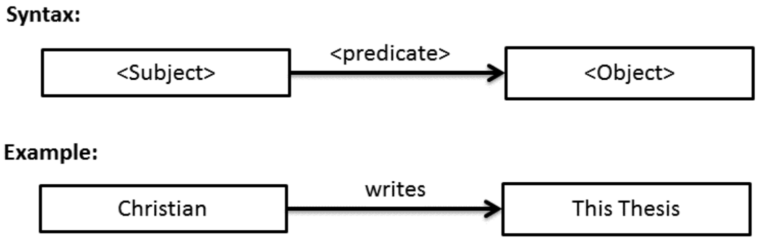

Figure 23: Syntax of RDF triples (cf. Klyne & Carroll, 2004)

We can differentiate between two different types of RDF triples, namely "Literal triples" and "RDF links" (Heath & Bizer, 2011). RDF links are triples with URIs in subject and object position (Heath & Bizer, 2011). Hence, the predicate of RDF links connects two resources with each other (Heath & Bizer, 2011). RDF links can, therefore, be used to describe relationships between two resources (cf. Heath & Bizer, 2011). RDF links have so called object properties in predicate position when using OWL (cf. Hitzler et al., 2009). Literal triples have data values in the object position which are known as literals (cf. Heath & Bizer, 2011). They may be restricted to a certain datatype and contain a language tag indicating the language in which the literal is represented (cf. Heath & Bizer, 2011). Literals with datatype indication are called typed literals, literals without datatype indication are called plain literals (cf. Heath & Bizer, 2011). Thus, Literal triples can be used to assign values to properties of a resource. In other words, Literal triples describe the states of properties of an entity (cf. Heath & Bizer, 2011). For example the triple `http://example.org/JonMyer foo:hasBirthday "1970-01-01"` is a Literal triple because the object position of the triple contains the literal `"1970-01-01"`. Literal triples can be modeled using OWL datatype properties in predicate position. An example for an RDF link triple would be `http://example.org/JonMyer foo:hasMother http://example.org/JanetMyer`, since two resources with URIs are linked to each other.

4.2.3 Constructing an Ontology with RDF, RDFS, and OWL

Main elements of ontologies in Semantic Web architectures are classes and properties. *Properties* are in predicate position of a triple and, therefore, define relationships between resources or describe facts about resources as explained in the previous section. *Classes* are conceptual entities that can be used to classify resources into categories (cf. Manola & Miller, 2004). The resources that belong to a class are called its *instances* (Manola & Miller, 2004). An ontology together with its instances is called a *knowledge base* (cf. Noy & McGuinness, 2001, p. 3). Knowledge bases are represented in so called RDF graphs (cf. Sirin et al., 2007, p. 12). Semantic Web programming languages provide several classes and properties that can be used to model semantic distinctions of user-defined classes and properties in a standardized and machine-interpretable way. In the following, core modeling constructs of RDF, RDFS, and OWL are explained which are important for the understanding of this thesis.

Datatype properties: With OWL, a property can be declared as a datatype property meaning that the property can only have literals in the object position. The range of the property may be restricted to a certain datatype either by using XML Schema datatypes[13] or via self-defined datatypes with OWL 2 (cf. Hitzler et al., 2009).

Language tag assignment: Language tags can be assigned at the end of literals to indicate the language in which the literal is written (cf. Alvestrand, 2001; Beckett, 2004).

Domain of a property: The property `rdfs:domain` is a property of RDF-properties. It can be used to specify classes that hold individuals which can be used as a subject for the described property (cf. Brickley & Guha, 2004). In other words, `rdfs:domain` specifies the class of individuals which may be described by the property. E.g. the domain of the property `foo:hasEAN` is the class `foo:Material`.

Range of a property: The property `rdfs:range` is also a property of RDF-properties. It is used to specify the allowed types used for the values of a property, i.e. which datatype the values must have or to which class the values must belong (cf. Brickley & Guha, 2004). E.g. the property `foo:hasName` has a range of datatype `xsd:string`. It is important to note that the consequences of applying a property to an instance of another type is that an additional class membership is inferred (cf. De Bruijn et al., 2005, p. 5).

Class membership: RDF allows the definition of class memberships of entities (cf. Brickley & Guha, 2004). E.g. the triple `Christian rdf:type PhD-Student` expresses that the individual "`Christian`" belongs to the class of PhD Students.

Class and property hierarchies: RDFS allows the expression of hierarchic relationships between classes and properties (cf. Brickley & Guha, 2004). For example, we can define that the class `PhD-Students` is a sub-class of the class `Person` or that the property `lastName` is a sub-property of the property `name`.

Equivalence between classes / properties: With the OWL properties `owl:equivalentClass` and `owl:equivalentProperty` we can express that classes or properties are equivalent in terms of that equivalent properties share the same values and equivalent classes share the same individuals (cf. Bechhofer et al., 2004; Hitzler et al., 2012).

[13] XML Schema datatypes, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

Identity between individuals: With the OWL property `owl:sameAs` we can express semantic equality between individuals, i.e. the resources connected with `owl:sameAs` represent the same real-world object (cf. Bechhofer et al., 2004).

Disjointness of classes: The property `owl:disjointWith` facilitates the expression of disjointness between two classes, i.e. that individuals cannot be member of both classes at the same time (cf. Bechhofer et al., 2004).

Transitivity of a property: OWL supports the definition of transitive properties by making the properties instances of the class `owl:TransitiveProperty`. Transitivity in this context means that the property relationship will also apply for the subject of one triple and the object of a second triple if the object of triple one is also the subject of triple two, although they are not directly connected to each other. E.g. if the property `foo:subProductOf` is defined to be a transitive property and the two triples X `foo:subProductOf` Y and Y `foo:subProductOf` Z exist, then we can derive that X `foo:subProductOf` Z (cf. Bechhofer et al., 2004).

Symmetry of a property: A property is symmetric if the subject and the object of the triple, in which the property is used, can be substituted without making an incorrect statement. Symmetric properties can be defined via OWL by making the property an instance of the class `owl:SymmetricProperty` (cf. Bechhofer et al., 2004). E.g. the property `foo:marriedTo` is symmetric because a marriage is always mutual.

Inverse properties: With OWL, we can define that one property is an inverse of another property (cf. McGuinness & van Harmelen, 2004). E.g. the property `foo:writtenBy` is an inverse of the property `foo:authorOf`.

Functional properties: Functional properties are properties "that can have only one (unique) value y for each instance x" (Bechhofer et al., 2004). A property is defined as functional by making it an instance of the class `owl:FunctionalProperty`. Functional properties are a way to express global cardinality restrictions (cf. Bechhofer et al., 2004). E.g. a car can only have one active license plate number.

Inverse functional properties: Inverse functional properties uniquely identify the subject in a triple. In other words, a value of an inverse functional property must only belong to the same individual. A property is defined as inverse functional by making it an instance of the class `owl:InverseFunctionalProperty`. Inverse functional properties are a way to express global cardinality restrictions (cf. Bechhofer et al., 2004). E.g. a certain social security number can only belong to one person.

Cardinality restrictions: OWL provides the properties `owl:maxCardinality`, `owl:minCardinality`, and `owl:Cardinality` to define cardinality restrictions on ranges of properties. The OWL cardinality properties hold values of datatype `xsd:nonNegativeInteger`. A restriction with `owl:maxCardinality` "describes a class of all individuals that have *at most* N semantically distinct values (individuals or data values) for the property concerned, where N is the value of the cardinality constraint" (Bechhofer et al., 2004). Analogous to the `owl:maxCardinality`, `owl:minCardinality` describes a class of individuals that must at least have N semantically distinct values, and `owl:Cardinality` describes a class that has exactly N semantically distinct values (cf. Bechhofer et al., 2004). Since the cardinality only applies to semantically distinct values and the same individuals may be represented by syntactically distinct values, it is possible that, although `owl:maxCardinality` has value "1", an instance has two values for a property that represent the same individual. If both values represent the same individual, then the restriction will still be followed.

The Semantic Web programming languages RDF, RDFS, OWL, and OWL 2 allow many more formal semantic expressions which are not explained in this thesis due to their lack of relevance for the focus of this work.

4.2.4 Language Profiles of OWL and OWL 2

The Web Ontology Language OWL has three common language profiles, namely OWL Lite, OWL Description Logic (DL), and OWL Full (Bechhofer et al., 2004). A language profile thereby provides a subset of language constructs of OWL and may constrain their usage (Bechhofer et al., 2004). In OWL Full, all elements of the language can be used with no restrictions as long as valid RDF documents are produced (Bechhofer et al., 2004). OWL DL and OWL Lite are subsets of OWL (Bechhofer et al., 2004). One of the major distinctions between OWL Full and OWL DL is the meta-modeling capability of OWL Full. In OWL Full, classes and properties can also be used as an individual. This is not allowed in OWL DL to provide a language profile for decidable reasoning, i.e. automated inferencing of implicit knowledge within finite time

(Bechhofer et al., 2004). OWL Lite is the simplest of all OWL profiles and provides a minimal subset of OWL with the most important ontological constructs to provide an easy way to engineer an ontology (cf. Hitzler, 2008, p. 151ff.). At present, most ontologies are coded in OWL DL.

OWL 2 introduces three new language profiles, namely OWL 2 EL, OWL 2 RL, and OWL 2 QL (W3C-OWL-Working-Group, 2012). The different language profiles of OWL 2 have been composed for specific cases. For example, OWL 2 EL is optimized for very large ontologies with many classes and properties (W3C-OWL-Working-Group, 2012). OWL 2 QL was designed to provide “sound and complete query answering” (Motik et al., 2009) at a reasonable time. And OWL 2 RL is optimized for reasoning (W3C-OWL-Working-Group, 2012). For a detailed overview about the different language profiles for OWL 2, please see (Motik et al., 2009).

Thus, when designing new ontologies, it is important to consider the required level of expressivity and the scenarios in which the ontology shall be used, in order to identify a proper language profile. In the following, the acronym OWL is used to refer to both, OWL and OWL 2.

4.3 SPARQL Query Language for RDF

Query languages have been used for several decades, e.g. the Structured Query Language (SQL) to update and retrieve data from relational databases (Oracle, 2013). The Semantic Web provides its own query language, called the SPARQL query language for RDF (SPARQL) (Harris & Seaborne, 2010). SPARQL can be used to store, update, retrieve, and delete data in knowledge bases and provides several mechanisms, such as aggregations, subqueries, or filters, that are very similar to features of SQL (cf. Harris & Seaborne, 2010). Other than with SQL, SPARQL can be combined with reasoners to also retrieve information that is not explicitly represented[14]. E.g. a SPARQL query asking for instances of the class `Person` could also retrieve instances of subclasses of the class `Person`, if subclass reasoning was enabled. A lot

[14] There has been work on deductive databases that combine logic programming and database management systems. However, to the best of the author’s knowledge they are not widely used in business information systems.

of triplestores and Semantic Web tools, such as Virtuoso[15] or TopBraid Composer[16], provide so called SPARQL endpoints (Feigenbaum et al., 2013) with query interfaces to access the knowledge base or RDF files via SPARQL queries. Moreover, a lot of the available SPARQL query interfaces provide additional, proprietary SPARQL functions (also known as SPARQL extensions), that extend the SPARQL standard functionalities[17] as specified by the World Wide Web Consortium (W3C). At time of this thesis, SPARQL 1.1 provides a mostly stable and expressive syntax that is already implemented in many commercial and non-commercial Semantic Web tools.

4.4 Reasoning and Inferencing

Besides the plain retrieval of Semantic Web data via SPARQL queries, it is also possible to employ the expressiveness of ontologies and the represented knowledge via so called reasoners (cf. Hebeler et al., 2009, p. 285). Reasoners are programs that use the represented logic of ontologies and / or user-defined rules (1) to infer implicit knowledge and (2) to check the logical consistency at ontology and instance level (cf. Antoniou & van Harmelen, 2008, pp. 97-103; Fensel & van Harmelen, 2007). According to Hebeler et al. (Hebeler et al., 2009, p. 285), there are two different types of reasoners which can also be combined in a single engine, namely inference reasoners and rule-based reasoners. Inference reasoners infer implicit knowledge and check logical consistency based on the axioms represented via RDFS and OWL (cf. Hepp, 2008b, p. 15f.). Rule-based reasoners process user-defined rules that are represented additionally to the axioms of an ontology (cf. Hebeler et al., 2009, pp. 231-233). Similar to the axioms of RDFS and OWL, user-defined rules can also be used to infer new knowledge or check consistency, but provide more flexibility for the definition of axioms (cf. O'Connor et al., 2005, p. 975). Depending on the processing capabilities of the reasoner, rules can be represented in different languages, such as the Semantic Web Rule Language (SWRL)[18] or via the vocabulary of the SPARQL Inferencing

[15] http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSIntro (Last accessed on April 10th 2012)
[16] http://www.topquadrant.com/products/TB_Composer.html (Last accessed on April 10th 2012)
[17] http://www.w3.org/TR/2010/WD-sparql11-query-20100126/ (Last accessed on April 10th 2012)
[18] http://www.w3.org/Submission/SWRL/ (Last accessed on April 11th 2012)

framework (SPIN)[19]. A popular open source reasoner that combines both, inference and rule-based reasoning, is Pellet[20].

The inferable knowledge via inference reasoning depends on the formal elements that are used within the ontology. In the following, we provide some examples of potential inferences that can be made when reasoning knowledge provided by an OWL DL ontology (cf. Hitzler, 2008, p. 176f.).

Class equivalency: Based on equivalency relationships, it can be inferred which classes belong to a specific domain concept. E.g. by specifying that class `Person` and class `HumanBeing` are equivalent, a reasoner can process this information to automatically infer the members of both classes.

Subclass relationships: Based on the definition of subclass relationships, a reasoner can derive all members of a superclass including members that are not explicit members of the superclass. E.g. a reasoner could infer that the individual `Christian` not only belongs to the class `PhD-Student`, but also belongs to the class `Person`, since the class `PhD-Student` is a subclass of the class `Person`. In the following, we will use the term "subclass reasoning" to refer to this kind of inferencing.

Disjunctive classes: With OWL, classes can be defined as disjunctive, i.e. that members of class A cannot also be members of class B at the same time, if class A and class B are disjunctive. Based on this knowledge representation, reasoners can identify individuals that are members of disjunctive classes and, thus, identify and report inconsistent class memberships.

Additional inferencing capabilities for knowledge represented in ontologies based on RDFS and OWL can be found in (Hitzler, 2008). As mentioned in the previous section, the more formal elements and axioms are used within an ontology, the more resources are needed for the reasoning based on the ontology (cf. Antoniou & van Harmelen, 2008, p. 158; Fensel & van Harmelen, 2007; Gómez-Pérez et al., 2004, p. 204). Hence, for efficient reasoning it is important to pay attention to the design of an ontology, especially regarding the chosen language profile.

[19] http://spinrdf.org/ (Last accessed on April 11th 2012)
[20] http://clarkparsia.com/pellet/features (Last accessed on April 11th 2012)

4.5 Ontologies and Relational Databases

Ontologies and relational databases (RDB) are related to each other in at least two aspects. First, a lot of data that is currently available on the Semantic Web has been published via mapping technologies between RDB and ontologies (cf. Bizer, Heath, et al., 2009). Secondly, some triplestores use the efficient and mature technologies of RDB management systems (RDBMS) to store RDF triples (Heymans et al., 2008, p. 92). In this section, we examine how data from relational databases can be linked to conceptual elements from ontologies and exposed as RDF data. Relational data can be lifted into the Semantic Web space, namely (1) virtually without a persistent representation of the data in RDF or (2) persistently with a persistent conversion of the data into RDF (Sahoo et al., 2009). In both cases, the elements of the relational schema have to be mapped to the target ontology. Table 6 shows how the different elements of an RDB schema can be mapped to the elements of an ontology based on findings from Astrova (Astrova, 2009).

Table 6: Simplified mapping between RDBs and ontologies (cf. Astrova, 2009)

RDB Element	**Ontology Element**
Table[21] / View	Class
Table with only two foreign key columns	Object property
Column containing datatype values	Datatype property
Column containing foreign keys	Object property
Primary keys	Individuals / URIs
Row	Instance

It must be stressed that there may also be much more individual mappings between elements of an RDB to elements of an ontology. E.g. one might want to populate tuples of a specific table to multiple different classes based on filters on certain column values. However, there are many ways to easily expose relational sources to the Semantic Web spaces, such as D2RQ or Virtuoso RDF-Views (please see (Sahoo et al., 2009)

[21] Tables that only contain two columns with foreign keys are mapped to object properties

for a survey about RDB2RDF mapping technologies). In summary, we can conclude that relational data can be used in Semantic Web architectures via mappings to ontology elements. This facilitates the use of Semantic Web technologies to process data of RDB.