Part II

Representation of Meta-Information

As we have seen in Chapter 3, quality-based information filtering policies rely on different types of meta-information about information itself, the information provider, or the information provision process. Thus, a prerequisite for being able to employ different information filtering policies, is to represent information together with quality-related meta-information using an adequate data model.

This part of the thesis proposes a data model for representing information together with quality-related meta-information. As web-based information systems integrate information from multiple sources, the proposed data model also has to satisfy requirements that arise from integrating information from the Web. According to [Dec02], a data model for representing information from the Web should fulfill the following requirements:

1. *Information on the Web is distributed.* The data model should therefore provide for the simple integration of information from multiple sources. As different information providers might describe different aspects of an object, the model should support the efficient merge of partial descriptions of objects.

2. *Objects of interest should be uniquely identifiable.* Different information sources might provide information about the same objects. Thus, in order to simplify merging information about an object from multiple sources, objects should be uniquely identifiable across information sources.

3. *Information on the Web is heterogeneous and often not strictly structured.* Therefore, the data model must be flexible enough to represent structured as well as less structured information.

4. *Support for semantic interoperability.* Different information providers use different schemata to represent information about a domain. Therefore, the data model should provide for the unique identification of terms and concepts from different schemata. This enables applications to keep track of semantic differences and lays the foundation for mapping information between schemata.

5. *Support for syntactic interoperability.* In order to ease the exchange of information, the data model should be accomplished with standard syntaxes for serializing information.

The proposed data model for representing information together with quality-related meta-information is based on the Resource Description

Framework [KC04], a state-of-the-art data model for web-based information systems. The Resource Description Framework has been standardized by the World Wide Web Consortium (W3C) and is a central base-technology within the Semantic Web architecture stack [KM01].

This part of the thesis is organized as follows:

Chapter 4: The Resource Description Framework. This chapter describes the Resource Description Framework (RDF) and explains how the RDF data model fulfills the requirements arising from integrating information from the Web. Afterwards, the capabilities of the RDF data model to represent information together with quality-related meta-information are examined. The analysis shows that the RDF data model does not provide an efficient mechanism for representing information together with quality-related meta-information.

Chapter 5: The Named Graphs Data Model. This chapter proposes the Named Graphs data model, an extension to the RDF data model which eliminates the shortcomings found in the previous chapter. Afterwards, the TriG and TriX syntaxes for exchanging sets of named graphs are introduced.

Chapter 6: The Semantic Web Publishing Vocabulary. An important type of meta-information is provenance information about the origin of information. This chapter develops the Semantic Web Publishing Vocabulary for representing provenance information and for assuring the origin of information with digital signatures.

Chapter 7: Use Case: Financial Information Integration. This chapter shows how the Named Graphs data model and the Semantic Web Publishing Vocabulary are used to represent information about stocks, companies, analyst reports, and financial news together with quality-related meta-information.

Chapter 4

The Resource Description Framework

The Resource Description Framework (RDF) [HSB06] is a set of standardized technologies designed to represent information about web resources, publish structured information on the Web, and exchange information between web-based information systems. RDF is a major technical component in the vision of extending the current Web to what is called the Semantic Web [BLHL01, SBLH06, AvH04]. The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries [Her06]. The basic idea of the Semantic Web is to make structured data accessible on the Web by using common formats and by referring to terms from shared conceptualizations of an application domain. The meaning of these terms is captured in the form of ontologies [Gru93]. Such a web of semantically interoperable data would allow a person or a machine to start browsing the content of one data source, and then move through a potentially huge set of data sources that provide related information [Her06]. It would also enable sophisticated queries, similar to SQL queries [ISO03b], to be executed against the data sources.

RDF originates from the Platform for Internet Content Selection (PICS) [MKRT96], a standard for associating meta-information with Internet content. RDF has been developed by two consecutive W3C working groups. The members of the working groups came from diverse backgrounds including databases, information integration, knowledge modeling, artificial intelligence, and information retrieval. The standardization effort lead in 2004 to a set of six W3C recommendations which together specify RDF:

1. The Resource Description Framework (RDF): Concepts and Abstract Syntax Recommendation [KC04] defines the RDF data model and de-

scribes the motivations which drove the design of the model.

2. The RDF Vocabulary Description Language 1.0: RDF Schema Recommendation [BG04] specifies a simple schema language for the RDF data model. An RDF schema defines terms for describing an application domain. A schema consists of class and property definitions and contains information about relations between classes and properties.

3. The RDF Semantics Recommendation [Hay04] contains a formal definition of the semantics of RDF and RDF Schema. It also defines a system of inference rules that allows implicit information to be derived from RDF data.

4. The RDF/XML Syntax Specification (Revised) [Bec04b] defines an XML-based serialization format for RDF data. The RDF/XML syntax is used to exchange RDF data between information systems and to embed RDF data into Web documents.

5. The RDF Primer [MM04] provides the reader with a basic overview of the Resource Description Framework and gives several usage examples.

6. The RDF Test Cases [GB04] define a set of test cases for validating the standard conformance of RDF implementations.

This section introduces the RDF data model and the RDF Schema language and gives an overview of RDF serialization syntaxes and query languages. Afterwards, the utility of the RDF data model to represent information together with quality-related meta-information is examined.

4.1 The RDF Data Model

The RDF data model represents information as node-and-arc-labeled directed graphs [KC04]. The data model is designed for the integrated representation of information that originates from multiple sources, is heterogeneously structured, and is represented using different schemata. RDF aims at being employed as a lingua franca, capable of moderating between other data models that are used on the Web [KC04]. There is a growing number of mediator and wrapper frameworks that expose information, represented within another data model, as RDF [BS04, HMC04, Cla99].

4.1.1 Triples and Graphs

Within the RDF data model all objects of interest are called *resources*. Resources have properties. Each property has a property type and a property value. Property values can be atomic, e.g. strings or numbers, or references to other resources, which in turn may have their own properties.

Information about resources is represented in the form of triples. Each triple represents a single property of a resource. Triples can be compared to simple sentences. Each triple consists of a subject, a predicate, and an object. The subject determines the resource which is described by the triple. The predicate determines a property type. The object contains the property value.

Triples can be visualized as node and arc diagrams. In this notation, a triple is represented by a node for the subject, a node for the object, and an arc for the predicate, directed from the subject node to the object node. The triple shown in Figure 4.1 represents the piece of information that Document1325 is authored by Chris Bizer. Document1325 is the subject of this triple. Chris Bizer is the object of this triple. The term `author` is the predicate and determines the relationship between the two.

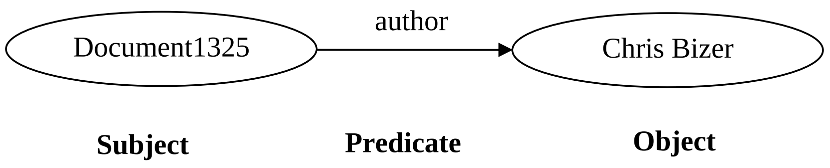

Figure 4.1: A triple representing the piece of information that Document1325 is authored by Chris Bizer.

A set of triples forms a directed labeled graph by sharing subjects and objects. For instance, the pieces of information that Document1325 has the title “Named Graphs” and is authored by Chris Bizer, who has the email address `chris@bizer.de` would be represented by three triples. Figure 4.2 shows the graph formed by these triples.

4.1.2 RDF Nodes

Subject, predicate and object of an RDF triple are RDF nodes. There are three different types of nodes [KC04]:

URI References are nodes that are identified by a globally unique identifier following the URI syntax [BLFM98]. Within RDF, URI references

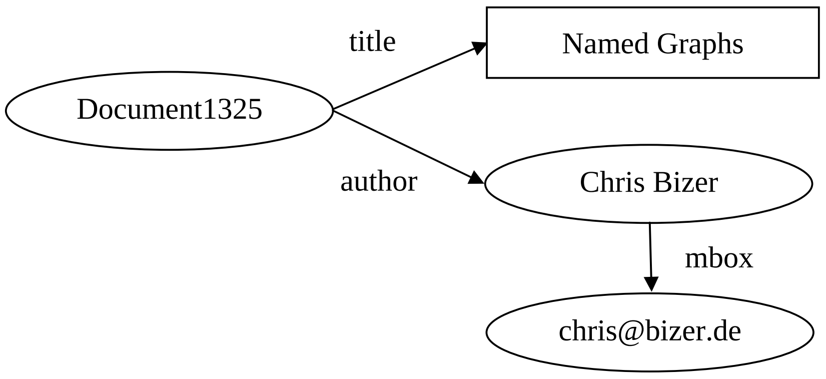

Figure 4.2: A graph formed by three triples.

may be used to identify any kind of object, including Web resources such as HTML documents, real world entities such as products, organizations and persons, and abstract concepts such as terms, classes, or property types. The globally unique identification of a resource eases the integration of information about a resource from distinct information providers. Therefore any resource which might be described by multiple information providers should be identified by a URI reference. A URI owner who assigns a URI reference to a resource should provide representations of the resource [JW04]. This enables information consumers to retrieve authoritative information about resources by dereferencing URIs [SBLH06]. For instance, an information consumer might discover an RDF term on the Web that he does not understand. As an attempt to understand the term, he could dereference the term's URI and retrieve a part of the ontology that defines the unknown term and might relate the term to terms which the information consumer understands.

Blank Nodes. For identifying resources, which need not be referenced from outside the RDF graph in which they occur, the RDF data model provides blank nodes as a second, alternative identification mechanism. Blank nodes are unique nodes that can be used in one or more RDF triples to identify a resource. The term "blank" refers to the fact that blank nodes do not have identifiers. It is only possible to determine whether two blank nodes are the same or not [KC04]. Within implementations of the RDF data model, blank nodes are often assigned identifiers for practical reasons. These identifiers do not have any meaning on data model level. As blank node identifiers are often only unique within the scope of the graph in which they occur, it is possible that distinct blank nodes in different graphs use the same blank node identifier. For instance, a blank node with the identifier `BN1` might be used within one RDF graph to refer to Bob's address. A different blank node, which also uses the identifier `BN1`, might be used in another graph to identify Peter's address. In order to avoid confusions between Bob's and Peter's addresses and to preserve the meaning of both graphs, their blank nodes must be kept distinct. Thus, when RDF graphs are merged within implementations, it might be necessary to rename blank nodes in order to avoid collisions [KC04].

Literals are used to represent property values such as text, numbers, and dates. Literals may be plain or typed: A plain literal is a string combined with an optional language tag. The language tag identifies a natural language, such as English or German [Alv01]. A typed literal is a string combined with a datatype URI. The datatype URI identifies the datatype of the literal. Datatype URIs for common datatypes such as integers, floating point numbers and dates are defined by the XML Schema datatypes specification [BPM01].

Let URI be the set of all URI references and BN the set of all blank nodes. Let PL be the set of all plain literals and TL be the set of all typed literals. Then, the RDF data model can be defined by the following three definitions:

$$Terms = URI \cup BN \cup PL \cup TL \tag{4.1}$$

$$Triples = (URI \cup BN) \times URI \times (URI \cup BN \cup PL \cup TL) \tag{4.2}$$

$$graph \subseteq Triples \tag{4.3}$$

Definition 4.1 defines the set of all RDF terms. The sets URI, BN, PL, and TL are pairwise disjoined. Definition 4.2 defines the set of all RDF triples. As resources may be identified by URI references or blank nodes, the subject of a triple is an element from $URI \cup BN$. The predicate of a triple has to be a URI reference that refers to a property type. Property types have to be identified by URI references in order to clearly capture the semantic difference of types from different RDF vocabularies. For instance, the property type "title" might be used in in one vocabulary to describe the title of a book. Within another vocabulary the same term might be used to refer to academic grades. The object of a triple can be a plain or typed literal or a reference to another resource. Definition 4.3 states that an RDF graph is a subset of $Triples$.

4.1.3 Example RDF Graph

Figure 4.3 shows an RDF graph representing that Document1325 has the title "Named Graphs" and is authored by Chris Bizer, who has the email address `chris@bizer.de`. The figure employs the graphical notation for RDF that is used within the RDF specifications [MM04, Bec04b].

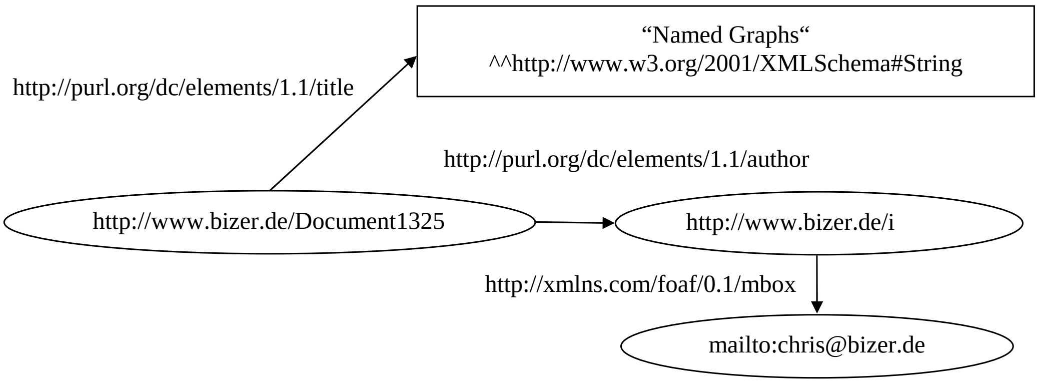
"Named Graphs"
^^http://www.w3.org/2001/XMLSchema#String
http://purl.org/dc/elements/1.1/title
http://purl.org/dc/elements/1.1/author
http://www.bizer.de/Document1325
http://www.bizer.de/i
http://xmlns.com/foaf/0.1/mbox
mailto:chris@bizer.de

Figure 4.3: Example RDF graph.

Within the graph, Chris Bizer and Document1325 are identified by URI references. These globally unique identifies enables other information providers to publish additional information about Chris Bizer and Document1325. Several ontologies have evolved within the RDF community for describing common types of resources such as people and documents[1]. The Dublin Core Element Set [ISO03a], for instance, is widely used to express meta-information about documents; the Friend-of-a-Friend (FOAF) [BM04] vocabulary is used to describe persons. In order to enable information consumers to understand the content of the graph, Chris and Document1325 are described using terms from these ontologies. The title of the document is represented as typed literal. The datatype URI `http://www.w3.org/2001/XMLSchema#String` is used to identify the title as a string. Chris' email address is represented by a URI reference using the `mailto:` URI naming scheme, defined in [BLFM98].

[1]A directory of publicly available RDF ontologies is maintained by the SchemaWeb project, http://www.schemaweb.info/ (retrieved 09/25/2006)

4.2 RDF Schema

The RDF vocabulary definition language RDF Schema [BG04] is a language for describing lightweight ontologies in RDF. RDF Schema ontologies consist of class and property type definitions, inheritance links between types of classes, inheritance links between types of properties, as well as domain and range constraints on properties.

For historic reasons, the RDF Schema language primitives are defined in two seperate namespaces: The `http://www.w3.org/2000/01/rdf-schema#` namespace which is conventionally associated with the `rdfs:` namespace prefix, and the `http://www.w3.org/1999/02/22-rdf-syntax-ns#` namespace which is associated with the `rdf:` namespace prefix. The two basic classes within RDF Schema are [BG04]:

- `rdfs:Class` which is the class of resources that are RDF classes.
- `rdf:Property` is the class of all RDF properties.

An RDF resource is declared to be a class by typing it as an instance of `rdfs:Class` using the `rdf:type` predicate.

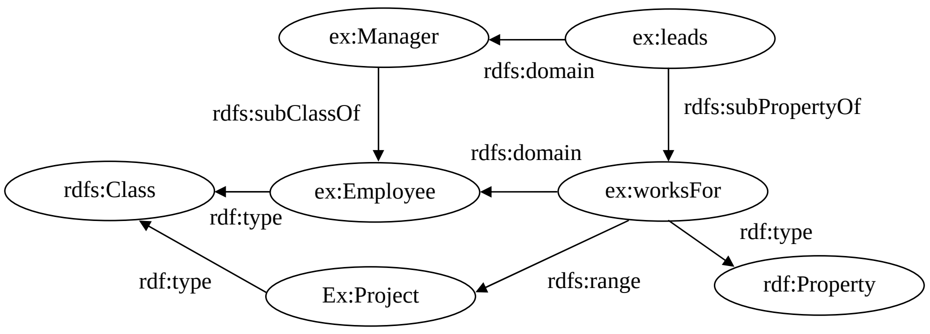

Figure 4.4: Example RDF schema.

Figure 4.4 shows an RDF Schema defining a simple ontology for describing employees, managers, and projects. The terms `ex:Employee` and `ex:Project` are declared to be classes by typing them as instances of `rdfs:Class`. The term `ex:worksFor` is declared to be a property by typing it as an instance of `rdf:Property`.

RDF Schema also provides primitives for describing relationships between classes and properties:

- The property `rdfs:subClassOf` is used to state that all the instances of one class are also instances of another. In our example, `ex:Manager` is declared to be a subclass of `ex:Employee`, implying that all managers are also employees.

- The property `rdfs:subPropertyOf` is used to state that resources related by one property are also related by another. In our example schema, the property `ex:leads` is a subproperty of `ex:worksFor`, meaning that a manager who leads a project also works for the project.

- `rdfs:domain` is used to state that any resource that has a given property is an instance of one or more classes. The domain of the `ex:worksFor` property is declared as `ex:Employee`, meaning that all resources which are described using the `ex:worksFor` property are instances of the class `ex:Employee`.

- The property `rdfs:range` is used to state that all values of a property are instances of one or more classes. In our example, the range of the `ex:worksFor` property is declared as `ex:Project` . Thus, a triple stating that somebody `ex:worksFor` something implies that this something is an instance of the class project.

By using these relational primitives, the author of an RDF schema implicitly defines rules that allow additional information to be inferred from RDF graphs. For instance, the rule that all managers are also employees, enables the triple `ex:Person1 rdf:type ex:Employee` to be inferred from the triple `ex:Person1 rdf:type ex:Manager`.

RDF Schema defines two properties for annotating resources: `rdfs:comment` may be used to provide a human-readable description of a resource. `rdfs:label` may be used to provide a human-readable name for a resource. Beside of being used to annotate of RDF schemata, these properties are also used to provide labels and descriptions for other types of RDF resources. Many RDF visualization tools [HMK05] rely on these properties for displaying RDF data.

The expressiveness of RDF Schema is relatively limited. Application which require a more expressive ontology language may use the Web Onotology Language (OWL) [MvH04, AvH04]. OWL extends RDF Schema with additional modeling primitives that can be used, for instance, to describe more detailed characteristics of properties (`owl:inverseOf`, `owl:equivalentProperty`) and to formulate cardinality constraints (`owl:maxCardinality`, `owl:minCardinality`) as well as value constraints (`owl:allValuesFrom`, `owl:someValuesFrom`) on properties.

4.3 Syntaxes for RDF

To facilitate the interchange of RDF data between information systems a concrete serialization syntax is needed. As XML [BPSMM00] is widely used to exchange information over the Web, it is also an obvious choice for serializing RDF graphs. The RDF/XML Syntax Specification (Revised) [Bec04b] defines a normative syntax for serializing RDF graphs as XML documents.

The RDF/XML syntax is rather verbose and not very readable for humans. Therefore various shorter, plain-text syntaxes for RDF have been developed. The most prominent of these syntaxes are N-Triples [GB04], N3 [BL98], and Turtle [Bec04c]. The N-Triples syntax is used by the RDF specifications as notation for examples and test cases. The N3 syntax, proposed by Tim Berners-Lee, is similar to N-Triples but introduces several abbreviations which simplify writing RDF documents by hand. Beside of these abbreviations, N3 also extends the RDF data model to allow subgraphs within graphs; a feature which is used to represent rules. Many RDF tools have adopted N3 in addition to RDF/XML, but most of them implement only an ad-hoc subset of N3, leaving out the more complex features which go beyond the RDF data model. In light of such development, David Beckett proposed Turtle as a standard subset of N3. Turtle builds on N-Triples and extends the notation with selected features from N3 without extending the RDF data model.

The following sections give a brief overview of RDF/XML and Turtle.

4.3.1 The RDF/XML Syntax

RDF/XML [Bec04b] is the normative syntax for serializing RDF graphs as XML documents and for embedding RDF data into other XML documents. XML represents information as a node-labeled tree. Within RDF, information is represented as an edge-and-node-labeled directed graph. Therefore, serializing RDF as XML requires mapping a graph into a tree structure.

RDF/XML provides different options for this mapping. The basic RDF/XML syntax converts both resources and properties into XML elements which are nested into each other. Figure 4.5 shows an RDF/XML serialization of the example graph shown in Figure 4.3. Line 1 contains the XML declaration, which indicates that the following content is XML. Line 2 starts with an `rdf:RDF` element indicating that the following content represents an RDF graph. RDF/XML uses the XML namespace mechanism [BHL06] to abbreviate URI references. Line 2-4 contain namespace declarations for the RDF, Dublin Core [ISO03a] and FOAF [BM04] vocabularies. Line 5-9 represent the two RDF triples stating that Document1325

```
1.  <?xml version="1.0"?>
2.  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.           xmlns:dc="http://purl.org/dc/elements/1.1/"
4.           xmlns:foaf="http://xmlns.com/foaf/0.1/">
5.    <rdf:Description rdf:about="http://www.bizer.de/Document1325">
6.       <dc:title rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
7.         Named Graphs </dc:title>
8.        <dc:creator rdf:resource= "http://www.bizer.de/i" />
9.    </rdf:Description>
10.   <rdf:Description rdf:about="http://www.bizer.de/i">
11.         <foaf:mbox rdf:resource="mailto:chris@bizer.de" />
12.   </rdf:Description>
13. </rdf:RDF>
```

Figure 4.5: Basic RDF/XML serialization of the example graph.

has the title Named Graphs and is authored by Chris Bizer. As both triples describe Document1325, they are represented by a single `rdf:Description` element. The `rdf:about` attribute contains the URI of the resource that is being described. The `dc:title` subelement represents the `dc:title` predicate. The element contains the title of the document. The `rdf:datatype` attribute represents the datatype of the title. The `dc:creator` element in line 8 contains a reference to Chris Bizer, whose email address is given by the `rdf:Description` and the `foaf:mbox` elements in line 10-12.

Beside of this basic syntax, RDF/XML offers a set of abbreviations which are intended to increase the readability of RDF/XML serializations and make the code look similar to the way information is usually represented in XML:

Nested Description Elements. Instead of referring to a resource using an `rdf:resource` attribute, it is also possible to nest the `rdf:Description` element which describes the referenced resource below the referring property element. Figure 4.6 shows an alternative serialization of our example graph. Instead of using two `rdf:Description` elements directly below the `rdf:RDF` element, the `rdf:Description` element about Chris is nested inside the `dc:creator` element.

XML Base. RDF/XML syntax supports the XML base mechanism [Mar01] which allows URI references to be abbreviated relative to a base URI. This mechanism is used in Figure 4.6. Line 5 defines `http://www.bizer.de/` as base URI for all elements inside the `rdf:RDF` element. The `rdf:ID` elements in line 6 and 10 contain abbreviated URI references which are interpreted relative to the base URI.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:foaf="http://xmlns.com/foaf/0.1/"
         xml:base="http://www.bizer.de/" >
 <rdf:Description rdf:ID="Document1325">
 <dc:title rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Named Graphs </dc:title>
  <dc:creator>
   <rdf:Description rdf:ID="i">
    <foaf:mbox rdf:resource="mailto:chris@bizer.de" />
   </rdf:Description>
  </dc:creator>
 </rdf:Description>
</rdf:RDF>
```

Figure 4.6: Abbreviated RDF/XML serialization of the example graph.

Properties as Attributes. RDF properties with plain literal values may be serialized as attributes of a `rdf:Description` element instead of serializing them as subelements.

Typed Nodes. As it is fairly common that resources have `rdf:type` properties, RDF/XML provides a special abbreviation for typed nodes. In this abbriviation, the `rdf:type` property and its value are removed, and the `rdf:Description` element for that node is relpaced by an element whose name is the QName corresponding to the value of the removed `rdf:type` property [MM04].

These abbreviation options allow an RDF graph to be serialized in a variety of different ways. The different serialization options complicate the parsing of RDF/XML documents and make it difficult to use RDF/XML together with other XML technologies such as XSLT [Cla99], XPath [CD99], and XQuery [BCF$^+$05], as theses technologies assume a single, fixed document structure [CS04a, Bec04a]. Based on the experience with plain-text formats like N3 or Turtle, several authors [Bec03a, CS04a] argue that XML serializations for RDF should be more clearly oriented towards the triple structure of RDF graphs and be minimal in the number of alternate forms for the same RDF graph.

4.3.2 The Turtle Syntax

The Turtle [Bec04c] syntax for RDF is a text-based serialization format which closely mirrors the triple structure of RDF graphs. The nodes of each triple are serialized in subject, predicate, object order. URI references are enclosed with brackets. URI references may be abbreviated using a base URI and namespace prefixes. Namespace prefixes are introduced by `@prefix` clauses. Literals are enclosed by quotation marks. Blank node identifiers are preceded with an _:. Two shortcuts are provided to combine several triples: A semicolon introduces another property of the same subject. A comma introduces another object with the same property and subject.

Figure 4.7 shows a Turtle serialization of our example graph. Line 1-3 define the namespace prefixes for XML Schema, FOAF, and Dublin Core. Line 5 represents the triple about Chris' email address. The predicate of the triple is abbreviated using the FOAF namespace prefix defined in line 2. Lines 6-8 represent the two triples about Document1325. As both triples share the same subject, a semicolon after the first triple indicates that the next line contains another property of the same subject.

```
1. @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
2. @prefix foaf: <http://xmlns.com/foaf/0.1/> .
3. @prefix dc: <http://purl.org/dc/elements/1.1/> .
4.
5. <http://www.bizer.de/i> foaf:mbox <mailto:chris@bizer.de> .
6. <http://www.bizer.de/Document1325>
7.       dc:title "Named Graphs"^^xsd:String ;
8.       dc:creator <http://www.bizer.de/i> .
```

Figure 4.7: Turtle serialization of the example graph.

4.4 Query Languages for RDF

Several query languages have been developed for the RDF data model. Within these languages, queries are expressed as a set of triple patterns containing variables. The triple patterns are matched against an RDF graph resulting in a set of matching solutions. A matching solution assigns values to the variables in the triple patterns. The solution set may be further constrained by posing conditions on variable values. A comparison of different RDF query languages is presented in [HBEV04].

Based on the experience with different RDF query languages, the W3C Data Access Working Group [Pru06] is developing SPARQL [PS05] as a standardized query language for RDF. Figure 4.8 shows an example SPARQL query. The query selects all email addresses that contain the string `@bizer.de`. The first line of the query defines a namespace prefix for the FOAF namespace. The SELECT clause in line 2 specifies the variables which are included into the solution set. Line 3 contains a triple pattern. The pattern matches all triples having the predicate `foaf:mbox`. The subjects and objects of these triples are bound to the variables `?x` and `?mbox`. The FILTER clause in line 4 filters the solution set to contain only solutions that assign a value containing the string `"@bizer.de"` to the variable `?mbox`.

```
1. PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2. SELECT ?mbox
3. WHERE { ?x foaf:mbox ?mbox .
4.         FILTER regex(str(?mbox), "@bizer\.de") }
```

Figure 4.8: Example of a SPARQL query.

4.5 RDF Reification

RDF provides a mechanism for representing meta-information about RDF triples, called reification [BG04, Hay04]. In theory, RDF reification can be used to represent quality-related meta-information about RDF triples. This chapter introduces the RDF reification mechanism and discusses its practical utility.

The basic idea of RDF reification is to represent an RDF triple as a resource in RDF. This allows other triples to describe properties of the triple, such as its creator or creation date. RDF provides a built-in vocabulary to represent reified triples. The vocabulary consists of the class `rdf:Statement`, and the properties `rdf:subject`, `rdf:predicate`, and `rdf:object`. Reified triples are instances of the class `rdf:Statement`. The `rdf:subject`, `rdf:predicate`, and `rdf:object` properties are used to describe the components of the triple.

Figure 4.9 shows the reification of our example graph together with information about the creator and the creation date of each triple. Lines 7-10 contain the reification of the triple about Chris' email address. Line 7 states that the resource identified by the URI reference `ex:triple1` is an RDF statement. Lines 8-9 contain the information that the subject of the triple refers to the resource identified by `<http://www.bizer.de/i>`, the predicate of the triple refers to the resource identified by `foaf:mbox`, and that the object of the triple refers to `<mailto:chris@bizer.de>`. Line 22-23 contain provenance information about `ex:triple1`, stating that the triple has been created by Chris on 2006-02-03.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefic ex: <http://www.bizer.de/ExampleDocument/> .

ex:triple1 rdf:type rdf:Statement ;
           rdf:subject <http://www.bizer.de/i> ;
           rdf:predicate foaf:mbox ;
           rdf:object <mailto:chris@bizer.de> .

ex:triple2 rdf:type rdf:Statement ;
           rdf:subject <http://www.bizer.de/Document1325> ;
           rdf:predicate dc:title ;
           rdf:object dc:title "Named Graphs"^^xsd:String .

ex:triple3 rdf:type rdf:Statement ;
           rdf:subject <http://www.bizer.de/Document1325> ;
           rdf:predicate dc:creator ;
           rdf:object dc:title <http://www.bizer.de/i> .

ex:triple1 dc:creator <http://www.bizer.de/i> ;
           dc:date "2006-02-03"^^xsd:date .

ex:triple2 dc:creator <http://www.bizer.de/i> ;
           dc:date "2006-02-03"^^xsd:date .

ex:triple3 dc:creator <http://www.bizer.de/i> ;
           dc:date "2006-02-03"^^xsd:date .
```

Figure 4.9: Reification of the example graph.

RDF reification provides a mechanism for representing meta-information about triples but tries to stay inside the bounds of a pure triple data model at the same time. This approach has some substantial drawbacks:

Triple Bloat. RDF reification increases the number of triples in an graph significantly. An effect which is called “triple bloat” [CS04a]. Describing the elements of a triple using the reification vocabulary causes an at least threefold increase alone.

Querying Reified Statements. It is rather cumbersome to query information which is represented as reified statements using RDF query languages such as SPARQL. As a single reified statement is represented by multiple triples, queries over reified statements also involve multiple triple patterns for a single statement and therefore quickly become unreadable and confusing. Figure 4.10 shows a SPARQL query to retrieve all information about people who have an email address against the reification our example graph. If a query engine is not especially optimized for this kind of queries it would answer them slowly, as evaluating multiple triple patterns implies a join for each pattern [MK03].

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?sub ?pred ?obj
WHERE  { ?statement1 rdf:subject ?subj .
        ?statement1 rdf:predicate ?pred .
        ?statement1 rdf:object ?obj .
        ?statement2 rdf:subject ?subj .
        ?statement2 rdf:predicate foaf:mbox }
```

Figure 4.10: SPARQL query against a reified graph.

Redundant Meta-Information. RDF reification requires meta-information to be attached separately to each reified statement. This further increases the size of the graph and might lead to inconsistencies when meta-information is changed. In order to allow meta-information to be expressed at a higher level, Graham Klyne proposes to group reified statements together using an `rdf:Bag` [BG04] and to attach meta-information to this bag instead of having to attach it separately to each reified statement [Kly00]. His approach eliminates redundant meta-information but leave the other problems of reification untouched, as each original triple is still described by at least three reification triples plus one extra triple to relate the reified statement to the bag.

Single Level of Granularity. The RDF reification mechanism allows meta-information to be expressed only on a single, fixed level of granularity. Within most information exchange and publication scenarios, RDF information is provided as graphs consisting of multiple statements. These scenarios therefore do not require meta-information about individual statements and it would be more suitable to use a mechanism that allows meta-information to be expressed at different levels of granularity.

Because of these problems, the RDF reification mechanism is hardly used by RDF applications and even members of the former W3C RDF Core working group, who have designed the mechanism, are recommending not to use it [CS04a, CBHS05b, MK03].