IOWA STATE UNIVERSITY
Digital Repository

Retrospective Theses and Dissertations

Iowa State University Capstones, Theses and Dissertations

2007

D-RDF: Dynamic Resource Description Framework

Kamna Jain
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

 Part of the Computer Sciences Commons

Recommended Citation
Jain, Kamna, "D-RDF: Dynamic Resource Description Framework" (2007). *Retrospective Theses and Dissertations*. 14853.
https://lib.dr.iastate.edu/rtd/14853

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

www.manaraa.com

D-RDF: Dynamic Resource Description Framework

by

Kamna Jain

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee:
Akhilesh Tyagi, Co-major Professor
Arun K. Somani, Co-major Professor
Shashi K. Gadia

Iowa State University

Ames, Iowa

2007

Copyright © Kamna Jain, 2007. All rights reserved.

www.manaraa.com

UMI Number: 1447541

UMI

UMI Microform 1447541

Copyright 2008 by ProQuest Information and Learning Company.
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

www.manaraa.com

DEDICATION

Dedicated to my Parents.

www.manaraa.com

TABLE OF CONTENTS

LIST OF TABLES v

LIST OF FIGURES vi

ACKNOWLEDGEMENTS viii

ABSTRACT ix

CHAPTER 1. Introduction 1

1.1 RDF 1

1.2 Semantic Web 2

1.3 Dynamic-RDF 4

CHAPTER 2. The Resource Description Framework 8

2.1 Terminology 9

2.1.1 Sample RDF Graph 13

2.2 RDF/XML 14

CHAPTER 3. The D-RDF FrameworkDesign and Implementation 17

3.1 XML to RDF/XML Mapping 18

3.2 XML Path Language 18

3.3 Design 20

3.3.1 Design Goals 22

3.4 Implementation 22

www.manaraa.com

3.4.1 JENA 23
3.4.2 SAXON 24
3.4.3 Execution Semantics 24
3.5 VALIDATION 26
CHAPTER 4. Discussion of Implemented Use-Cases 27
4.1 Analysis of Research Publications 29
4.2 Minimum Priced Computer 31
4.3 Construct a Travel Package 34
4.4 Transitive Closure - An Experimental Feature 35
4.5 Significance of D-RDF 36
CHAPTER 5. Related Work and Motivation 39
5.1 RDF and Related Technologies 39
5.1.1 RDF and XML 39
5.1.2 RDF and RSS 41
5.1.3 RDF and RDB 42
5.2 How Dynamic-RDF Fits in the Semantic Web Architecture 43
5.2.1 The Semantic Web Architecture 44
5.2.2 D-RDF and Yahoo Pipes 45
5.3 Path Traversal Languages 46
CHAPTER 6. Contribution and Future Work 48
6.1 Key Contributions 48
6.2 Challenges and Limitations 48
6.3 Future Work 49
BIBLIOGRAPHY 50

www.manaraa.com

LIST OF TABLES

Table 5.1 Mapping RDF Triples to a table in a database 42

www.manaraa.com

LIST OF FIGURES

Figure 1.1 An RDF Triple 3

Figure 1.2 A D-RDF Model 6

Figure 2.1 An RDF Triple is made of *Subject, Predicate, Object* 10

Figure 2.2 An RDF Literal 12

Figure 2.3 Using RDF Schema to describe the Property - *foaf:gender* 12

Figure 2.4 A Connected RDF Graph [27] 13

Figure 2.5 An RDF Graph 14

Figure 2.6 RDF/XML Serialization of the graph in Figure 2.5 15

Figure 2.7 RDF/XML Serialization of the cycle in Figure 2.4 16

Figure 3.1 Usage of the *dynamic intermediate step* 21

Figure 3.2 Sample JENA Code [14] 23

Figure 4.1 Base RDF publication data 30

Figure 4.2 Input D-RDF model with dynamic property for Use-Case 4.1 30

Figure 4.3 Output D-RDF model for Use-Case 4.1 30

Figure 4.4 Minimum Priced Computer 32

Figure 4.5 Project and Merge 33

Figure 4.6 Resource describing *User Preferences*in the input D-RDF file 35

Figure 4.7 Resource with *dynamic properties* to select best package 36

www.manaraa.com

Figure 5.1 Single RDF Representation 40
Figure 5.2 Multiple XML Representations 40
Figure 5.3 The Semantic Web Architecture 45

www.manaraa.com

ACKNOWLEDGEMENTS

I take this opportunity to express gratitude towards everybody who helped me through the completion of this research and the Master's degree program here at Iowa State University.

I would like to thank Drs. Akhilesh Tyagi and Arun Somani for the patience and encouragement they offered. I am very grateful to Dr. Tyagi for his insightful guidance on every aspect of this Thesis. Without his direction this work would not have taken the form it is in today.

I am thankful to Dr. Somani for trusting my capabilities and giving me the opportunity to pursue my graduate degree at Iowa State University. I am also thankful to the Information Infrastructure Institute (ICUBE) for funding my research work. My thanks are also due to Dr. Shashi Gadia for being on my committee and for his help and contribution towards this work.

The credit for my M.S. degree also goes to my family. The support and encouragement from my Parents has been invaluable throughout the course of this work.

Last but not the least, I would like to thank all my friends here at Iowa State. They made this experience a memorable one with all the fun I had amidst them. Special thanks to Satyadev Nandakumar for sharing his knowledge on XML and XPATH with me. Not to forget all the services provided by the members of the ECpE Graduate Office, especially Pam Myers.

www.manaraa.com

ABSTRACT

Semantic Web is described as the Web of Data, as opposed to the World Wide Web which is a Web of Documents. As research in the field of Semantic Web is gaining momentum, the focus is shifting on the effective representation of the data that constitutes the Semantic Web. RDF or the Resource Description Framework is the W3C standardized language for describing the semantics of the data and hence, sharing it's meaning across applications. In RDF, all entities are modeled as resources and facts about these resources are asserted in terms of properties and their values.

In this thesis, we propose a computation model for RDF called Dynamic RDF or D-RDF. While RDF models are restricted to describing resources in terms of their assertive or static properties, D-RDF is a generalization of RDF where there exist the assertive properties of the resources along with certain *dynamic* properties that operate on the values of existing properties and infer new data. In such a model, the information carries the semantics with it in the form of computing methods. In other words, whereas RDF represents semantically enhanced data, D-RDF represents both data and the programs that operate on the data. This design ensures that when the D-RDF model is processed, the dynamic properties would operate on the current values of the base data and hence, the values of the dynamic properties will always be consistent with changes that occur on that data. Hence, we develop a model of *context-sensitive* semantics and implement an interpretation engine for this language. The power of D-RDF is demonstrated by implementing use-cases of varying levels of complexity that highlight how highly customized data models can be constructed with D-RDF to represent

www.manaraa.com

information in a form that does not already exist.

www.manaraa.com

CHAPTER 1. Introduction

After a decade of enjoying the power to obtain information on any topic under the Sun from the World Wide Web at the click of a mouse, the world is now preparing for a transition from a Web of Documents, as we see it today, to a Web of Data the Semantic Web. The WWW was a brainchild of Sir Tim Berners-Lee and so is the Semantic Web but, whereas most of the Webs content today is designed for humans to read, the Semantic Web is all about adding structure and expressing meaning such that the machines can interpret the data. Hence, as opposed to the WWW where computers are just required to parse web pages for layout and routine processing, when its extension comes into existence in the form of Semantic Web, computers will be able to process the semantics reliably.

1.1 RDF

RDF [1] stands for Resource Description Framework. As suggested by the name, it is a language standardized by the W3C intended to represent every piece of information that exists, about a Web resource. The Web Resource could be a resource that can be directly retrieved from the Web or it could just be an entity that can be identified on the Web and the description of the resource consists of assertive, atomic properties like its *name* that would help identify the resource, a *link* that would identify its location on the Web (whenever applicable) and a *date of creation* to specify when it came into existence. Fundamentally, RDF is about adding structure to the voluminous data that exists in todays electronic world as plain text.

www.manaraa.com

RDF is the foundation of Semantic Web [2]. Semantic Web is a Web-of-Data as opposed to the existing Web-of-Documents and hence, RDF is to Data what HTML is to Web documents. More so, the RDF properties that describe relations among the data on the Web are compared to the hyperlinks on the current Web. While the hyperlinks link to documents on the Web, the RDF properties can link any two resources. This notion of being able to semantically link resources like, documents, images, people, concepts etc., is an important contribution of RDF. It makes explicit the contextual relationships that are implicit in the current Web.

The underlying idea of RDF [1] is to represent every resource using a Web identifier, more formally known as a Uniform Resource Identifier or URI and all information is encoded in the form of *triples* of {subject, predicate, object}. In other words, triples are the basic building blocks of RDF. They are simple, atomic statements about the resources being described where the resource itself is the *subject*, the property being described is the *predicate* and the value of the property for this particular resource is the *object*. Most commonly all RDF data is depicted as a graph where the nodes represent the resources and arcs represent their properties or relationships with other resources. Additionally, RDF also uses an XML-based syntax, RDF/XML [3], for recording and exchanging these graphs.

The following figure depicts the graph representation of a triple where a web page is a resource, property being creator and the value of the creator is another resource identifying the person who created the page.

A formal definition of RDF follows in Chapter 2.

1.2 Semantic Web

Semantic Web [2] is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners and according to the World Wide Web Consortium the Semantic Web provides a common framework that allows data to be shared and reused

www.manaraa.com

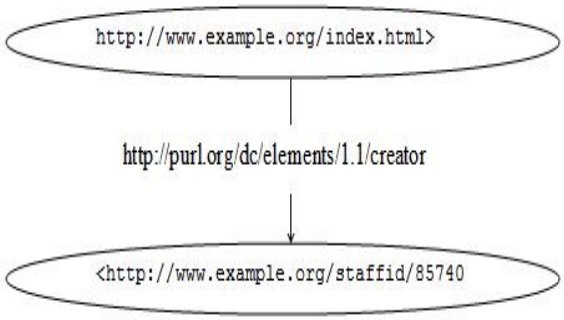

Figure 1.1 An RDF Triple

across application, enterprise, and community boundaries. The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation [4].

A simple use-case would explain how this might be useful. Consider a scenario where Alice records all data about her daily schedule and all pictures taken by her in two different RDF models named Calendar and Image. The purpose of creating RDF models for all the data is to add structural detail to the data so that based on a particular property, say, *date* (assuming that it is used in both the models); Alice can easily determine what she was doing from her Calendar RDF model when she clicked a particular photograph, details of which are in the Image RDF model.

Similarly, volumes of raw information that may answer numerous such questions may indeed exist on the Web, but they are not in a machine-usable form. The Semantic Web addresses this problem in two ways [4]. First, it enables communities to expose their data semantics so that a program doesn't have to strip the formatting, pictures and ads from a Web page to identify the relevant bits of information. Secondly, it allows people to write (or generate) files

www.manaraa.com

which explain - to a machine - the relationship between different sets of data. The Resource Description Framework (RDF) is a W3C recommendation to enable Semantic Web, i.e. it is a framework that helps add structure to the data.

Finally, as Sir Tim Berners-Lee puts it [5], "Semantic Web is not Artificial Intelligence. The concept of machine-understandable documents does not imply artificial intelligence which allows machines to comprehend human mumblings. It only indicates a machine's ability to solve a well-defined problem by performing well-defined operations on existing well-defined data. Instead of asking machines to understand people's language, it involves asking people to make the extra effort of defining any data they deal with."

1.3 Dynamic-RDF

As is clear from the description of RDF in section 1.1, at present the properties that describe relations among data are atomic, say, *title* of a web page, *creator* of a book or *price* of an item. These are assertive properties that can be used to describe resources and result in RDF models which depict relationships among different resources. As the size of the available RDF data increases, so does the effort required to derive meaningful information from the data.

In our work, which is described in this thesis, we have made an attempt to define a generalization of RDF called Dynamic RDF (D-RDF). D-RDF is a *programming language* to specify the derived semantics. Moreover, the language has the same syntax as RDF. Hence, the raw data and programs operating on this data look alike. With Dynamic RDF, users can not only add structure to the data at their disposal, as is possible with RDF, but also, capitalize on the availability of structured data by adding *programs* that can access the data and operate on it. This combination of data and programs has the potential to derive information by correlating the existing static properties and processing the data represented by them. Traditionally, a suitable query language would be used to query the existing RDF models and these queries

www.manaraa.com

would not exist as part of the RDF data model. The important point in this implementation is that these user-defined programs take the form of RDF properties and hence, they exist inside the RDF model with other such properties and data. These special properties are called *Dynamic Properties* and term Dynamic RDF comes from the idea of introducing such programs as properties in an RDF model. D-RDF involves defining and performing actions on the existing details to infer new ones. It is dynamic because the information objects now not only contain raw data but also the semantics or computing methods that operate on the raw data. This allows for late binding between these semantic data models and physical resources. One among many possible applications of this paradigm is to specify *device-sensitive* semantics that will allow for dynamic adaptation of the same data based on the capabilities of different devices.

In other words, the Dynamic model is built above the existing RDF model as an extension with the difference being in the nature of the properties that describe relations. So, apart from having static properties we also have dynamic properties which go beyond just stating some facts. These properties are such that they create new relations based on the existing ones; relations or properties that can not be created when a stand alone program or query is used to retrieve the information.

Currently, the programs that make up the dynamic properties are XPath [6] expressions. XPath is a standardized path traversal language used to navigate through XML documents. The fact that RDF employs RDF/XML format for serialization of its models makes it appropriate to choose XPath as the preferred language.

Figure 1.2 shows a simple example of dynamic RDF. Here, we have an RDF model listing all the publications of an author as resources and being described with static properties like *title and status*. (The number of properties being used is limited to maintain simplicity of the example.) The property *status* indicates whether the publication has been *accepted or rejected*. If there is a need to know the total number of papers that have been *accepted*, the first instinct

www.manaraa.com

would be to write a query that would perform this operation for us. Unfortunately, SPARQL [7], which is the W3C standardized query language for RDF does not support aggregate functions at this time. We then turn to more matured technologies like XPath for help and come up with an XPath expression that can do the count for us. Although this solution works out at the moment yet, if we need to repeat this in future, we need to write the expression again. In this case it is a simple one but, there will be instances when complex XPath expressions would be needed to extract the required information, with an added potential for reuse.

In such a situation, adding a dynamic property (XPath expression) to count this number would be perfect. The advantages being, getting results that are always consistent with any modifications that may occur in the base data and the flexibility of re-using this property to create new ones in the future.

```
<rdf:Description rdf:about="http://ecpe.univ.edu/ProfPubs/pub1">
  <pub:status>Accepted</pub:status>
  <dc:title>Publication1</dc:title>
</rdf:Description>
<rdf:Description rdf:about="http://ecpe.univ.edu/ProfPubs/pub2">
  <pub:status>Rejected</pub:status>
  <dc:title>Publication2</dc:title>
</rdf:Description>
<rdf:Description rdf:about="http://ecpe.univ.edu/ProfPubs/pub3">
  <pub:status>Rejected</pub:status>
  <dc:title>Publication3</dc:title>
</rdf:Description>
<rdf:Description rdf:about="http://ecpe.univ.edu/ProfPubs/pub4">
  <pub:status>Accepted</pub:status>
  <dc:title>Publication4</dc:title>
</rdf:Description>
    .
    .
<rdf:Description rdf:about="http://ecpe.univ.edu/ProfPubs/pub50">
  <pub:status>Accepted</pub:status>
  <dc:title>Publication4</dc:title>
</rdf:Description>

<rdf:Description rdf:about="http://ecpe.univ.edu/ProfPubs>
  <dc:numberAccepted>
      fn:count(//rdf:Description/pub:status[text()='Accepted'])
  </dc:numberAccepted>
</rdf:Description>
```

Figure 1.2 A D-RDF Model

We believe that this is a novel contribution in the field of RDF and Semantic Web, which

www.manaraa.com

would help achieve one of the foremost goals of the Semantic Web - data integration. The use-case described in Section 1.1 is a simple example of integrating data from different sources and deriving a useful conclusion out of it.

We claim that D-RDF achieves this goal because these 'dynamic-property enriched' RDF models are easy to create, facilitate easy nesting of the properties and hence, reuse and the outcome is a semantic entity which helps users comprehend the underlying data from different perspectives. It also facilitates the discovery of non-trivial semantic relations that are alwasy consistent with the existing data. Not to forget, these D-RDF models are extensible. As SPARQL matures with time, it should be possible to give users the choice of using a query language with maximum expressivity.

www.manaraa.com

CHAPTER 2. The Resource Description Framework

The W3C document on RDF Semantics [8] introduces RDF as an assertional language intended to be used to express propositions using precise formal vocabularies, particularly those specified using the RDF Schema (RDFS) [9], for access and use over the World Wide Web, and is intended to provide a basic foundation for more advanced assertional languages with a similar purpose. The overall design goals emphasize generality and precision in expressing propositions about any topic, rather than conformity to any particular processing model.

In other words, the underlying idea of the Resource Description Framework is that every entity, better known as a *resource*, can be described in terms of statements that assert the values for corresponding properties. Consider a plain English sentence, ***http://www.example.org/index.html*** *has a* ***creator*** *whose value is* ***John Smith***. RDF uses a particular terminology for talking about the various parts of such statements. Specifically, the part that identifies the thing the statement is about (the Web page in this example) is called the *subject*. The part that identifies the property or characteristic of the subject that the statement specifies (creator, creation-date, or language in this example) is called the *predicate*, and the part that identifies the value of that property is called the *object*.

The development of RDF has been motivated by the following uses, among others [10]:

- Web metadata: providing information about Web resources and the systems that use them (e.g. content rating, capability descriptions, privacy preferences, etc.)
- Applications that require open rather than constrained information models (e.g. schedul-

www.manaraa.com

ing activities, describing organizational processes, annotation of Web resources, etc.)

- To do for machine processable information (application data) what the World Wide Web has done for hypertext: to allow data to be processed outside the particular environment in which it was created, in a fashion that can work at Internet scale.

- Collaboration among applications: combining data from several applications to arrive at new information.

- Automated processing of Web information by software agents: the Web is moving from having just human-readable information to being a world-wide network of cooperating processes. RDF provides a world-wide lingua franca for these processes.

2.1 Terminology

- *Subject*: The Subject of an RDF statement is the resource being described in the statement and may be represented by a URI reference or a blank node. If the subject is a blank node it is typically a container for sub-properties.

- *Predicate*: The Predicate establishes the relationship between a subject and an object in an RDF statement and makes the object value a characteristic of the subject. Unlike the Subject and Object, the predicate must always be represented by a URI. Note that the terms predicate and property will be used interchangeably throughout this document.

- *Object*: The Object is the property value that is mapped to a subject by the predicate. An object can be an RDF URI reference, an RDF literal, or a blank node. If the object is a blank node, it serves as a parent property to encapsulate a group of sub-properties.

- *Resource*: A Resource is an entity that can take the role of a subject, predicate or object in an RDF statement. Identified by a URI, a resource is not limited to be something that

www.manaraa.com

is network-accessible. In fact, a resource can be anything that is not network-accessible such as human beings and books or abstract concepts that do not even exist physically.

- *RDF Triple*: The construct for writing natural language statements in terms of a subject, predicate and object as explained in the previous section is called an RDF Triple.

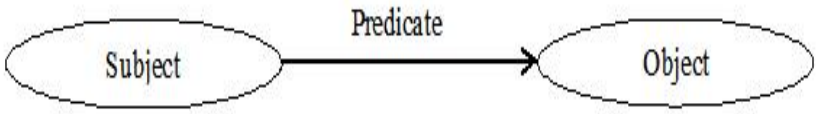

Figure 2.1 An RDF Triple is made of *Subject, Predicate, Object*

- *RDF Graph*: An RDF graph is a set of RDF triples. The set of nodes of an RDF graph is the set of subjects and objects of triples in the graph. A *subgraph* of an RDF graph is a subset of the triples in the graph. A triple is identified with the singleton set containing it, so that each triple in a graph is considered to be a subgraph [10].

- *URI*: A Uniform Resource Identifier is a simple and extensible string of characters that is used to identify any abstract or physical resource. Any person or organization can create URIs and because of this generality, RDF uses URIs as the basis of its mechanism for identifying the subjects, predicates, and objects in statements. More precisely, RDF uses URI Reference (URIRef) which is a URI followed by an optional fragment identifier (#). For example, the URI reference *http://www.w3.org/1999/02/22-rdf-syntax-ns#Description* consists of the URI *http://www.w3.org/1999/02/22-rdf-syntax-ns#* and the fragment identifier *Description* [1].

- *XML Namespace*: An XML namespace is a collection of names which are identified by URI references and used in an XML document as element types and attribute names.

www.manaraa.com

In RDF, the XML namespaces are used to define the collection of names of resources and properties. The primary use of such names is to enable identification of logical structures in documents by software modules such as query processors. For example, *http://www.w3.org/1999/02/22-rdf-syntax-ns#* is the namespace URI for RDF terminology [11].

- *XML Qualified Name*: Also known as QName, Qualified Name is shorthand for a URI reference. A QName contains a *prefix* that has been assigned to a namespace URI, followed by a colon, and then a *local name*. So, for example, if the QName prefix rdf is assigned to the namespace URI *http://www.w3.org/1999/02/22-rdf-syntax-ns#*, then the QName *rdf:Description* is shorthand for the URIRef *http://www.w3.org/1999/02/22-rdf-syntax-ns#Description* [1].

- *Vocabulary*: An RDF vocabulary or schema is a defined set of predicates that can be used in an application. The vocabulary also defines the *domain and range* of the predicates in terms of RDF classes. The domain indicates that the predicate can be used to describe the instances of all the classes (that are mentioned as domain) and range of the property indicates the type of object that can be used when the particular property appears in a triple. For example, Dublin Core Element Set is a vocabulary with members like *title, creator, subject, description* and many more elements that can be used to describe publications.

- *Literal*: Literals are constant values that appear in RDF statements as Object values. They are used to identify values such as numbers and dates by means of a lexical representation. Note that a literal may be the object of an RDF statement, but not the subject or the predicate. The triple in Figure 2.2 depicts that the age of particular employee is 25. Here 25 is the literal.

www.manaraa.com

Figure 2.2 An RDF Literal

- *RDFS*: RDFS stands for RDF Schema [9] and is the base RDF vocabulary that the RDF user community can use to develop their own vocabularies. It provides facilities needed to describe the classes of resources and the properties and to indicate which classes and properties are intended to be used together. For example, the property gender is defined as shown in Figure 2.3 where RDFS is used to define its domain and range.

```
<rdf:Property rdf:about=http://xmlns.com/foaf/0.1/gender>
  <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Agent"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>
```

Figure 2.3 Using RDF Schema to describe the Property - *foaf:gender*

The *domain* of a property specifies the class whose elements can be described using the property and the *range* of a property defines the values the property can take. Hence, according to this example, the property *foaf:gender* can be used to describe any resource of type http://xmlns.com/foaf/0.1/Agent and the value that it takes would be a literal, say, Male or Female.

www.manaraa.com

2.1.1 Sample RDF Graph

The RDF graph shown in Figure 2.4 uses the *foaf* (Friend of a Friend) RDF vocabulary to provide information about Jon Foobars blog. It is also an illustration of a connected graph where the object of one triple is the subject of another.

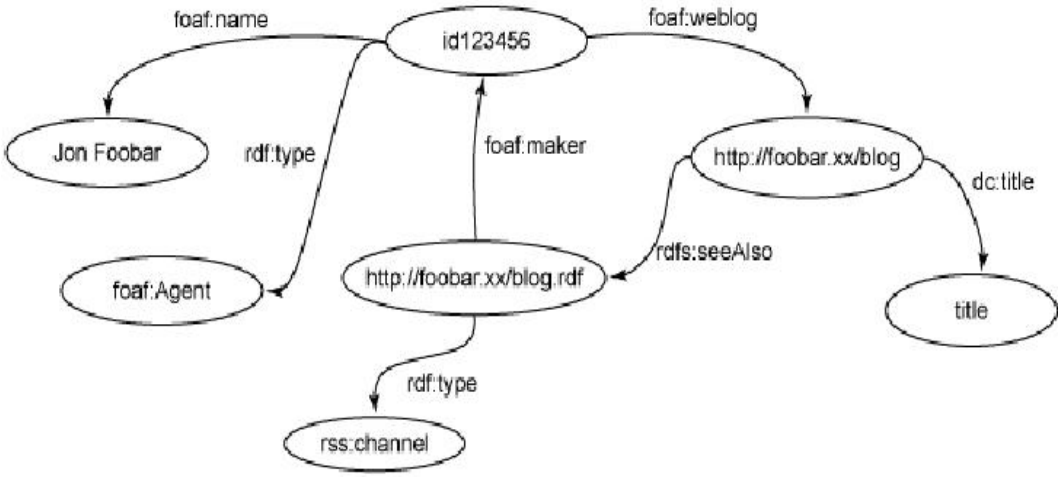

Figure 2.4 A Connected RDF Graph [27]

The following is a mapping between a few of the terms described above and the elements of this graph.

- Subjects - {id 23456, http://foobar.xx/blog, http://foobar.xx/blog.rdf}
- Objects - {Subjects, title, rss:channel, foaf:Agent, Jon Foobar}
- Predicates - {foaf:weblog, rdf:type, foaf:name, dc:title}
- URIs - {http://foobar.xx/blog, http://foobar.xx/blog.rdf}
- XML Qualified Names - {foaf:weblog, rss:channel}

www.manaraa.com

2.2 RDF/XML

Conceptually an RDF model is represented as a graph of triples but, RDF also provides XML syntax called *RDF/XML* [3], for writing down and exchanging RDF graphs. Unlike triples, which are intended to be a shorthand notation, RDF/XML is the normative syntax for serializing RDF models. The syntax and usage of RDF/XML will become more comprehensible with the help of the following example [1] where the resource being described is a web page and *creation-date, creator and language* are the properties used to describe the resource.

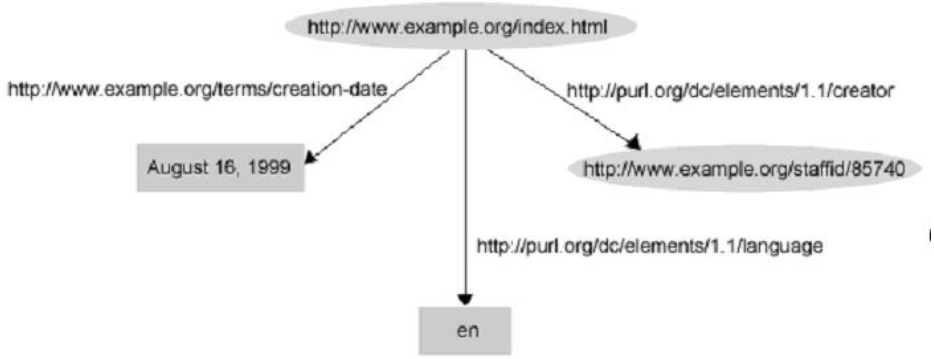

Figure 2.5 An RDF Graph

Figure 2.6 displays a typical RDF/XML serialization. The *rdf:RDF* element in line 2 indicates that the following XML content (starting here and ending with the *</rdf:RDF>* in line 9) is intended to represent RDF. Following the *rdf:RDF* on the same line is an XML namespace declaration, represented as an *xmlns* attribute of the rdf:RDF start-tag. Similarly, *dc* and *exterms* are the other namespaces being used in the serialization. What we see in lines 5-9 is the way RDF/XML represents the RDF statements depicted in the graph. The *rdf:Description* start-tag in line 5 indicates the start of a description of a resource and goes on to identify the resource the statement is about, the subject of the statement (in this case, http://www.example.org/index.html), using the *rdf:about* attribute to specify the URIref

www.manaraa.com

```
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:exterms="http://www.example.org/terms/">

5.    <rdf:Description rdf:about="http://www.example.org/index.html">
6.     <exterms:creation-date>August 16, 1999</exterms:creation-date>
7.     <dc:language>en</dc:language>
8.     <dc:creator
               rdf:resource="http://www.example.org/staffid/85740"/>
9.    </rdf:Description>

10. </rdf:RDF>
```

Figure 2.6 RDF/XML Serialization of the graph in Figure 2.5

of the subject resource. Lines 6, 7 and 8 provide property elements, *exterms:creation-date, dc:language and dc:creator*, to represent the predicate and object of the statement. The contents of all these property elements are the objects of the respective statements. The property elements are nested within the containing *rdf:Description* element, indicating that the properties apply to the resource specified in the *rdf:about* attribute of the *rdf:Description* element. Line 9 indicates the end of this particular *rdf:Description* element.

On the same lines, the cycle in the connected graph shown in the previous section in Figure 2.4 can be serialized is shown in Figure 2.7. Even though the RDF graph in Figure 2.4 has arbitrary depth and a cycle, when it is serialized using RDF/XML as shown in Figure 2.7, every RDF resource is just one level deep the *resource* being described at the top level and value of its properties at the leaf level. If this value of the resource happens to be another resource, it appears as a resource at the top level with its properties at the leaf level.

www.manaraa.com

```
<rdf:Description rdf:about = "id123456">
    <foaf:weblog rdf:resource = "http://foobar.xx/blog">
</rdf:Description>
<rdf:Description rdf:about = "http://foobar.xx/blog">
    <rdfs:seeAlso rdf:resource = "http://foobar.xx/blog.rdf">
 </rdf:Description>
<rdf:Description rdf:about = "http://foobar.xx/blog.rdf">
    <foaf:maker rdf:resource = "id123456">
</rdf:Description>
```

Figure 2.7 RDF/XML Serialization of the cycle in Figure 2.4

www.manaraa.com

CHAPTER 3. The D-RDF FrameworkDesign and Implementation

The Dynamic RDF model is a generalization of the existing RDF model in that the Dynamic models consist of composite properties apart from the conventional atomic properties. This generalization is achieved by embedding dynamic properties in the form of queries or path expressions that navigate through the existing static RDF data to extract the required information. RDF models with all the assertive data can be considered as basic building blocks and the D-RDF models are the various structures that can be constructed from these blocks. To generalize, the dynamic properties are not just restricted to queries that can retrieve information but, they can also be computations on the extracted data. These computations may be arithmetic, logical or relational.

The scope of the dynamic properties is dictated by that of the underlying language and in this case the properties are written in the form of XPATH [6], a language standardized by the W3C for navigating through XML documents. Since RDF models are most commonly serialized in the RDF/XML format; XPath turns out to be a good choice for defining the dynamic properties. In this case it has to deal with node/element names which belong to distinct namespaces; this being an inherent feature of RDF. Hence, the XPath expressions are written in terms of the qualified names of the resources and properties.

www.manaraa.com

3.1 XML to RDF/XML Mapping

In order to be able to use the functionality of XPath in the RDF domain, the data needs to be formatted in XML and the fact that RDF models can be serialized in the RDF/XML format made it easy to make the choice. Simply stated, RDF/XML is XML syntax for representing RDF data. Further, a mapping of nodes and elements as they appear in any XML document to the equivalent entities in RDF/XML shows that the use of XPath in RDF is indeed useful and correct. Basically, in order to encode the RDF graph in XML, the nodes and predicates have to be represented in XML terms - element names, attribute names, element contents and attribute values. RDF/XML uses XML Qualified Names (QNames) as defined in Namespaces in XML to represent the RDF URI references.

An RDF graph can be considered as a collection of paths of the form - node, predicate arc, node, predicate arc, node, predicate arc ... node, which cover the entire graph. In RDF/XML these turn into sequences of elements inside elements which alternate between elements for nodes and predicate arcs. This has been called a series of node/arc stripes [10], where the node at the start of the sequence turns into the outermost element; the next predicate arc turns into a child element, and so on.

3.2 XML Path Language

XPATH (XML Path Language) [12] is an expression language for addressing portions of an XML document, and for computing values based on the content of an XML document. The primary purpose of XPath is to address parts of an XML document. In support of this primary purpose, it also provides basic facilities for manipulation of strings, numbers and booleans. XPath gets its name from its use of a path notation for navigating through the hierarchical structure of an XML document. Hence, the mode of operation in XPath is a path expression that can select nodes or node-sets in an XML document. XPath uses a compact,

www.manaraa.com

non-XML syntax to facilitate its usage within URIs and XML attribute values. Apart from the basic XML tree traversal and retrieval, XPath is also useful for its more than 100 built-in functions.

As already understood, XPath models an XML document as a tree of nodes. There are different types of nodes, including element nodes, attribute nodes and text nodes. XPath fully supports XML Namespaces. Thus, the name of a node is modeled as a pair consisting of a local part and a possibly null namespace URI; this is called an expanded-name. A *location path* is the most basic yet, the most important grammatical construct of an XPath expression and is written as a sequence of *steps* to get from one XML node, the current 'context node', to another node or set of nodes. The steps are separated by "/" (i.e. path) characters. Each step has three components: Axis Specifier, Node - Test and Predicate as explained below:

- The Axis Specifier defines the tree-relationship between the selected nodes and the current node
- The Node - Test identifies a node within an axis
- Zero or more predicates to further refine the selected node-set

The syntax for a location step is the axis name and node test separated by a double colon, followed by zero or more expressions each in square brackets as shown: *axisname::node-test[predicate]*. So, a location step that looks like *child::node name[predicate]* will result in all the children of the current node, named *node name* which satisfy the condition expressed by the *predicate*. An XPath expression may contain such location steps separated by a '/ or it may take a form similar to: *node1/node2[@attribute = value]*. This expression will result in all the children of *node1* that have the name *node2* and satisfy the given predicate.

For example, the following would be the XPath expression to determine the *language* in which the webpage being described in Figure 2.5 is written:

www.manaraa.com

//rdf:Description/dc:language[text()]

An XPath expression is always evaluated based on its *context*. The context specifies the node from which the expression should start navigating the document and helps in the resolution of any variable bindings, inbuilt functions or namespaces that are used in the expression. So, when just a node is the expression, it selects all the child nodes of this context node. Further location steps are added to filter out the children based on the predicates. In the case of RDF/XML, whenever a node (resource) is selected, all its properties are accessible within the context. The dynamic properties are based on the existing properties in the base RDF data and hence, when we use XPath expressions as dynamic properties within our D-RDF models, the properties capitalize on the availability of the context.

3.3 Design

To demonstrate the functionality of D-RDF, the framework is designed such that, there exists an input model and an output model. The input model consists of the dynamic properties formulated in terms of XPath expressions and the output model has the actual values obtained as results of evaluating those XPath expressions. Just like any other RDF property is identified by a prefix, which represents the namespace to which the property belongs, the dynamic properties in our framework are identified by the prefix - *dp*. In this way the execution engine can identify that the value of this property is an XPath expression which needs to be processed to get the actual value of the property. So, the input model is the actual D-RDF model where the user can create new resources which are described with the help of dynamic properties.

Further, we introduce the concept of composition of XPath queries in the D-RDF framework. This serves two purposes - first, it allows referencing XPath expressions within other XPath expressions and second, it helps break down long and complex XPath expressions into simpler ones. The significance in the former case is that we can include one or more dynamic

www.manaraa.com

properties within a new dynamic property just like any other XPath step, eliminating the process of rewriting the path expressions every time there is a need. In the latter case, the focus is on substitution. For this purpose we define a *dynamic intermediate step*, identified by the prefix - *dis*. These path expressions are named dynamic intermediate steps because they appear as steps within a dynamic property and the main purpose of introducing these steps is to avoid redundancy.

For example, if an XPath expression appears in a model other than the model it is querying, then it has to begin with the location of that model as that would be the context for the path expression. This location may be specified as a URL if the model is accessible on the Web or it could be a path to the file on the local drive, both of which can be fairly long. In either case, the context needs to be set and if we assign the path as a value for a *dis* property, thereafter, we just need to include this property in the dynamic property. This is illustrated with the help of the following expressions.

```
<dp:user_quote> fn:doc("file:///C:/RDF_Files/BestPackage1.rdf")//
                rdf:Description/travel:user_quote
</dp:user_quote>

<dis:f2>fn:doc("file:///C:/RDF_Files/BestPackage1.rdf")</dis:f2>

<dp:user_quote>dis:f2//rdf:Description/travel:user_quote</dp:user_quote>
```

Figure 3.1 Usage of the *dynamic intermediate step*

As we can see from the expressions in Figure 3.1, both values of *dp:user_quote* are equivalent but, the second expression is simpler to read and understand due to the use of *dis:f2*. Moreover, *dis:f2* can be reused any number of times.

www.manaraa.com

3.3.1 Design Goals

To summarize, the D-RDF framework has been designed to achieve the following goals. These goals reflect how the D-RDF framework can be put to use to achieve the desired result. A mapping of these design goals to the respective use-cases that we implemented is covered in the following chapter.

1. Ability to perform *Data Analysis* by allowing mathematical, logical or relational computations on the base data.

2. Creation of D-RDF models by *projection* of required data from the base data and *merging* all the projected elements into a new model.

3. *Construction of new resources* whose properties are derived from one or more properties that exist in the base data.

4. *Support for user-defined functions* to create powerful dynamic properties that can discover complex relations in the base data.

5. *Nesting of properties* to help break down complex expressions into simpler ones.

6. *Reusability* of the dynamic properties once the underlying properties are standardized.

7. *Preserving the RDF Format* in the dynamic models so that they can further be used as base data for other dynamic models. In this way, D-RDF leverages the availability of RDF parsers and processing tools.

3.4 Implementation

In this section the technologies used in our implementation of the D-RDF framework are described followed by the execution semantics. We use Jena [13] as the API for creating and manipulating the RDF models and Saxon [15] is the engine required to process the XPath

www.manaraa.com

expressions which make the dynamic properties. The section on execution semantics describes the steps that are taken to process the dynamic properties present in the D-RDF model.

3.4.1 JENA

Jena is a Java API which can be used to create and manipulate RDF graphs as per the RDF specification. In Jena, a graph is called a model and is represented by the Model interface. Jena contains interfaces for representing models, resources, properties, literals, statements and all the other key concepts of RDF and supports both statement-centric and resource-centric approaches to create the RDF models [14]. This implies that models may be created either by creating resources and adding properties and values to the resources or by adding statements of the form - *subject, predicate, object.* For example, the code to create a simple graph, or model, which states that the full name of a person represented by *"http://somewhere/JohnSmith"* is *John Smith* is shown in Figure 3.2:

```
// some definitions
static String personURI = "http://somewhere/JohnSmith";
static String fullName  = "John Smith";

// create an empty Model
Model model = ModelFactory.createDefaultModel();

// create the resource
Resource johnSmith = model.createResource(personURI);

// add the property to the resource
 johnSmith.addProperty(VCARD.FN, fullName);

// add resource and property to the model
 model.addStatement(johnSmith, VCARD.FN, fullName);
```

Figure 3.2 Sample JENA Code [14]

An RDF Model is represented as a set of statements and Jena has an interface called Statement which provides accessor and modifier methods to the subject, predicate and object

www.manaraa.com

of a statement. Jena also has methods for reading and writing RDF as RDF/XML. These can be used to save an RDF model to a file and refer to it whenever required. We use Jena as the base framework to create and process the D-RDF models used in our example use-cases.

3.4.2 SAXON

Saxon [15], an open source implementation of XPath 1.0 and 2.0 among others like XQuery and XSLT, has been used to execute the dynamic RDF properties. The XPath processor is accessible to applications via supplied APIs. It is fully conformant to the W3C standardized specification of XPath. Being open source, we could add our own functionality of interpreting and executing, what we call the dynamic properties, which could either be just simple XPath expressions or composite XPath expressions, one being a location step within another.

3.4.3 Execution Semantics

The processing of the D-RDF model is completed in two passes. In the first pass, all the dynamic properties (dp) and the dynamic intermediate steps (dis) are identified and their values (actual path expressions) saved in a hash table. Once, the entire file is traversed and all such dynamic elements are recognized, it is time to execute them. The second pass deals with this process. In the second pass, each of the dynamic properties is analyzed and executed. First, the object value of the property, which is actually an expression, is broken down into tokens. Each location step is a token. After this, each of these tokens is checked to determine if any of them has a prefix: *dp*. This indicates that there is a nested dynamic property within the current one. If found, the value of this *dp* is retrieved from the hash table created during the first pass to be substituted in the current *dp* and the same procedure is repeated with the new *dp*. If no more such dynamic properties are found, the expanded XPath expression is executed to yield the value of the dynamic property.

The reason for implementing the D-RDF framework with the help of two models - input

www.manaraa.com

and output, is to be able to accommodate any changes that may occur in the base RDF data models. Keeping in mind the volatile nature of some data sets, it is in the best interest of the users of this framework, to work with two models - one that has the dynamic properties and one that has the evaluated answers. The answers would change with any change in the queried RDF models and as a result, no data will be obsolete or incomplete.

Following is the algorithmic representation of the two-pass execution of the D-RDF model:

1. First Pass
    (a) Get an iterator over all the statements of the input D-RDF model.
    (b) For every statement until the end of the model is reached, if the prefix of the predicate is *dp or dis*
        i. Add the value of the property (XPath expression) to a Hash table based on the qualified name of the property as the key.
2. Second Pass
    (a) Get an iterator over all the statements of the input D-RDF model.
    (b) For every statement until the end of the model is reached, if the prefix of the predicate is *dp*, then
        i. Retrieve the corresponding XPath expression from the hash table
        ii. Tokenize the expression in terms of location steps
        iii. Initialize a string that will hold the expanded expression
        iv. For every token, if the prefix is *dp or dis*
            A. Repeat steps i) to iii) until an expression without any dynamic property is obtained
            B. Concatenate the expression to the final string

www.manaraa.com

v. else, concatenate the location step to the final string

vi. when the end of the token list is reached, execute the expression held in the string

vii. add the statement with the evaluated value to the output model

(c) If the prefix of the predicate is *dis*, do not add it to the output model

(d) If the prefix is neither *dp* nor *dis*, add the statement as is to the output model

3.5 VALIDATION

To confirm the correctness of the D-RDF models, all the models from the use-cases covered in the following chapters, were validated using an online RDF validation service, hosted by the W3C. This RDF validator [16] is based on Another RDF Parser (ARP) and supports the Last Call Working Draft specifications issued by the RDF Core Working Group, including data types. If RDF-compliant, the tool provides different display options for the serialization that is being tested. One could view the RDF/XML file in the form of RDF Triples or RDF Graph or both. Also, different image formats are supported to display the graphs.

www.manaraa.com

CHAPTER 4. Discussion of Implemented Use-Cases

Having described the design and implementation of the D-RDF framework in the previous chapters, we now describe the use-cases that we implemented to justify the significance of this concept. RDF by itself is a technology that can be applied to virtually any data domain. On the same lines, with the help of a few use-cases that we implemented, we have tried to demonstrate that D-RDF models can also be built from different data domains - online shopping data, travel package descriptions and research publications to name a few.

The use-cases described in this document not only differ in terms of the underlying data domain they operate on, but also by the level of complexity with respect to the dynamic properties that describe the resources in these models. The simplest of these involves constructing a new RDF model from existing models by grouping resources that match the criteria specified by the dynamic properties. A more complex example involves demonstration of how the XPath dynamic properties can determine complex transitive relations between resources such as research publications.

While describing the use-cases in this section, an attempt has been made to compare them with some of the use-cases and requirements described in the W3C Working Draft called RDF Data Access Use Case and Requirements [17]. The W3C draft specifies use cases, requirements, and objectives for an RDF query language and data access protocol. It suggests how an RDF query language and data access protocol could be used in the construction of novel, useful Semantic Web applications in areas like web publishing, personal information management, transportation, tourism and many more domains.

www.manaraa.com

Broadly speaking, these use-cases help us understand how different types of D-RDF models may be created from the base RDF models. In other words, the design goals of D-RDF as mentioned below are reflected in the use-cases.

- *Data Analysis*: D-RDF models may be created for performing mathematical analysis on the base RDF data. A simple use-case of this type would be use-case 4.1 - Analysis of Research Publications. Managing a stock portfolio would also fit in this category.

- *Merge resources from different RDF models*: The D-RDF models may be created by merging resources from base RDF models. By merging we imply that the resources that match the specified criteria are included in the D-RDF model without any change. Use-case 4.2 - Minimum Priced Computer, falls in this category.

- *Construct new resources*: Users may also construct their own resources by defining new properties based on existing ones. Use-case 4.3 - Construct a Travel Package fits in this category.

- *Nested properties*: The ability to use the result of one dynamic property in computing the result of others turns out to be very powerful in the D-RDF design, especially when a new resource is being constructed from existing ones. Use-case 4.3, which describes the construction of a highly customized travel package, is one where nested properties are put to use extensively.

- *Reuse*: Nested properties enables reuse locally. Once, the RDF vocabularies are standardized the dynamic properties based on them could also be standardized and added to vocabularies which could then be used globally.

- *Support for User-defined Functions*: D-RDF models could also be created by including user-defined XPath functions in the dynamic properties. The feature for finding the transitive relations is one such example.

www.manaraa.com

4.1 Analysis of Research Publications

This is a simple use-case to demonstrate the power of D-RDF. Every department in every university has a database of publications representing years worth of research. If all the information about publications is stored in RDF format, very fine granularity can be added to the description of each publication, in the form of RDF properties. In this direction, the Dublin Core Metadata Initiative introduced the Dublin Core vocabulary of terms and properties to be used to describe publications of any form.

In this simplified representation of the publication data, each publication is a resource like *http://ecpe.univ.edu/ProfPubs/pub3*. Each of these resources is described with the help of properties like *dc:title* which is a part of the Dublin Core vocabulary and *pub:status* to indicate whether the publication was *accepted or rejected*. Note that the property *status* is not a part of the Dublin Core vocabulary. It was introduced for the purpose of this use-case. One of the advantages of using RDF is that the schema or vocabulary for any domain is extensible in this manner, literally by anyone who wants to use them.

Let us consider a scenario where the management of a department decides to keep track of the number of publications accepted and rejected and generate useful statistics based on such data. This could be accomplished by writing a dynamic RDF property to count all the resources in a given file where the value of the property named *status* is *accepted or rejected*. Figure 4.1 shows the base publication data and Figure 4.2 shows the dynamic property to be embedded in the base data.

The result of this dynamic property is depicted in Figure 4.3. The execution engine replaces the namespace *dp* by *dpResult*, keeping the name of the property as is in the output model and the value of this property is the result obtained by processing the XPath expression.

The decision to use XPath expressions as dynamic properties proved to be very useful. In this case, the use of just one inbuilt function, *count*, proved helpful to come up with the

www.manaraa.com

```
<rdf:Description
  rdf:about="http://ecpe.univ.edu/ProfPubs/pub3">
    <pub:status>Rejected</pub:status>
    <dc:title>Publication3</dc:title>
 </rdf:Description>
 <rdf:Description
   rdf:about="http://ecpe.univ.edu/ProfPubs/pub4">
    <pub:status>Accepted</pub:status>
    <dc:title>Publication4</dc:title>
 </rdf:Description>
```

Figure 4.1 Base RDF publication data

```
<rdf:Description
   rdf:about="http://ecpe.univ.edu/ProfPubs/Publications">
   <dp:numAccepted>
        fn:count(//rdf:description / pub:status[text() = 'Accepted'])
   </dp:numAccepted>
</rdf:Description>
```

Figure 4.2 Input D-RDF model with dynamic property for Use-Case 4.1

```
<rdf:Description
  rdf:about="http://ecpe.univ.edu/ProfPubs/pub3">
    <pub:status>Rejected</pub:status>
    <dc:title>Publication3</dc:title>
 </rdf:Description>
 <rdf:Description
   rdf:about="http://ecpe.univ.edu/ProfPubs/pub4">
    <pub:status>Accepted</pub:status>
    <dc:title>Publication4</dc:title>
 </rdf:Description>
 <rdf:Description
   rdf:about="http://ecpe.univ.edu/ProfPubs/Publications">
   <dpResult:numAccepted> 1 </dpResult:numAccepted>
 </rdf:Description>
```

Figure 4.3 Output D-RDF model for Use-Case 4.1

www.manaraa.com

total number of resources that matched all the criteria expressed in the dynamic properties. Likewise, with the help of over a hundred built-in functions that may be used for string, numeric or node manipulation, the dynamic properties can be used to perform operations on the data and analyze it from various perspectives.

4.2 Minimum Priced Computer

The goal of this use case is to create a D-RDF model consisting of the minimum priced components available from different online shopping portals to build a customized computer system.

The base data set for this model consisted of RDF files, one for each type of component being described, for e.g., monitors and processors, with properties like price and a link to their web page, from where it could be ordered. The D-RDF model has dynamic properties that determine the minimum priced component of each type. The *min* function of XPath language is applied. The arguments to this function are the values of the RDF property - *price*, for each component. In this use-case, each component is an RDF resource. This helps select the cheapest available component and for each selected component, the resource and its description are a part of the output D-RDF model. By description we mean the properties and their values for a given resource. If desired, the user can choose to include just a few required properties in their D-RDF model. For example, in this case, the Web link of the desired component would be most useful as the user can follow the link and order the component online.

Figure 4.4, demonstrates that resources that satisfy required criteria can be projected from the base RDF data models and added to custom designed D-RDF model.

More precisely, the Figure 4.5 depicts how the D-RDF file *MinPricedComputer.rdf* gathers information from other D-RDF files that contain the details like links and prices of the cheapest components gathered from two or more base RDF data files. This is a good example of how

www.manaraa.com

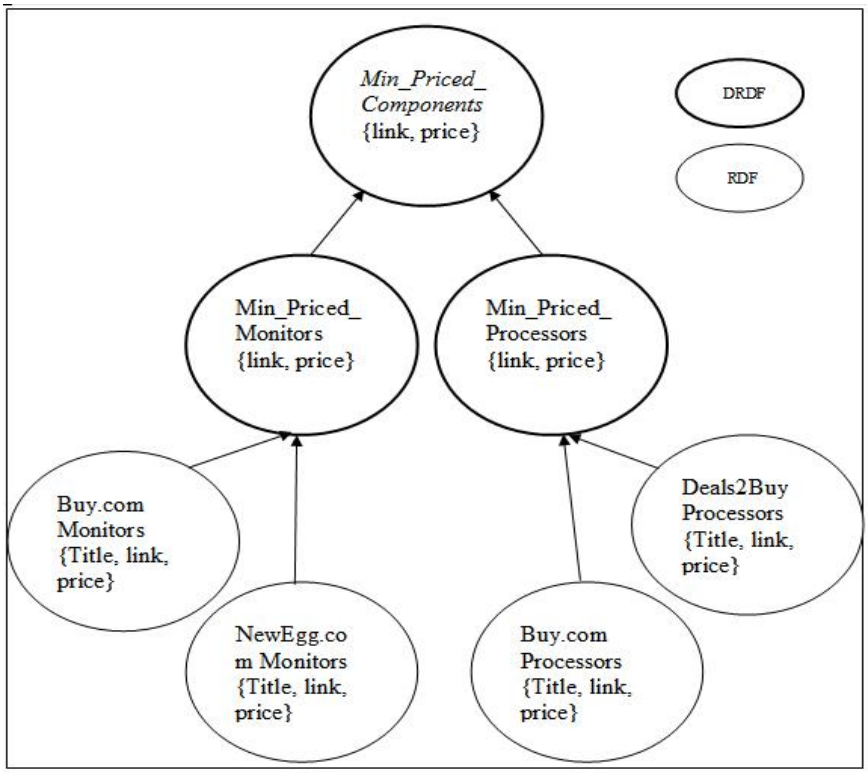

Figure 4.4 Minimum Priced Computer

www.manaraa.com

these information models form an intertwined hierarchy.

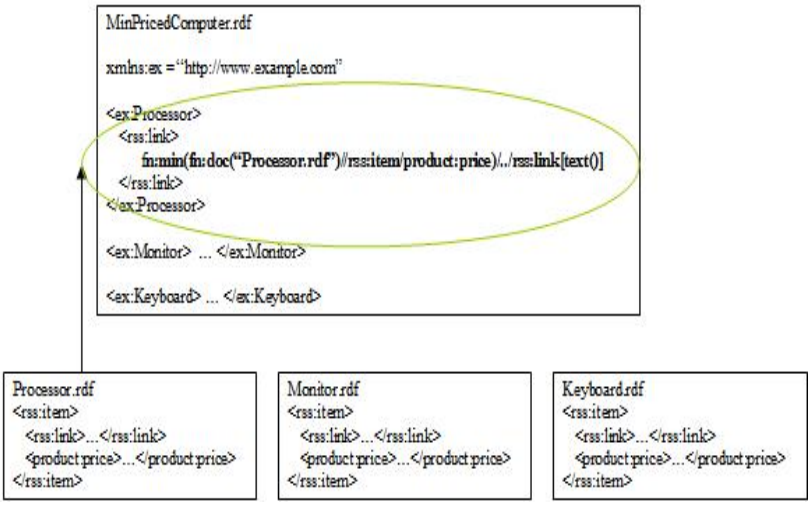

Figure 4.5 Project and Merge

The purpose of this use-case is similar to that of the use-case described in section 2.2 of the RDF Data Access Use Case and Requirements document [17] which deals with finding information about Motorcycle parts and is categorized under the Supply Chain Management domain. In this use case, a user queries an RDF database of motorcycle parts for a particular component and gets all the information - the resource which represents the component, and its properties. Information is also provided about the resources which appear as objects in the triples obtained for the searched component. In short, a sub graph of the original graph is returned. Hence, it can be concluded that our system satisfies the Subgraph Results requirement which states that 'It must be possible for query results to be returned as a subgraph of the original queried graph.' In other words, the dynamic properties are capable of projecting resources from different base models and merging them together to create a new model that serves a particular purpose.

www.manaraa.com

4.3 Construct a Travel Package

Going a step further, in this use-case we introduce the concept of using dynamic properties to satisfy user preferences among other criteria. The user preferences are expressed as RDF data in the D-RDF model and new resources are constructed and described with the help of dynamic properties that are defined in terms of the user's preferences.

The scenario for this use case is such that a user, say Alice, needs to travel to a location say, San Francisco. During her visit, she would need to rent a car and stay in a hotel. The base RDF data that is available to Alice includes information about flights to the actual destination and to nearby airports, information about rental car charges at those airports and hotel charges for a few hotels at the destination. The D-RDF model helps Alice plan her trip by allowing her to set her preferences and create resources whose dynamic properties are framed to help determine the most satisfactory combination of flight, car and hotel in terms of budget.

Among her preferences, budget is of prime importance followed by the length of the visit in terms of number of days and then comes the rating of the hotel she would like to stay in. Instead of browsing through each travel web site and collecting information manually, she can just choose to create her own packages, represented by resources in the D-RDF model with dynamic properties to help pick the best option each for flight, car and hotel. For instance, there is a resource that is described by properties that search for the cheapest flight to the actual destination, pick the cheapest car rental at that destination, calculate the total money required for these two taking into account the *number of days* preference for the car rental and finally pick the best hotel such that the total price for that would be less than or equal to the difference of the actual budget and the amount calculated for flight and hotel together. Just to confirm that this is the best hotel she can afford, she can create another resource that has properties to select the cheapest car rental, determine the airport at which this deal is available, cheapest flight to that particular airport and hotel in the actual destination with

www.manaraa.com

the remaining money from the budget. It turns out that cheaper car rental from a non-major airport saves her enough money to reserve a hotel with a higher rating in San Francisco.

```
<rdf:Description
  rdf:about="http://dynamic_property/travel/User_Preferences">
    <travel:user_budget>900</travel:user_budget>
    <travel:destination>SFO</travel:destination>
    <travel:alt_destination>OAK</travel:alt_destination>
    <travel:no_of_days>4</travel:no_of_days>
    <travel:hotel_rating>2</travel:hotel_rating>
</rdf:Description>
```

Figure 4.6 Resource describing *User Preferences*in the input D-RDF file

Figure 4.7 depicts the resource that is created with the help of dynamic properties. As mentioned in the Construct new Resources design goal, this use-case demonstrates how a D-RDF model can be composed of newly created resources which do not exist in any base RDF models. Only the data that is needed to define these resources (in terms of properties) exists in the base RDF models and is fetched by the XPath expressions that make up the dynamic properties of these resources. In this use-case nesting of the dynamic properties has also been used extensively. The dependency of one dynamic property on another explains the need for nesting of these properties in the D-RDF model.

4.4 Transitive Closure - An Experimental Feature

As an experimental yet useful feature, we have implemented a function to find the transitive closure of a given relation. The function, which takes two arguments - the relation and the context node, is meant to be a part of the XPath functions library so that it can be used in dynamic properties. Primarily, the function is used to find the transitive closure of all publications that reference a given publication. In other words, the relation is 'refer to' and the context node is the publication for which the closure needs to be calculated. To start with,

www.manaraa.com

```
<BudgetFlightDeal>

      <CheapFlight>pick out the cheapest flight resource</CheapFlight>

      <MatchCar>pick out all car rental resources at the destination to
       which the cheapest flight was obtained in dis:CheapFlight
      </MatchCar>

      <CarFare>calculate the car rental fare for the user specified
      number of days</CarFare>

      <MinFlightCarFare> sum of the flight and car rental fares
      </MinFlightCarFare>

      <HotelBudget>difference of user's budget and MinFlightCarFare
      </HotelBudget>

      <BestHotel>hotel with fare less than or equal to HotelBudget
      </BestHotel>

</BudgetFlightDeal>


On the same lines, the <BudgetCarDeal> would start with finding the
cheapest car rental and its location followed by the cheapest flight to
that location and then the best hotel in the same way as
<BudgetFlightDeal>
```

Figure 4.7 Resource with *dynamic properties* to select best package

this would determine all the publications that refer to the given publication and thereafter, do the same for each of the resulting publications. This would work best for fairly recent publications, as the computation would not have to run for an arbitrarily long time which might be the case otherwise.

Support for user-defined functions to enable complex queries in the D-RDF design gives the users the power to utilize this framework to the maximum possible extent. With this feature, the users are not limited in terms of their usage of the system. The functions created can further be reused any number of times.

4.5 Significance of D-RDF

As mentioned in the introduction to RDF in Section 1.1, Semantic Web is essentially the web of data. Analogous to the WWW where one document or page may link to another and

www.manaraa.com

that to another and so on (actually creating the Web) through hyper links, the vision for Semantic Web is that all data resources should be linked together. In some sense, this can be seen as rendering higher level of granularity as compared to providing links between the documents that contain the data. For this to happen, URIs or the global identifiers must be put to use extensively so that finally, every resource is identified by one URI and that is known and available for use by everybody.

Once the data model is agreed upon, the next step is to figure out how to retrieve meaningful and useful information from the existing data. Use of query languages is the universally accepted solution and more recently, with XML coming into existence, path traversal languages are catching up with the query languages. The D-RDF model is our answer to the challenge of retrieving required information from the web of data. In D-RDF the query is a dynamic property which becomes a part of the RDF model and derives the required information. The D-RDF design is such that if the information required is available within a single RDF model then, the dynamic property may be included in the same model. On the other hand, if the information required needs to be derived by correlating data from more than one RDF models then, a new dynamic RDF model may be created to achieve the goal. This can be thought of as a program in itself that performs the required operations and outputs the result. Moreover, the result is an RDF model by design and hence can be further queried. These design decisions are reflected in use-cases 4.1 and 4.2 respectively.

Once the vocabularies that users around the world employ to describe the data they own are standardized, the dynamic properties created by one user can be reused by all others time and again. The Dublin Core Metadata Initiative is a promising example. If all institutions were to use the properties and classes defined by Dublin Core to describe their publications it would only make sense to create a library or vocabulary of dynamic properties such as *dp:numAccepted*, as defined in use-case 4.1, and others, like number of publications accepted in a given year or rate of acceptance in a given year. Users may use them as is or modify them

www.manaraa.com

for their purposes; create new ones that do not exist and extend the vocabulary just like base RDF vocabularies.

Having these programs take the form of RDF has the advantage that the existing RDF processing engines can be used to parse them by just adding the dynamic property processing engine. So not only do they help us process the RDF data but also, the computations on them.

The use-cases described here, are but a few applications of the powerful D-RDF model. As mentioned earlier, these can be used in any data domain to enable users to customize the information according to their needs.

www.manaraa.com

CHAPTER 5. Related Work and Motivation

To better understand the significance of RDF, this section starts with the comparison of the Resource Description Framework with its close relatives XML (Extensible Markup Language, RSS (Really Simple Syndication) and RDB (Relational Database). The motivation for introducing D-RDF follows with sub-section dedicated to comparison of D-RDF with the latest Yahoo Pipes. Finally, moving on to the use of path language in D-RDF, some of the existing work in this field is described.

5.1 RDF and Related Technologies

The following sections focus on understanding the significance of RDF as compared to similar technologies that exist today.

5.1.1 RDF and XML

RDF/XML being the normative syntax for serialization of RDF models is one of the reasons for the concern *How is RDF different from XML?* To start with, XML has a tree structure and RDF has a graph structure. As a result, an XML element is hierarchical and goes more than one level deep. Even though an RDF graph can have arbitrary depth like an XML tree, when it is serialized using RDF/XML, an RDF resource is just one level deep the *resource* being described at the top level and value of its properties at the leaf level. If this value of the resource happens to be another resource that can be described, it appears as a resource at the top level with its properties at the leaf level. This was depicted in Figure 2.7. Moreover,

www.manaraa.com

XML can be used in different ways to encode the same information but, the same RDF graph results from different XML trees [18]. For instance, the statement, "The author of the *webpage* is Alice," which is represented in RDF as shown in Figure 5.1, can be represented in more than one XML forms as shown in Figure 5.2.

```
<rdf: Description  rdf: about = "webpage">
      <dc: author>Alice</dc: author>
</rdf: Description>
```

Figure 5.1 Single RDF Representation

(i)

```
<document href="webpage">
  <author>Alice</author>
</document>
```

(ii)

```
<document>
  <details>
   <uri>href="webpage"</uri>
   <author>
     <name>Alice</name>
   </author>
   </details>
</document>
```

Figure 5.2 Multiple XML Representations

While XML is a text format, RDF is a data model. XML adds structure to the voluminous data available today but, it must be noted that its design makes it more suitable for communication at the message level. On the other hand, RDF adds meaning to the voluminous data apart from adding the structure. The basic unit of information in RDF is a triple and every member of the triple is identifiable by a globally unique identifier. This facilitates representing information meaningfully and correlation of data from a variety of sources. It is important to note that using a URI in RDF is fundamentally different from using unique identifiers in XML because the uniqueness of the former is ensured globally, whereas that of the latter is only guaranteed within a document. The document-centric view of XML makes it difficult to

www.manaraa.com

refer an embedded entity outside the XML document.

In essence, RDF is a declarative language and provides a standard way for using XML to represent metadata in the form of statements about properties and relationships of items on the Web.

5.1.2 RDF and RSS

RSS [19], from its inception as Rich Site Summary to its current status of Really Simple Syndication is a lightweight XML vocabulary for describing metadata about Web content and is ideal for news syndication. Originated by UserLand Software in 1997 and used by Netscape to populate Netscape's My Netscape portal with external news feeds ("channels") RSS has become perhaps the most popular XML format today for gathering and distributing news. Among all its versions RSS 1.0 was based on RDF [20]. The current version in use is RSS 2.0.

The main difference between RDF and RSS arises from the underlying purpose of the two technologies. RSS was designed for syndication of data in the form of feeds and making it available for use by readers. Instead of digging out the crux of matter from verbose Web sites users could now subscribe to RSS feeds and get the gist of what was happening in the world around them. On the other hand, the goal of RDF was to restructure the entire Web, not limited to the information that is of interest to a specific audience as in the case of RSS but, all the information that exists, from any source or domain, and that can be made public.

Although the syntax of the two technologies is similar yet, RSS is limited in terms of the amount of data it can represent in terms of the number of items described in each feed. RSS is also limited in its expressivity as compared to RDF. For every *channel* or *feed* which is a collection of items and their descriptions, there is a set of required elements title, link and description [19]. Apart from these there are other optional elements language, pubDate, image, etc. that could be added to the item description if needed. These issues do not surface in the case of RDF as there is no restriction on the property elements that can be used in an

www.manaraa.com

RDF model. The community is encouraged to create useful vocabularies for every domain and reuse the elements that are already defined in those vocabularies across domains to enforce interoperability among the data models. Furthermore, there is no notion of *items* in the RDF model but, the set of triples describing each resource could be grouped into a single record. It is a description model and hence, there is no limit on the size of the RDF model in terms of the number of records.

5.1.3 RDF and RDB

The semantic web data model can be considered analogous to the relational database model if we consider the mapping of an RDF node to a DB record, the RDF property name to the DB field (column) name and the RDF object value to the value of the DB record for that particular field [21] as shown in Table 5.1. But again, it is to be noted that while an individual cell or field in an RDB is not often thought of on its own, the corresponding RDF property is a resource in itself.

Table 5.1 Mapping RDF Triples to a table in a database

Resource id	**foaf:name**	**foaf:knows**
http://example.com/person1	Alice	http://example.com/person15
http://example.com/person3	Bob	
http://example.com/person8		http://example.com/person4

In the relational model, a row in a table is actually an assertion that the relation is true for the values in the row. A SELECT query is a filter on the assertions that are true for the given conditions. Another significant difference between relational databases and RDF is that in the former, for a certain set of values a relation is either considered true - if there is a corresponding row in the table, or false - if there isn't. In the RDF model, in the general case, if a set of values isn't in the "row," i.e. a particular statement is not present, then it's not false; it's just unknown.

www.manaraa.com

With the directed graph structure of RDF there's an obvious analogy to the interlinked structure of the Web. But, the most important point of RDF in regards to the Web is that the subject, predicate and object can be resources in the Web sense, things identified with URIs. This means that they can act as ID fields/keys not just in the local store but anywhere they appear. In other words, from the relational perspective the (Semantic) Web as a whole can be considered a single database. In this view an individual RDF store or file is just a cache of a little bit of the data in the Semantic Web.

5.2 How Dynamic-RDF Fits in the Semantic Web Architecture

In a keynote session at XML 2000 Tim Berners-Lee, Director of the World Wide Web Consortium outlined his vision for the Semantic Web [22]. Berners-Lee said that in the context of the Semantic Web, the word "semantic" meant "machine processable." He explicitly ruled out the sense of natural language semantics. RDF is the basis of Semantic Web and D-RDF is one mechanism through which Semantic Web can become a success. The Resource Description Framework was created to achieve certain goals in the field of data representation and exchange and the introduction of dynamic properties in RDF helps achieve many of those goals.

To start with, the design of RDF was motivated by the idea of inter-working between applications: combining data from several applications to deduce new information. The dynamic properties are designed to achieve this goal in its entirety. If the underlying data is modeled in RDF and the source of such data is known, creating D-RDF models can help gather required information from different sources in one model. Thus, we shift away from today's provider-centric model to a user-interest-centric model where the users can not only subscribe to the available information but also restructure the data at their disposal.

Inference is one of the driving principles of the Semantic Web, because it allows the creation of Semantic Web applications quite easily. The principle of "inference" is one of being able

www.manaraa.com

to derive new data from data that is already known. In a mathematical sense, querying is a form of inference (being able to infer some search results from a mass of data, for example). We claim that D-RDF is a means of achieving useful inference from the existing RDF models because the dynamic properties, in the form of path expressions, basically retrieve required information and add it to the D-RDF model.

5.2.1 The Semantic Web Architecture

Figure 5.3 depicts the layered architecture of Semantic Web [23] starting with the foundation of URIs and Unicode. RDF URIs can contain Unicode characters. The Unicode Standard is the universal character encoding standard for written characters and text. It defines a consistent way of encoding multilingual text that enables the exchange of text data internationally and creates the foundation for global software. Above that is the syntactic interoperability layer in the form of XML and that forms the basis of the data interoperability layer represented by RDF and RDF schemas. These layers sum up most of the Semantic Web that is currently available in implementation form. Digital Signature runs right up the side of the stack emphasizing its widespread utility. On top of RDF lie ontologies, which allow the further description of objects and their inter relations. An ontology is capable of describing relationships between types of things, such as "this is a transitive property", but does not convey any information about how to use those relationships computationally.

The logic layer is envisioned to provide an interoperable language for describing the sets of deductions one can make from a collection of data and derive new facts about it. Dynamic RDF fits perfectly in the logic layer. The proof language will provide a way of describing the steps taken to reach a conclusion from the facts.

The Semantic Web vision is that once all these layers are in place, there will exist a system in which we can place trust that the data we are seeing, the deductions we are making, and the claims we are receiving have some value. That's the goal: to make a user's life easier by

www.manaraa.com

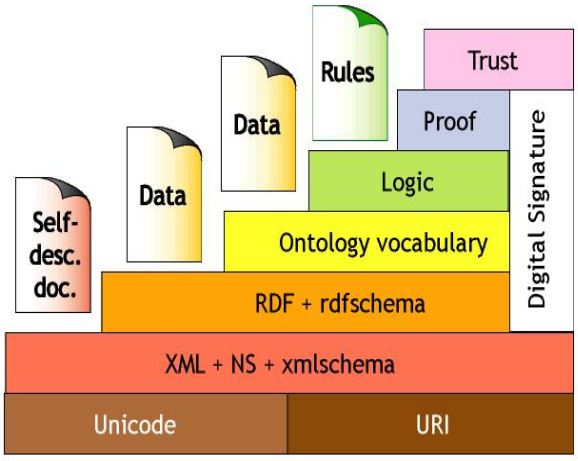

Figure 5.3 The Semantic Web Architecture

the aggregation and creation of new, trusted information over the Web.

5.2.2 D-RDF and Yahoo Pipes

Pipes from Yahoo Inc. [24] is an interactive composition tool that enables the aggregation, manipulation and mash up of data feeds, like RSS, from around the Web. From the high level description, the functionality of Pipes and D-RDF sounds similar - both are aimed at giving users the power of customizing the data at their disposal and see only what they want to see. Although, from users' perspective, Pipes has a very easy to use interface with a number of modules and operators at their disposal yet, the graphical user interface limits the expressivity of the tool. D-RDF on the other hand, is more flexible and extensible as users write their own dynamic properties and are not limited by what the GUI has to offer.

Another architectural difference between Pipes and D-RDF is that Pipes is limited by the domain in which it can be used - the Web. It is a web-based tool and all the source data that it operates on has to be on the Web, identifiable and retrievable by the respective URLs. Initially, limited only to the typical format of Web documents like Yahoo or Google Search

www.manaraa.com

results, more recently, Pipes has included support for XML and CSV documents available on the Web. D-RDF is capable of processing RDF data maintained on the web (URL) or in a file on the local system. What this implies is that if users manage their data with the help of RDF models, D-RDF can be used to infer useful information from all these models. Whereas Pipes runs on the Yahoo servers and not on the clients machine, D-RDF is not server dependent.

D-RDF uses a path expression language; it can be used in principle, to work with huge databases; Pipes are designed to work on the web, where data available from a source is assumed to be limited in size. RDF and hence, D-RDF, could also be a database technology, but Pipes is not.

Pipes is evolving and one of the recently added features is the ability to use a pipe as an intermediate module in another pipe. Our system achieves similar functionality by allowing the composition of dynamic properties - using a previously defined property as a step in a new property.

5.3 Path Traversal Languages

The fact that the RDF/XML serialization format for RDF inherits its features from the comparatively older and matured XML is the motivating factor for employing path languages which are inspired from XPath to query the RDF models, for its powerful navigation capabilities that help find information in XML documents. Stated more precisely, RDF Path is to RDF/XML what XPath is to XML.

What began as a concept to transform RDF triples or graphs into other forms, typically a web interface in HTML, is slowly moving towards being applied as an intuitive and easy to learn query language for RDF models. There have been many initiatives in this direction but, only a handful succeeded in developing a full-fledged specification for a practical RDF Path language.

www.manaraa.com

Versa [25] is an evolving, graph-based language implemented in Python for querying RDF models and its design was motivated primarily by XPath. Its main features include traversal of arcs, processing of node contents and general expression evaluation. Like XPath, Versa also provides facilities - such as in-built functions, Boolean logic and set operations, aggregates, substring matching, and other core data type manipulation. Additionally, Versa expressions are evaluated with regard to a context in the associated RDF model.

A more direct, but not fully matured, implementation of an RDF Path language is done by Damian Steer in his project named TreeHugger [26]. In his own words, TreeHugger is an attempt to use XPath over RDF/XML and is implemented as a Saxon extension function. Saxon package is a collection of tools for processing XML documents and has a built-in XML parser. Being an implementation of the most recent standards of XPath 2.0, XSLT 2.0 and XQuery 1.0, it enables the compilation of XPath expressions in Java code.

Our implementation of a path traversal language for RDF is similar to the idea initiated in the TreeHugger project but, we also added the capability of parsing nested path expressions. This helped decomposing complex path expressions into simpler, more intuitive ones. There are many reasons behind choosing a path traversal language to query the RDF models, context-specific, intuitive to write complex expressions, standardized for a good number of years now, to name a few.

Though there exist standards for RDF query languages, these languages are fairly new and research is ongoing to make them mature. Currently SPARQL is the W3C candidate for being a standard query language for RDF. It has SQL-like constructs and a well-defined grammar but, it does not support any aggregate functions as of yet. Research in the field of RDF is advancing with time and soon the query languages will be standardized. The good news is that the D-RDF framework is extensible and in future properties may not only be written in XPath but also in the current standard query language, which ever is most suitable and easy to use.

www.manaraa.com

CHAPTER 6. Contribution and Future Work

To summarize the work done towards the completion of this thesis, in this chapter we present the contributions at this point in time and suggest some ideas to improve the design in the future.

6.1 Key Contributions

The concept of dynamic RDF properties has resulted in a flexible and scalable computation model for RDF data. With D-RDF, the data not only carries semantics in terms of static properties described by URIs, but also contains computing methods on the data in the form of dynamic properties, the value of which changes dynamically with any change in the underlying data and with the needs of the requesting party.

An insight into putting D-RDF to use across varied domains is presented with the help of the different use-cases in Chapter 4.

6.2 Challenges and Limitations

One of the major challenges faced while working on D-RDF was the unavailability of base RDF databases. All of the RDF files were created based on sample data from the Web.

Some of the standards and technologies used in the implementation are still evolving in terms of optimization and feature set. This affects the efficiency and performance of the D-RDF framework. For example, the program that Jena [13] uses to serialize the RDF models

www.manaraa.com

as RDF/XML, may not perform well for large models.

6.3 Future Work

A few directions that this work could take in the future are identified in this section.

Development of D-RDF vocabularies which define the dynamic properties that could be reused by everybody will be achievable when the underlying RDF vocabularies are standardized and used by the community consistently.

The system can also be extended by allowing dynamic properties to be written in other RDF query languages like SPARQL as it continues to evolve and accommodate more features than what it has today.

Investigation of how the RDF vocabulary for Composite Capability/Preference Profiles (CC/PP) [29] can be used in D-RDF will open new avenues to support dynamic binding of information with multiple device types, based on their capabilities and preferences.

As the number and variety of devices connected to the Internet grows, there is a corresponding increase in the need to deliver content that is tailored to the capabilities of different devices. The CC/PP framework helps create profiles that describe the user preferences and capabilities of the device one is using to request the required information.

Including such device-specific semantics in the data can be useful in customizing the data to the needs of the various pervasive computing devices. This would eliminate the need to store device-specific data for each available device. Instead, the same data will be dynamically adapted to fit the needs of many devices.

www.manaraa.com

BIBLIOGRAPHY

[1] "RDF Primer," World Wide Web Consortium, 2004. Available at *http://www.w3.org/TR/rdf-primer/*, Date accessed: October 29, 2007.

[2] "W3C Semantic Web Activity," World Wide Web Consortium, 2001. Available at *http://www.w3.org/2001/sw/*, Date accessed: October 29, 2007.

[3] "RDF/XML Syntax Specification (Revised)," World Wide Web Consortium, 2004. Available at *http://www.w3.org/TR/rdf-syntax-grammar/*, Date accessed: October 29, 2007.

[4] Tim Berners-Lee, James Hendler and Ora Lassila, "The Semantic Web," Scientific American, 2001. Available at *http://www.sciam.com/article.cfm?articleID =00048144-10D2-1C70-84A9809EC588EF21&catID=2*, Date accessed: October 29, 2007.

[5] Tim Berners-Lee, "Web Design Issues; What Semantic Web can represent," World Wide Web Consortium, 1998. Available at *http://www.w3.org/DesignIssues/RDFnot.html*, Date accessed: October 29, 2007.

[6] "XML Path Language (XPath) 2.0," World Wide Web Consortium, 2007. Available at *http://www.w3.org/TR/xpath20/*, Date accessed: October 29, 2007.

[7] "SPARQL Query Language for RDF," World Wide Web Consortium, 2007. Available at *http://www.w3.org/TR/rdf-sparql-query/*, Date accessed: October 29, 2007.

www.manaraa.com

[8] "RDF Semantics," World Wide Web Consortium, 2004. Available at *http://www.w3.org/TR/rdf-mt/*, Date accessed: October 29, 2007.

[9] "RDF Vocabulary Description Language 1.0: RDF Schema," World Wide Web Consortium, 2004. Available at *http://www.w3.org/TR/rdf-schema/*, Date accessed: October 29, 2007.

[10] "Resource Description Framework (RDF): Concepts and Abstract Syntax," World Wide Web Consortium, 2004. Available at *http://www.w3.org/TR/rdf-concepts/*, Date accessed: October 29, 2007.

[11] "Namespaces in XML," World Wide Web Consortium, 1999. Available at *http://www.w3.org/TR/1999/REC-xml-names-19990114/*, Date accessed: October 29, 2007.

[12] "XML Path Language (XPath) Version 1.0," World Wide Web Consortium, 1999. Available at *http://www.w3.org/TR/xpath*, Date accessed: October 29, 2007.

[13] "Jena - A Semantic Web Framework for Java," Sourceforge.net. Available at *http://jena.sourceforge.net/index.html*, Date Accessed: October 30, 2007.

[14] Brian McBride, "Jena: Implementing the RDF Model and Syntax Specification." Available at *http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-40/mcbride.pdf*, Date accessed: October 29, 2007.

[15] "XPath API for Java," Saxonica: XSLT and XQuery Processing. Available at *http://www.saxonica.com/documentation/xpath-api/intro.html*, Date accessed: October 29, 2007.

www.manaraa.com

[16] "W3C RDF Validation Service," World Wide Web Consortium. Available at *http://www.w3.org/RDF/Validator/* Date accessed: October 29, 2007.

[17] "RDF Data Access Use Cases and Requirements," World Wide Web Consortium, 2005. Available at *http://www.w3.org/TR/rdf-dawg-uc/*, Date accessed: October 29, 2007.

[18] Tim Berners-Lee, "Semantic Web: Why RDF is more than XML," World Wide Web Consortium, 1998. Available at *http://www.w3.org/DesignIssues/RDF-XML.html*, Date accessed: October 29, 2007.

[19] "RSS 2.0 Specification (Version 2.0.10)," RSS Advisory Board, 2002. Available at *http://www.rssboard.org/rss-specification*, Date accessed: October 29, 2007.

[20] Gabe Beged-Dov, et. al., "RDF Site Summary (RSS) 1.0," web.resource.org, 2000. Available at *http://web.resource.org/rss/1.0/spec*, Date Accessed: October 29, 2007.

[21] Tim Berners-Lee, "Relational Databases and the Semantic Web (in Design Issues)," World Wide Web Consortium, 1998. Available at *http://www.w3.org/DesignIssues/RDB-RDF.html*, Date accessed: October 29, 2007.

[22] Edd Dumbill, "Berners-Lee and the Semantic Web Vision," XML.com, 2006. Available at *http://www.xml.com/pub/a/2000/12/xml2000/timbl.html*, Date accessed: October 29, 2007.

[23] Tim Berners-Lee, "Semantic Web - XML2000," World Wide Web Consortium, 2000. Available at *http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html*, Date accessed: October 29, 2007.

[24] "Yahoo Pipes," Yahoo.com, 2007. Available at *http://pipes.yahoo.com/pipes/*, Date Accessed: October 29, 2007.

www.manaraa.com

[25] Chimezie Ogbuji, "Versa: A Path-Based RDF Query Language," XML.com, 2005. Available at *http://www.xml.com/pub/a/2005/07/20/versa.html*, Date Accessed: October 29, 2007.

[26] Damian Steer, "TreeHugger 0.1," RDFWeb, 2003. Available at *http://rdfweb.org/people/damian/treehugger/index.html*, Date Accessed: October 29, 2007.

[27] Philip McCarthy, "Search RDF Data with SPARQL," IBM, 2005. Available at *http://www.ibm.com/developerworks/xml/library/j-sparql/*, Date accessed: October 29, 2007.

[28] "XPath Tutorial," W3Schools.com. Available at *http://www.w3schools.com/xpath/default.asp*, Date accessed: October 29, 2007.

[29] "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0," World Wide Web Consortium, 2004. Available at *http://www.w3.org/TR/CCPP-struct-vocab/#CCPPArchitecture*, Date accessed: November 16, 2007.

www.manaraa.com