Querying Relational Database in Semantic Web

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Thapar University, Patiala

By:
Ruby Sawhney
(80632021)

Under the supervision of:
Ms. Shalini Batra
Lecturer (SS)

MAY 2008

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Querying Relational Database in Semantic Web"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Ms. Shalini Batra.*

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(***Ruby Sawhney)***

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(**Ms. Shalini Batra)**
Lecturer(SS)
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

(Dr. SEEMA BAWA)
Professor & Head
Computer Science & Engineering. Department
Thapar University
Patiala

(Dr. R.K.SHARMA)
Dean (Academic Affairs)
Thapar University,
Patiala.

Acknowledgement

It is a great pleasure for me to acknowledge the guidance, assistance and help I have received from Ms. Shalini Batra, Lecturer (SS), Computer Science and Engineering Department. I am thankful for her continual support, encouragement, and invaluable suggestions throughout the thesis. She not only provided me help whenever needed, but also provided me the resources required to complete this thesis report on time.

I am also thankful to Dr. Seema Bawa, Head, Computer Science and Engineering Department, for giving me an opportunity to do this work.

I would also like to thank all the staff members of Computer Science and Engineering Department for providing me all the facilities required for the completion of this work.

I am deeply grateful to my parents for the inspiration and ever encouraging moral support, which enabled me to pursue my studies.

I would like to thanks for everyday support to my dearest Classmates. I want to express my appreciation to every person who contributed with either inspirational or actual work to this report.

Ruby Sawhney

Abstract

The advent of the World Wide Web (WWW) has taken the availability of information to an unprecedented level. The rapid growth of the web poses new problems. The next generation of Web is expected to be the Semantic Web. The vision of the Semantic Web is to give data on the web a well defined meaning by representing it in languages like RDF and OWL and linking it to commonly accepted ontologies.

Although information necessary for daily work is readily accessible over internet via various search engines, the current infrastructure for managing information is not well suited for information-oriented activities. Information and functionality is scattered across applications and websites, making it difficult to aggregate and reuse just the right set of content and operations required for unique user tasks. As the Semantic Web has matured, it has become clear that there is a need to integrate older technologies. One such technology is relational databases. Most formatted data today is stored in relational databases. To be able to use relational databases data on the Semantic Web, we need a flexible but easy to use mechanism to map relational data into RDF or OWL.

In this thesis the intent is to enable Semantic access to Relational Database using D2R mapping and JENA API. D2R mapping language performs mapping between relational database schemata and RDF and then JENA (a Java query engine) is used to manipulate RDF data as SQL server is used for Relational data.

Table of Contents

Certificate---------- i
Acknowledgement---------- ii
Abstract---------- iii
Table of Contents---------- iv
List of Figures and Tables---------- vi

Chapter 1: Introduction---------- **1**
1.1 Semantic Web---------- 2
1.1.1 History---------- 2
1.1.2 Structure of Semantic Web---------- 3
1.1.3 Technologies of current Semantic Web---------- 4
1.2 XML (Extensible Markup Language)---------- 5
1.2.1 Introduction to XML---------- 5
1.2.2 Weakness of XML---------- 7
1.2.3 XML Schema---------- 7
1.3 RDF (Resource Description Framework)---------- 9
1.3.1 Introduction to RDF---------- 9
1.3.2 Basic RDF Model---------- 11
1.3.3 RDF General Format---------- 14
1.3.4 Advantages of using RDF---------- 17
1.3.5 RDFS (RDF Schema)---------- 17
1.4 D2R Mapping---------- 19
1.5 Jena---------- 21

Chapter 2: Literature Review---------- **26**

Chapter 3: Problem Statement---------- **30**
3.1 Problem Definition---------- 30
3.2 Methodology---------- 31

Chapter 4: Implementation---------- 32

4.1 Mapping from relational database table into RDF file---------- 32

4.2 Query RDF file using JENA---------- 34

4.3 Mapping between relational database tables in JENA---------- 35

Chapter 5:Conclusion and Future Scope---------- 39

5.1 Conclusion---------- 39

5.2 Future Scope---------- 40

References---------- 41

Appendix A: D2R Mapping---------- 44

Appendix B: JENA Query---------- 46

List of Publications---------- 48

List of Figures and Tables

Figure 1.1: Berners-Lee's Semantic Web 'Birthday Cake'---------- 3
Figure 1.2: Simple node and arc diagram---------- 13
Figure 1.3: Property with structured values ---------- 14
Figure 1.4: The D2R Mapping process---------- 20
Figure 1.5: JENA Architecture---------- 22
Figure 1.6: Interfaces for RDF Model in JENA---------- 23
Figure 4.1: D2R Map Fragment---------- 33
Figure 4.2: RDF output---------- 34
Figure 4.3: JENA querying RDF file---------- 35
Figure 4.4: vcardd RDF file---------- 36
Figure 4.5: evcard RDF file---------- 37
Figure 4.6: Mapped file---------- 38

Table 1.1: RDF format---------- 13
Table 4.1: vcard table---------- 32
Table 4.2: vcardd table---------- 36
Table 4.3: ecvard table---------- 37

Chapter 1
INTRODUCTION

In today's world, information is the most important resource available. Information has diverse and overlapping groups of producers and consumers, which needs several media for communication between them. *Internet* represents the latest evolution of such communication channels. The success of Internet is greatly attributed to two application layer protocols HTTP (Hypertext Transfer Protocol) and HTML (Hypertext Markup Language). Since HTML has no inherent semantics, i.e., it does not provide a description of the web page, a major shift in this paradigm is required. The World Wide Web Consortium, recognizing this need, introduced standardized protocols such as XML (Extensible Markup Language), RDF (Resource Description Language) and OWL (Web Ontology Language) to address the semantics issue.

Semantic Web is an extension of the "existing" Web in which information is given a well-defined meaning. The ultimate goal of the Semantic Web is to transform the Web into a medium through which data can be shared, understood and processed by automated tools. Semantic Web techniques, which consist of applying knowledge representation techniques in a distributed environment (potentially on a Web-wide scale), have proven useful in providing richer descriptions of Web resources.

Now a days, most of the semantic data is present in OWL, RDF and XML. And a vast amount of information resources is stored as relational data and inaccessible to RDFS-based systems. The relational data model, on the other hand, is widely accepted and currently supported by thousands of applications ranging from simple spreadsheets to complex relational databases. To be able to use relational data in a semantic context, it has to be mapped into RDF or OWL, the data format of the Semantic Web. The data model behind RDF is a directed labeled graph, which consists of nodes and labeled directed arcs linking pairs of nodes. To export data from an RDBMS into RDF, the relational database model has to be mapped to the graph-based RDF data model.

1.1 Semantic Web

1.1.1 History

The advent of the World Wide Web (WWW) has taken the availability of information to an unprecedented level. The rapid growth of the web poses new problems. Anyone can easily publish a new document or add a link to a site with no restrictions on the structure or validity of the new content. The lack of restrictions has been partly the reason of the success of the web. It has kept the expertise necessary to create content for the web low. The overabundance of unstructured and possibly faulty information has made it difficult for the users of the web to find relevant information easily. It also poses scaling difficulties for current web crawlers and search engines.

These difficulties have given rise to the successor of the current web: the Semantic Web. The Semantic Web is a mesh of information linked up in such a way as to be easily processable by machines, on a global scale. It is an efficient way of representing data on the World Wide Web, or as a globally linked database [19]. Currently under the direction of the Web's creator, Tim Berners-Lee of the World Wide Web Consortium, the Semantic Web extends the World Wide Web through the use of standards, markup languages and related processing tools [1]. With the Semantic Web almost exact results will be received when searching for information. Information can be from different sources, different domains and can provide all kinds of automated services.

The Semantic Web is a project that intends to create a universal medium for information exchange by giving meaning (semantics) to the content of documents on the Web, in a manner understandable by machines. The idea is to make computers present formatted documents to the user and enable them to deal with the contents. Therefore, documents should be supplemented with additional mark-ups containing meta information. For the realization of this idea, vocabularies for the meta information, like RDFS (Resource Description Framework Schema) or ontologies, would be required.

1.2 Structure of Semantic Web

Currently, the World Wide Web consists of documents written in HTML. This makes the Web readable for humans, but since HTML has limited ability to classify the blocks of text apart from the roles they play, the Web in its current form is very hard to understand. The purpose of the Semantic Web is to add a layer of descriptive technologies to web pages so they become readable [9].

The Semantic Web is implemented in the layers of Web technologies and standards. The layers are presented in Figure 1.1 and described as follows:

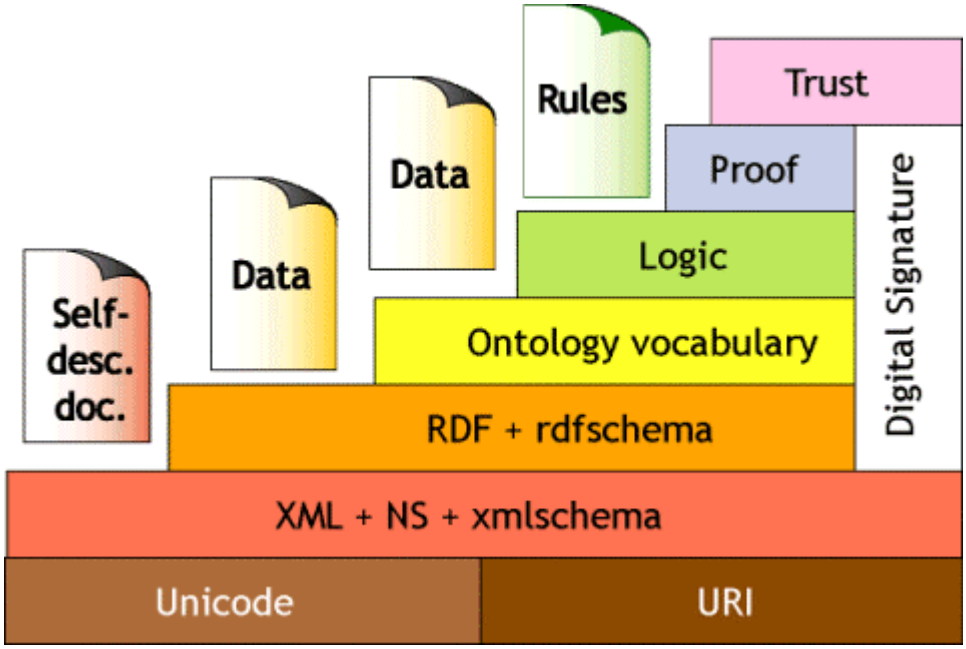

Figure 1.1: Berners-Lee Semantic Web 'Birthday Cake' [20]

• The **Unicode and Uniform Resource Identifier** (URI) layers make sure that international characters sets are used and provide means for identifying the objects in the Semantic Web. The most popular URI's on the World Wide Web are Uniform Resource Locaters (urls).

• The **XML** layer with namespace and schema definitions make sure the Semantic Web definitions can integrate with the other XML based standards. XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents. XML Schema is a language for restricting the structure of XML documents [7].

• **RDF Schema** is a simple type modeling language for describing classes of resources and properties between them in the basic RDF model. It provides a simple reasoning framework for inferring types of resources.

• The **ontology layer** supports the evolution of vocabularies as it can define relations between the different concepts. In this layer knowledge is expressed as descriptive statements, stating some relationship exists between one thing and another.

• A **digital signature** is an electronic signature that can be used to authenticate the identity of the sender of a message or the signer of a document. The Digital Signature layer ensures that the original content of the message or document is unaltered.

• The top layers **Logic, Proof and Trust** are currently being researched and simple application demonstrations are being constructed. The Logic layer enables the writing of rules while the Proof layer executes the rules and evaluates together with the Trust layer mechanism for applications whether to trust the given proof or not.

The Semantic Web comprises the standards and tools of XML, XML Schema, RDF, RDF Schema and OWL.

1.1.3 Technologies of current Semantic Web

Two important technologies for developing the Semantic Web are already in place: Extensible Markup Language (XML) and the Resource Description Framework (RDF). XML allows everyone to creates their own tags—hidden labels such as or those annotate Web pages or sections of text on a page. Scripts, or programs, can make use of these tags in sophisticated ways, but the scriptwriter has to know what the page writer uses each tag for. In short, XML allows users to add arbitrary structure to their documents but says nothing about what the structures mean [7].

Meaning is expressed through RDF, which encodes it in sets of triples; each triple is composed of a subject, predicate, and object that form elementary sentence. These triples can be written using XML tags. In RDF, a document makes assertions that particular things (people, Web pages or whatever) have properties (such as "is a sister of", "is the author of") with certain values (another person, another Web page). This structure turns out to be a natural way to describe the vast majority of the data processed by machines. Subject and object are each identified by a Universal Resource Identifier (URI), just as used in a link on a Web page. (URLs, Uniform

Resource Locators, are the most common type of URI.) The verbs are also identified by URIs, which enables anyone to define a new concept, a new verb, just by defining a URI for it somewhere on the Web.

1.2 XML (Extensible Markup Language)

1.2.1 Introduction to XML

XML (Extensible Markup Language) is an industry-standard, system-independent way of representing data. It is a markup language for documents containing structured information. XML is not a replacement for HTML. Like HTML (HyperText Markup Language), XML encloses data in tags, but there are significant differences between these two markup languages.

XML was designed to transport and store data, with focus on what data is whereas HTML was designed to display data, with focus on how data looks. HTML is about displaying information; while XML is about carrying information. While XML tags relate to the meaning of the enclosed text, HTML tags specify how to display the enclosed text. Unlike HTML, the tag vocabulary of XML is not restricted and it enables users to create arbitrary tags that have arbitrary meanings.

The following XML example shows a college with the stream and name of two students.

```
<college>
    <student>
        <stream>Computer Science</stream>
        <name>XYZ</name>
    </student>
    <student>
        <stream>Mechanical</stream>
        <name>ABC</name>
    </student>
</college>
```

The <college> and </student> tags tell a parser that the information between them is about a college and the two other tags inside the <student> tags specify that the

enclosed information is the student stream and its name. Because XML tags indicate the content and structure of the data they enclose, they make it possible to do things like archiving and searching. One of the major difference between XML and HTML is that XML tags are extensible and allow writing own XML tags to describe the content. With HTML, only limited tags can be used that have been predefined in the HTML specification. With the extensibility that XML provides, the required tags can be created for a particular type of document. Tags are defined using an XML schema language, which describes the structure of a set of XML documents and can be used to constrain the contents of the XML documents. The most widely used schema language is the Document Type Definition schema language. A schema written in this language is called a DTD. The DTD also defines the hierarchical structure of an XML document, including the order in which the tags must occur.

<!ELEMENT college (student)+>
<!ELEMENT student (stream, name) >
<!ELEMENT stream (#PCDATA) >
<!ELEMENT name (#PCDATA) >

A DTD, such as the college DTD, gives XML data its portability. If an application is sent a college document in XML format and has the college DTD, it can process the document according to the rules specified in the DTD. For example, given the college DTD, a parser will know the structure and type of content for any XML document based on that DTD. If the parser is a validating parser, it will know that the document is not valid if it contains an element not included in the DTD, such as <department>, or if the name element precedes the stream element. Other features also contribute to the popularity of XML as a method for data interchange. It is written in a text format, which is readable by both human beings and text-editing software. Applications can parse and process XML documents, and errors can be easily detected and processed as XML is human readable and easily understandable. Another feature is that because an XML document does not include formatting instructions, it can be displayed in various ways.

1.2.2 Weakness of XML

XML allows any type of data to be encoded, as there are no restrictions on XML tags. The only constraint is that a well-formed and valid XML document conforms to a DTD or schema, that is it requires the XML file adhere to a pre-defined grammar. The XML schema provides a syntactic description of data and it does not provide semantics to the data. If web users all over the world will use the same terms to describe entities, then parsing a XML file will yield both syntactic and semantic information.

XML Schemas use a restricted tag set to describe the structure of an XML document and are syntactically constrained to form a balanced tree (i.e., every starting tag must have a corresponding closing tag). Only advantage of using XML is reusing the parser and document validation. XML provides many different possibilities to encode a domain of discourse, but this leads to difficulties when understanding of foreign documents is required. RDF overcomes the semantics limitations to a great extent and can represent semantics more efficiently than XML.

1.2.3 XML Schema

An XML Schema describes the structure of an XML document. XML Schema is a much more complex schema language than DTD, and XML Schema is a language for restricting the structure of XML documents and extends XML with datatypes. A schema is composed of entities known as schema components [18]. The two major differences to DTD are the following:

• **Structural Types:** XML Schema introduces a type layer, which means that elements as well as attributes are not declared by specifying their contents directly. Instead, types are being used, with XML Schema introducing complex (for elements only) and simple (for elements and attributes) types. Types can be declared with names, reused, and can be derived from other type through various type derivation mechanisms.

• **Datatypes:** XML Schema introduces a library of built-in simple type datatypes, which can be used or derived for often used concepts such as URIs, dates, times, or durations. This is a huge improvement over DTDs, which did not have any application-oriented datatype concept and did not even allow to specify that an attribute value had to be a number. The type derivation mechanisms described above can be used to derive own types from the built-in types, for example defining a number type with lower and upper bounds.

Apart from the most fundamental features and improvements in comparison to DTDs described above (structural types, datatypes, and XML syntax), XML Schema also has the following features:

• **Namespace Support:** XML Schema supports XML Namespace by allowing a schema to specify its target namespace, and by allowing a schema to refer to other namespaces through wildcards which only allow elements and/or attributes from specified namespace(s). This makes it easier to create modular vocabularies, which can be combined on the namespace level.

• **Identity Constraints:** DTDs has the concept of ID/IDREF attributes, where IDs are guaranteed to be unique within a document, and IDREFs are guaranteed to reference an existing ID. These are both global concepts, so it is impossible to reuse the same ID value on different elements. XML Schema introduces identity constraints as a generalized ID/IDREF concept, which decouples constraints from the type of the values, makes it possible to define scoped constraints, and supports multi-field and element-based constraints.

• **Improved Model Groups:** The sequence and choice model groups of XML DTDs are complemented with a all model group, which allows the specification of parts which must occur, but in any order. As another improvement, XML allows more specific mixed content, which means that is possible to specify a sequence of elements in a mixed content model, and these elements must then occur in this sequence within the mixed content.

• **Improved Reuse of Schema Components:** XML Schema introduces named model groups and attribute groups, which are both ways to group and name parts of complex types, which can then be reused in different contexts. These features help to build better schemas by defining and using reusable components.

• **Type Annotations:** XML Schema is used for validating documents, but it is equally important for the type annotation of documents. This means that an XML Schema can be used to augment an XML document with type annotations, which can then be used in later processing stages for type-aware processing of the document's contents. This is defined by the XQuery 1.0 and Xpath 2.0 Data Model , which defines a data model for XML documents on the foundations of the XML Infoset and XML Schema.

While XML Schema has added new features to the previously rather simple world of XML schemas using DTDs, it has also introduced new complexity, and in particular a whole new modeling layer through the structural types using complex and simple types. This means that writing schemas has become more complex, and experience has shown that many users of XML Schema are not fully aware of language's complexity and intricacies, and thus do not use the language in the best possible way. In particular many XML Schemas are structurally equivalent to DTDs, and simply use the XML Schema formalism because this is required by the environment.

1.3 RDF (Resource Description Framework)

1.3.1 Introduction To RDF

The World Wide Web was originally built for human consumption, and although everything on it is *machine-readable*, this data is not *machine-understandable*. It is very hard to automate anything on the Web, and because of the volume of information the Web contains; it is not possible to manage it manually. The solution is to use *metadata* to describe the data contained on the Web. Metadata is "data about data" (for example, a library catalog is metadata, since it describes publications) or specifically in the context of this specification "data describing Web resources" [11]. The distinction between "data" and "metadata" is not an absolute one; it is a

distinction created primarily by a particular application, and many times the same resource will be interpreted in both ways simultaneously.

RDF and its schema language Resource Description Framework Schema (RDFS) were proposed by the World Wide Web Consortium to overcome the semantic limitations of XML. Resource Description Framework (RDF) is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web. RDF emphasizes facilities to enable automated processing of Web resources.
RDF can be used in a variety of application areas for example:

- In *resource discovery* to provide better search engine capabilities,
- In *cataloguing* for describing the content and content relationships available at a particular Web site, page, or digital library,
- By *intelligent software agents* to facilitate knowledge sharing and exchange,
- In *content rating*, in describing *collections of pages* that represent a single logical "document",
- For describing *intellectual property rights* of Web pages,
- For expressing the *privacy preferences* of a user as well as the *privacy policies* of a Web site.
- RDF with *digital signatures* will be key to building the "Web of Trust" for electronic commerce, collaboration, and other applications.

One of the goals of RDF is to make it possible to specify semantics for data based on XML in a standardized, interoperable manner. RDF is a model of metadata and only addresses by reference many of the encoding issues that transportation and file storage require (such as internationalisation, character sets, etc.). For these issues, RDF relies on the support of XML. It is also important to understand that this XML syntax is only one possible syntax for RDF and alternate ways to represent the same RDF data model may emerge.

The broad goal of RDF is to define a mechanism for describing resources that makes no assumptions about a particular application domain, nor defines (a priori) the semantics of any application domain.

RDF has a class system much like many object-oriented programming and modelling systems. A collection of classes is called a *schema*. Classes are organized in a hierarchy, and offer extensibility through subclass refinement. This way, in order to create a schema slightly different from an existing one, it is not necessary to "reinvent the wheel" but one can just provide incremental modifications to the base schema. Through the share ability of schemas, RDF will support the reusability of metadata definitions [22]. Due to RDF's incremental extensibility, agents processing metadata will be able to trace the origins of schemata they are unfamiliar with back to known schemata and perform meaningful actions on metadata they weren't originally designed to process. The share ability and extensibility of RDF also allows metadata authors to use multiple inheritance to "mix" definitions, to provide multiple views to their data, leveraging work done by others. In addition, it is possible to create RDF instance data based on multiple schemata from multiple sources. Schemas may themselves be written in RDF; RDF Schema describes one set of properties and classes for describing RDF schemas.

As a result of many communities coming together and agreeing on basic principles of metadata representation and transport, RDF has drawn influence from several different sources. The main influences have come from the *Web standardization community* itself in the form of HTML metadata and PICS, the *library community*, the *structured document community* in the form of SGML and more importantly XML, and also the *knowledge representation (KR) community*. There are also other areas of technology that contributed to the RDF design; these include object-oriented programming and modelling languages, as well as databases. While RDF draws from the KR community, that RDF does not specify a mechanism for *reasoning*.

1.3.2 Basic RDF Model

RDF is model and XML syntax for representing information in a way that allows programs to understand the intended meaning. RDF represents information about resources on the Web, i.e., metadata about resources. A resource in RDF is mapped to a Uniform Resource Identifier (URI) and is described in terms of its properties. The purpose of RDF is to make statements about resources, which can be viewed as labeled edge (property) between two nodes (object and property value).

The foundation of RDF is a model for representing named properties and property values. RDF properties may be thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs. RDF properties also represent relationships between resources and an RDF model can therefore resemble an entity-relationship diagram. In object-oriented design terminology, resources correspond to objects and properties correspond to instance variables.

The RDF data model is a syntax-neutral way of representing RDF expressions. Two RDF expressions are equivalent if and only if their data model representations are the same. The basic data model consists of three object types:

- **Resources:** All things being described by RDF expressions are called *resources*. A resource may be an entire Web page; such as the HTML document "http://www.w3.org/Overview.html" for example. A resource may be a part of a Web page; e.g. a specific HTML or XML element within the document source. A resource may also be a whole collection of pages; e.g. an entire Web site. A resource may also be an object that is not directly accessible via the Web; e.g. a printed book. Resources are always named by URIs plus optional anchor ids. Anything can have a URI; the extensibility of URIs allows the introduction of identifiers for any entity imaginable.

- **Property:** A *property* is a specific aspect, characteristic, attribute, or relation used to describe a resource. Each property has a specific meaning, defines its permitted values, the types of resources it can describe, and its relationship with other properties. This object type does not address how the characteristics of properties are expressed.

- **Statement:** A specific resource together with a named property plus the value of that property for that resource is an RDF *statement*. These three individual parts of a statement are called, the *subject*, the *predicate*, and the *object* respectively. It's built on the concept of a statement, a triple of the form {predicate, subject, object}. The interpretation of a triple is that <subject> has a property <predicate> whose value is <object>. In RDF a <subject> is always a resource named by a URI with an optional anchor id, <predicate> is a

property of the resource, and the <object> is the value of the property for the resource. The object of a statement (i.e., the property value) can be another resource or it can be a literal; i.e., a resource (specified by a URI) or a simple string or other primitive data type defined by XML. In RDF terms, a *literal* may have content that is XML markup but is not further evaluated by the RDF processor.

Examples

Resources are identified by a *resource identifier*. A resource identifier is a URI plus an optional anchor id. Consider as a simple example the sentence:

Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila.

{Creator, http://www.w3.org/Home/Lassila, Ora Lassila}

This sentence has the following parts:

Subject (Resource)	http://www.w3.org/Home/Lassila
Predicate (Property)	Creator
Object (literal)	"Ora Lassila"

Table 1.1: RDF Format

The diagram of this RDF statement pictorially using directed labelled graphs (also called "nodes and arcs diagrams") is shown in figure 1.2. In these diagrams, the nodes (drawn as ovals) represent resources and arcs represent named properties. Nodes that represent string literals will be drawn as rectangles.

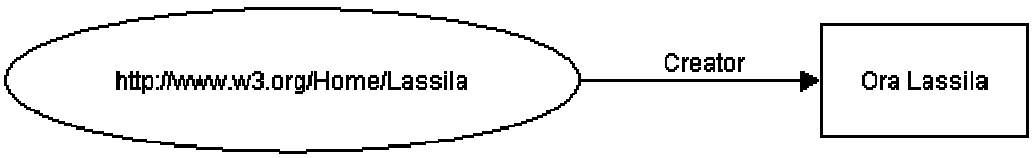

Figure 1.2: Simple node and arc diagram

In prose, such a sentence would be:

> *The individual whose name is Ora Lassila, email <lassila@w3.org>, is the creator of http://www.w3.org/Home/Lassila.*

The intention of this sentence is to make the value of the Creator property a structured entity. In RDF such an entity is represented as another resource. The sentence above does not give a name to that resource; it is anonymous, so in the diagram below we represent it with an empty oval:

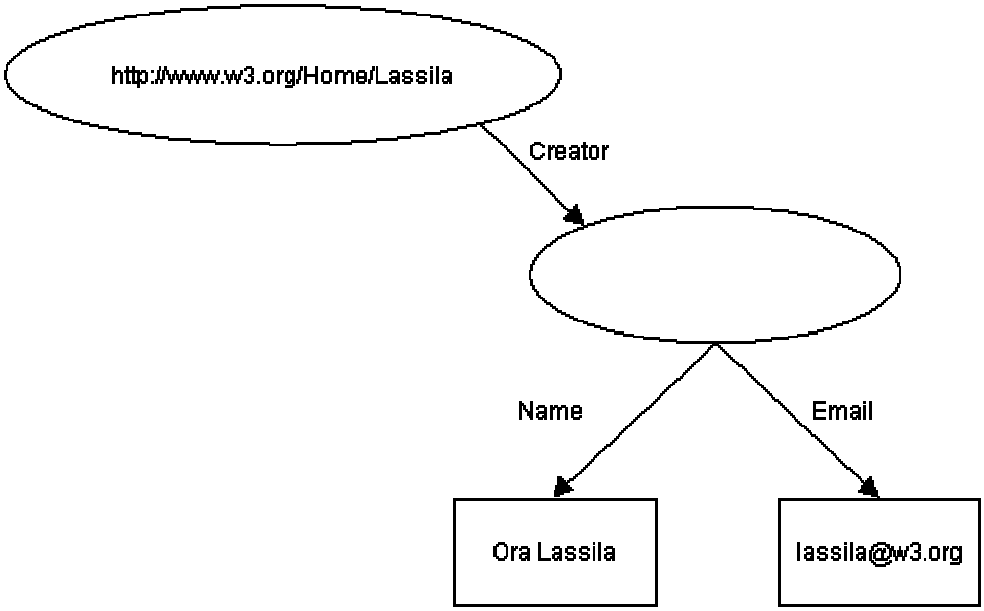

Figure 1.3: Property with structured value

1.3.3 RDF General format

```
<?xml version="1.0"?>
<Class rdf:ID="Resource"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns="uri">
        <property>value</property>
       <property>value</property>
        ...
</Class>
```

Element description

rdf:ID	:	RDF provides an ID attribute for identifying the resources being described. The value of rdf:ID is a "relative URI". The "complete URI" is obtained by concatenating the URL of the XML document with "#" and then the value of rdf:ID.
resource	:	Identifies the **resource** being described.
xmlns:rdf	:	The ID attribute is in the RDF namespace.
Class	:	Identifies the **type**(class) of the resource being described.
property	:	These are properties, or attributes, of the type (class).
value	:	Values of the properties.

Some other elements

rdf:about	:	Instead of identifying a resource with a relative URI (which then requires a base URI to be attached before relative URI), we can give the complete identity of a resource. However, we use rdf:about, rather than rdf:ID.

rdf:ID versus rdf:about :

• To **introduce** a resource, and provide an initial set of information about a resource use rdf:ID.

• To **extend** the information about a resource use rdf:about .

Data Model-XML Syntax

•Well-formed XML

•Validation through RDF schemas

•No XML/DTD type of validation

Example

Here's an example of an XML document that specifies data about China's Yangtze river:

```
<?xml version="1.0"?>
<River id="Yangtze" xmlns="http://www.geodesy.org/river">
    <length>6300 kilometers</length>
    <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
    <endingLocation>East China Sea</endingLocation>
</River>
```

Data about the Yangtze River. It has a length of 6300 kilometers. Its startingLocation is western China's Qinghai-Tibet Plateau. Its endingLocation is the East China Sea.

Modify the previous XML document so that it is also a valid RDF document:

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.geodesy.org/river#">
    <length>6300 kilometers</length>
    <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
    <endingLocation>East China Sea</endingLocation>
</River>
```

The three different types of container objects defined by the RDF specification are Bag, Sequence and Alternative.

- **Bag**

  It is an unordered list of resources or literals. *Bag*s are used to declare that a property has multiple values and when there is no significance to the order in which the values are given.

- **Sequence**

  It is an ordered list of resources or literals. *Sequence* is used to declare that a property has multiple values and when the order of the values is significant. *Sequence* might be used, for example, to preserve an alphabetical ordering of values. Duplicate values are permitted.

- **Alternative**

It is a list of resources or literals that represent alternatives for the (single) value of a property. *Alternative* might be used to provide alternative language translations for the title of a work, or to provide a list of Internet mirror sites at which a resource might be found. An application using a property whose value is an alternative collection is aware that it can choose any one of the items in the list as appropriate.

1.3.4 Advantages of using RDF

1. The RDF format, if widely used, will help to make XML more interoperable:
    - Tools can instantly characterize the structure, "this element is a type (class), and here are its properties".
    - RDF promotes the use of standardized vocabularies, standardized types (classes) and standardized properties.
2. The RDF format gives a structured approach to design XML documents. The RDF format is a regular, recurring pattern.
3. It enables to quick identification of weaknesses and inconsistencies of non-RDF-compliant XML designs. It helps to better understand the data.
4. RDFS gives the benefits of both worlds:
    - Standard XML editors and validators to create, edit, and validate XML file.
    - use the RDF tools to apply inferencing to the data.
5. It positions the data for the Semantic Web.

1.3.5 RDFS (RDF Schema)

RDF asserts statements about resources in terms of properties, the description of these properties is provided by the RDF schema specification, i.e. RDF schemas add semantics to RDF. They provide a type system for resources and define relationships between resources and can be encoded using RDF resources, such as Class, SubclassOf, Property, etc. which are defined in the namespace "rdfs". Unlike XML schemas, they do not impose syntactic constraints on the RDF document but provide a

mechanism to describe metadata about resources. Every entity in RDF is a resource and a class represents a group of resources [15]. Every resource in RDF is an instance of at least one class. RDF schemas use a core set of classes and properties to describe metadata about web resources, i.e., there exists a subset of RDF resources using that all other RDF schema constructs can be defined. The core classes in RDF schema are rdfs:class, rdfs:resource, rdfs:property and rdfs:constraint resource. rdfs:class represents the set of entities used to describe concepts in a schema, rdfs:resource represents the set of all resources, rdfs:property represents the set of all RDF properties and rdfs:constraintresource represents the set of all constraints applicable to RDF resources. Core properties include properties of the class rdfs:property namely, rdf:type, rdfs:subclassof, rdfs:range and rdfs:domain. The RDF model assigns a dual role to these core classes and properties, which is used to define other schema constructs and can be used in the specification of an RDF schema itself that causes self-referentiality in the RDF model.

Some important feature of RDFS:

- RDF describes resources with classes**,** properties, and values. In addition, RDF also needs a way to define application-specific classes and properties. Application-specific classes and properties must be defined using extensions to RDF. One such extension is RDF Schema RDFS.
- RDF Schema does not provide actual application-specific classes and properties. Instead RDF Schema provides the framework to describe application specific classes and properties.
- Classes in RDF Schema are much like classes in object oriented programming languages. This allows resources to be defined as instances of classes, and subclasses of classes.

Example of RDFS:

A simple RDF file

An RDF description

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xml:base= "http://www.animals.fake/animals#">
```

```
        <rdf:Description rdf:ID="animal">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
        schema#Class"/>
        </rdf:Description>
        <rdf:Description rdf:ID="horse">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
        schema#Class"/>
        <rdfs:subClassOf rdf:resource="#animal"/>
        </rdf:Description>
    </rdf:RDF>
```

In the example above, the resource "horse" is a subclass of the class “animal”.

Since an RDFS class is an RDF resource, the example above can be simplified by using rdfs:Class instead of rdf:Description, and drop the rdf:type information:

RDFS file:

```
    <?xml version="1.0"?>
    <rdf:RDF
        xmlns:rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xml:base= "http://www.animals.fake/animals#">
        <rdfs:Class rdf:ID="animal" />
        <rdfs:Class rdf:ID="horse">
        <rdfs:subClassOf rdf:resource="#animal"/>
        </rdfs:Class>
    </rdf:RDF>
```

1.4 D2R Mapping Language

D2R is a declarative language to describe mappings between relational database schemas and RDF. A D2R processor to export data from a relational database into RDF can use this mapping.

D2R MAP is a declarative, XML-based language to describe such mappings. The main goal of the language design is to allow flexible mappings of complex relational

structures without changing the existing database schema. This flexibility is achieved by employing SQL statements directly in the mapping rules. The resulting record sets are grouped afterwards and the data is mapped to the created instances. This approach allows the handling of binary and higher degree relationships, multivalued class properties, complex conditions and highly normalized table structures, where instance data is spread over several tables [4]. The mapping process performed by the D2R processor has four logical steps as shown in Figure 1.4:

1. Selection of a record set from the database using SQL
2. Grouping of the record set by the d2r: groupBy columns.
3. Creation of class instances and identifier construction.
4. Mapping of the grouped record set data to instance properties.

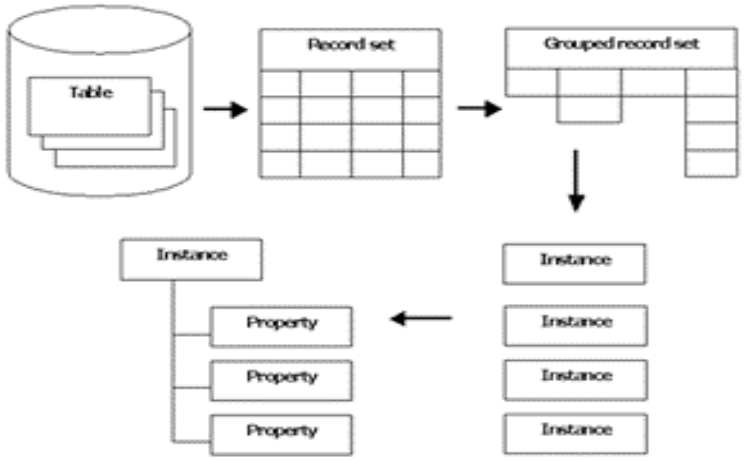

Figure 1.4: The D2R mapping process [4].

For each class or group of similar classes a record set is selected from the database. The record set is grouped according to the groupBy columns of the specific ClassMap. Then the class instances are created and assigned a URI or a blank node identifier. Finally, the instance properties are created using data type and object property bridges. The division between step three and four allows references to blank nodes within the model and to instances dynamically created in the mapping process.
Another aim is to keep D2R MAP as simple as possible. Apart from elements to describe the database connection and the namespaces used, the actual mappings are expressed with just three elements:

- For each class or group of similar classes in the RDF a ClassMap element is used. Each ClassMap has an sql attribute and a groupBy attribute. To create instance URIs, patterns and value substitution tables can be used.
- The instance properties are constructed with DataTypePropertyBridge elements for literal properties, which can be typed using XML datatypes and xml:lang attributes. Datatype properties can be converted similarly using patterns and value substitution tables.
- References to external resources or instances within the model are created with an ObjectPropertyBridge element. To refer to the instances created on the fly, a referredClass together with a referredGroupBy attribute is used.

Multiple values of a single property can be put in rdf:Bag, rdf:Alt or rdf:Seq containers, using the useContainer attribute together with a DataTypePropertyBridge or ObjectPropertyBridge element.

1.5 JENA

Jena is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule based inference engine. Jena enables programmatic access for RDFS and OWL documents [21]. It is an open-source library that grew out of work with the HP Labs Semantic Web Programmer and has become the standard API for semantic-web document manipulation. The Jena Framework includes:

• A RDF API
• Reading and writing RDF in RDF/XML, N3 and N-Triples
• An OWL API
• In-memory and persistent storage
• SPARQL query engine

JENA Architecture

The heart of the Jena2 architecture is the RDF graph, a set of triples of nodes. As seen in figure 1.5, graph layer following the RDF abstract syntax, is minimal by design wherever possible functionality is done in other layers. This permits a range of implementations of this layer such as in memory or persistence triple stores.

The EnhGraph layer is the extension point on which to build APIs. Within Jena2 the functionality offered by the EnhGraph layer is used to implement the Jena1 Model API and the new Ontology functionality for OWL and RDFS, upgrading the Jena1 DAML API [6]. I/O is done in the Model layer. The Jena2 architecture supports fast path query that goes all the way through the layers from RDQL at the top right through to an SQL database at the bottom, allowing user queries to be optimized by the SQL query optimizer.

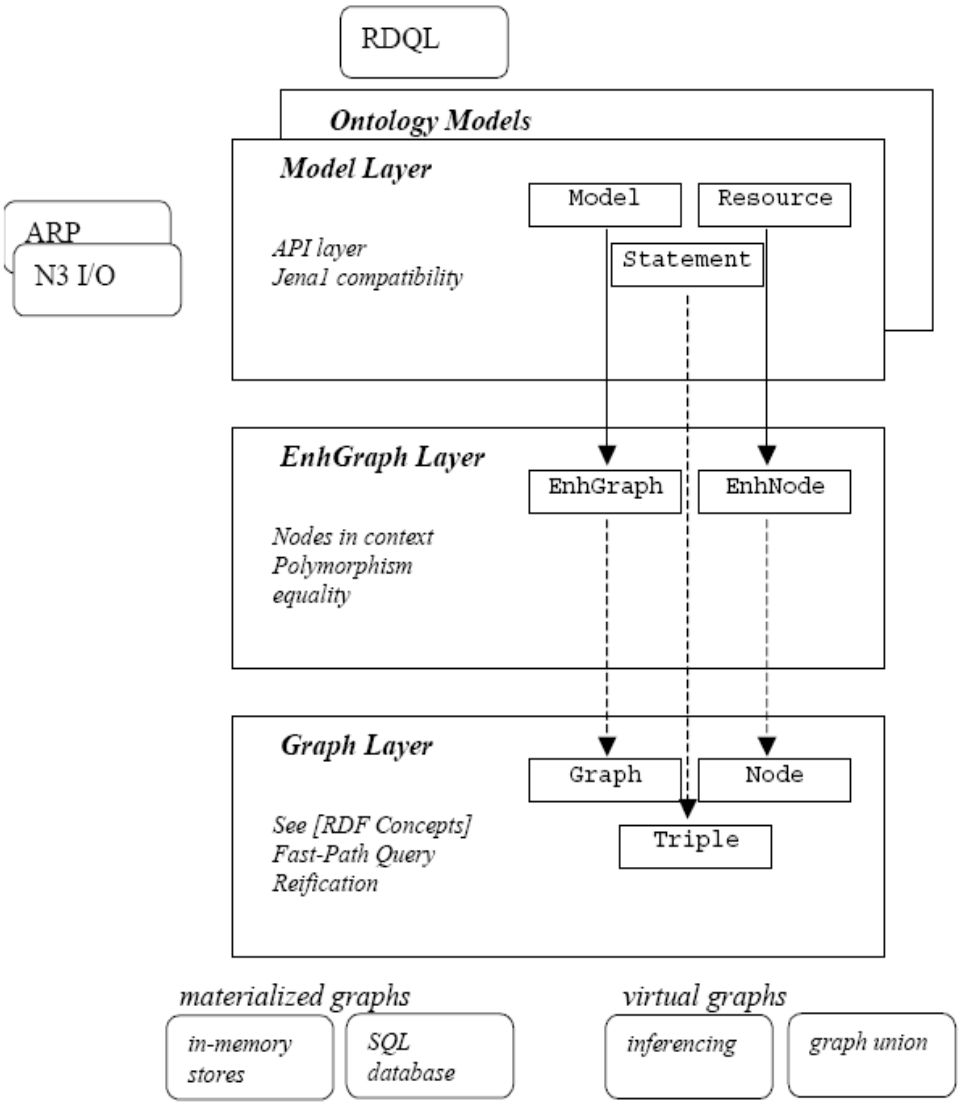

Figure1.5: JENA Architecture [15]

The two key architectural goals of Jena are:

• Multiple, flexible presentations of RDF graphs to the application programmer. This allows easy access to, and manipulation of, data in graphs enabling the application programmer to navigate the triple structure.

• A simple minimalist view of the RDF graph to the system programmer wishing to expose data as triples.

The functions of Jena's interfaces and classes

The core of the semantic web recommendation is the RDF diagram, which is used as the universal data structure. An RDF diagram is a group of simple (S, P, O) triples. Here S refer to subject, O refer to object, P refers to the predicate working on the (S, O). Jena2 also uses the diagram as its core interfaces, and all the other components are built on this diagram. Jena allows one to parse, create, and search RDF models.

Jena defines a number of interfaces for accessing and manipulating RDF statements as shown in the figure 1.6.

The RDFNode interface provides a common base for all elements that can be parts of RDF triples. The Literal interface represents literals such as "white bird" or 786 that can be used as the <object> in {predicate, subject, object} triples. The Literal interface provides accessor methods to convert literals to various Java types such as String, int, and double.

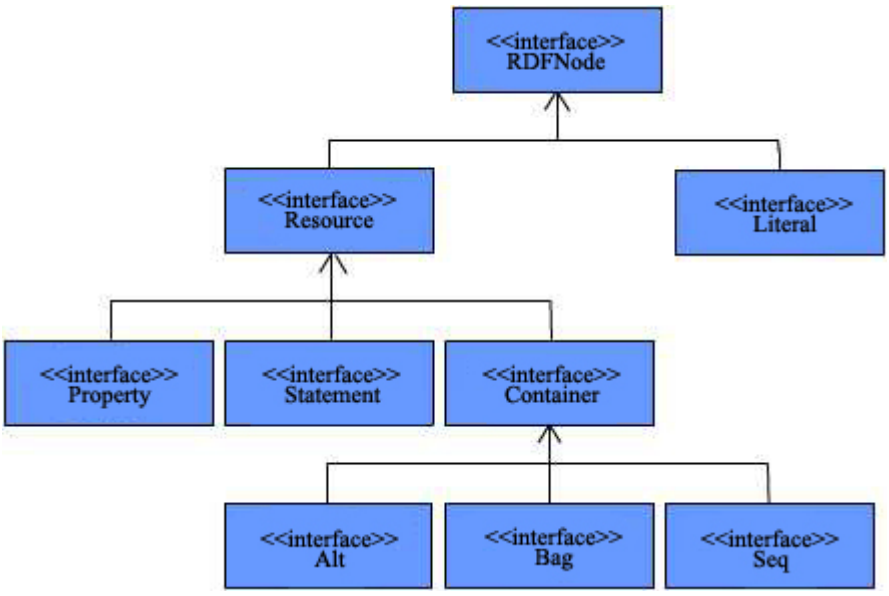

Figure 1.6: Interfaces for RDF Models in JENA [17]

Objects implementing the Property interface can be the <predicate> in {predicate, subject, object} triples. The Statement interface represents a {predicate, subject, object} triple. It can also be used as the <object> in a triple since RDF allows statements to be nested. Objects implementing the Container, Alt, Bag, or Seq interface can be the <object> in {predicate, subject, object} triples.

The Jena tool package mainly includes the following functions:

- **An RDF API** (mainly includes the com.hp.hpl.jena.rdf.model package): It uses the statement centric methods for manipulating an RDF model as a set of RDF triples. It supports cascading method calls for more convenient programming. It supports the RDF containers - bag, alt and seq and the application can extend the behavior of resources. It also supports the integrated parsers and writers for RDF/XML (ARP), N3 and N-TRIPLES and typed literals [14].

- **RDQL query language** (mainly includes the com.hp.hpl.jena.rdql passage): RDQL is a query language for RDF data. The implementation in Jena is coupled to relational database storage so that optimized query is performed over data held in a Jena relational persistence store.

- **Reasoning Subsystem** (mainly includes the com.hp.hpl.jena.reasoner package): The Jena2 reasoner subsystem includes a generic rule based inference engine together with configured rule sets for RDFS and for the OWL/Lite subset of OWL Full. Of these components, the underlying rule engine and the RDFS configuration should be reasonably stable. The OWL configuration is preliminary and still under development.

- **In-memory and persistent storage** (mainly includes the com.hp.hpl.jena.db passage): Jena provides the temporary memory-based RDF model methods and also provides the permanent data base storage method. Currently, Jena2 can use three SQL database engines, MySQL, Oracle and PostgreSQL. These are supported on Linux and Windows XP.

The general structure of Jena

Jena is a Java class library, and is composed mainly of API (Application Programming Interface) and SPI (System Programming Interface). In general, the users only use API in programming. And SPI mainly provides the core data structure for the Jena system and most of the functions are realized in API. Like all the Java codes, the Jena class library is maintained by the Package mechanism. The Jena API has been defined in terms of interfaces so that application code can work with different implementations without change.

In the RDF API, the key RDF package for the application developer is com.hp.hpl.jena.rdf.model. This package contains interfaces for representing models, resources, properties, literals, statements and all the other key concepts of RDF, and a ModelFactory for creating models [14]. To keep the application code independent of the implementation, it is best not to specify class implementations if using interfaces wherever possible. So applications should rarely, if ever, use these classes directly.

Chapter 2
LITERATURE REVIEW

Relational databases have been an important part of the enterprise IT infrastructure, mainly because of their proven efficiency in dealing with huge volumes of data. However, the advent of the Internet has brought many end users into the information cyberspace. They usually have little, if any, training in either query languages or database systems, and thus have difficulties using database languages, e.g. SQL. A new semantic query language, such as SPARQL, is much preferred, and will enable users to retrieve information solely by the semantic understanding of the applicable domain.

The main drawback of relational systems is that database schemas change over the time, even though the semantics of the data have not changed. Non-semantic changes may be caused by many different things, including schema normalization or de-normalization, migrating from one DBMS product to another, changing of data type for a particular data field, or by using different techniques, for example using a stored procedure instead of a traditional view [19]. And the addition of a new set of data or archiving existing data may also result in schema changes.

For any software-based information consumer, all these non-semantic changes imply one thing: the existing pre-defined or pre-cached query statements have to be modified accordingly. Otherwise, an exception may be returned instead of the query answers. Given the fact that a database usually has many software applications on top of it, it is really challenging and expensive to modify all the applications appropriately and in a timely fashion. There is an approach that is able to hide these non-semantic changes from software applications so that it is not necessary to modify an application as long as there isn't any change in semantics.

With the ever-growing information sharing requirement in almost every enterprise, retrieved information (query answers) needs to be shared among many information consumers, not only humans but also computers and their software components. This implies that the semantics of query answers must be both human and machine

understandable. In the database query, a flat table, with multiple rows, usually represents answers and each row may have a number of fields. Suppose that a table has two columns, one being aircraft tail number and the other being a part number. It's difficult to understand what this data really means without asking any questions to another person.

Semantic access to existing relational database data holds the promise of bringing explicit semantics at several different levels and for different parties: the semantics of the domain, data content, user-defined query itself, and the returned query answers. Its not only provides information consumers a more convenient and user-friendlier interface to retrieve information, but it also offers a foundation for better system maintainability, better semantic interoperability among multiple data systems, and, hence, better data leverage.

Semantic web is relying on two important technologies: RDF and Ontology. While the development of RDF as a general metadata framework, and as such, a simple knowledge representation mechanism for the web, was heavily inspired by the PICS specification, no one individual or organization invented RDF. RDF is a collaborative design effort. Several W3C Member companies are contributing intellectual resources. It is based upon the XML design as well as proposals submitted by Microsoft's and Netscape [11]. Other metadata efforts, such as the Dublin Core and the Warwick Framework have also influenced the design of the RDF. RDF provides interoperability between applications that exchange machine-understandable information on the Web.

RDF emphasizes facilities to enable automated processing of Web resources. RDF provides a model for describing resources. It's built on the concept of a statement, a triple of the form {predicate, subject, object}. The interpretation of a triple is that <subject> has a property <predicate> whose value is <object>.

RDF statement subject, or object, is identified with a Uniform Resource Identifier [3]. This is something like <http://www.w3.org/> or http://www.w3.org/2000/10/swap/test/s1.n3#River.

The RDF Vocabulary Description Language, RDF Schema (RDFS), adds semantics to an RDF data model by defining its vocabulary. It is similar to XML and XML Schema, in a way that it transfers a general RDF concept to a specific application domain. Written in RDF, the schema description provides possibilities to characterize groups of related resources and their relationships. Moreover, it allows for inference with the use of metadata.

The semantic web data model is directly connected with the model of relational databases. A relational database consists of tables, which consists of rows, or records. Each record consists of a set of fields. The record is nothing but the content of its fields, just as an RDF node is nothing but the connections the property values [12]. In this mapping

- A record is an RDF node;
- The field (column) name is RDF propertyType; and
- The record field (table cell) is a value.

Indeed, one of the main driving forces for the Semantic web has always been the expression, on the Web, of the vast amount of relational database information in a way that can be processed by machines.
RDF's serialization format (its syntax in XML) is a suitable format for expressing relational database information.

Jena, a Semantic Web Framework is used for building Semantic Web applications and it provides a programmatic environment for RDF, RDFS and OWL. Jena was at first developed to be a Java API for RDF, but later implementations include other functionalities such as the Jena2 ontology API.

Jena is a leading Semantic Web programmers' toolkit. The heart of the Semantic Web recommendations is the RDF Graph, which is a universal data structure. RDF graph is simply a set of triples. Jena allows easy access to, and manipulation of, data in graphs enabling the application programmer to navigate the triple structure. Particularly, the Model API presents the graph using the terms and concepts from the RDF recommendations, and the Ontology API presents the graph using concepts from OWL and RDFS.

SPARQL a query language and data access protocol for the Semantic Web. SPARQL is defined in terms of the W3C's RDF data model and will work for any data source that can be mapped into RDF. The specification is under development by the RDF Data Access Working Group. SPARQL delivers information from RDF graphs, which are a set of triples [16]. The triples consist of a subject, a predicate, and an object. SPARQL provides functionalities to

• Extract information represented as literals, blank nodes, and URI's,

• Gather RDF sub graphs, and

• Build new RDF graphs upon information achieved from the queried graphs.

Chapter 3
PROBLEM STATEMENT

3.1 Problem Definition

During the literature survey it was found that the Semantic Web aims to build a common framework that allows data to be shared and reused across applications, enterprises, and community boundaries. It proposes to use RDF as a flexible data model and use ontology to represent data semantics. Currently, relational models and XML tree models are widely used to represent structured and semi-structured data. But they offer limited means to capture the semantics of data. One of the main bottleneck of the Semantic Web is the lack of semantically rich data, since most of the information is still stored in relational databases. But Semantic query language will enable users to retrieve information solely by the semantic understanding of the applicable domain. It not only provides information consumers a more convenient and user-friendlier interface to retrieve information, but it also offers a foundation for better system maintainability, better semantic interoperability among multiple data systems, and, hence, better data leverage.

The main goal of thesis is to semantically access Relational database, using D2R, a declarative language to perform mappings between relational database schemas and RDF and JENA API. It is an XML-based language, which allows flexible mappings of complex relational structures without changing the existing database schema. Jena is an open source semantic web tool based on Java used for building Semantic Web applications. JENA RDF API and OWL API (Application Programming interface), is used for manipulating RDF/XML files.

3.2 Methodology

The step-by-step methodology to be followed in enabling semantic access to relational database is given below:

- Create a table in Relational database.
- Map the table in to an RDF file using D2R Mapping. The mapping process performed by the D2R processor has four logical steps:
  1. Selection of a record set from the table using SQL
  2. Grouping of the record set by the d2r: groupBy columns.
  3. Creation of class instances and identifier construction.
  4. Mapping of the grouped record set data to instance properties.
- Query the RDF file using JENA API.
- To perform mapping between relational databases, create two tables in relational database.
- Map these tables into their RDF files. Merge these two RDF files into one RDF file based on their primary keys.

Chapter 4
IMPLEMENTATION

4.1 Mapping from Relational database table to RDF File

In order to implement mapping from Relational Database table to RDF file, D2R mapping language is used. The mappings can be used by a D2R processor to export data from a relational database into RDF. The processor is implemented in Java and is based on the Jena API.

Example

The following example illustrates the use of a D2R Mapping to export data about vcard table from a database named vcard into RDF.

PersonName	FirstName	LastName
John Smith	John	Smith
Sarah Jones	Sarah	Jones
Matt Jones	Matthew	Jones
Becky Smith	Rebecca	Smith

Table 4.1: vcard

D2R Map Fragment

The first three sub elements in figure 4.1 define the database connection, the desired output format and an example namespace.

The ClassMap element describes the mapping for the vcard class. The instance URIs are created using a pattern. In ClassMap element two attributes are required: An SQL statement to select data from the database and the groupBy attribute containing a columnlist to group the rows of the resultset.

A DatatypePropertyBridge defines a bridge between a column of the result set and a literal property of the instances created.

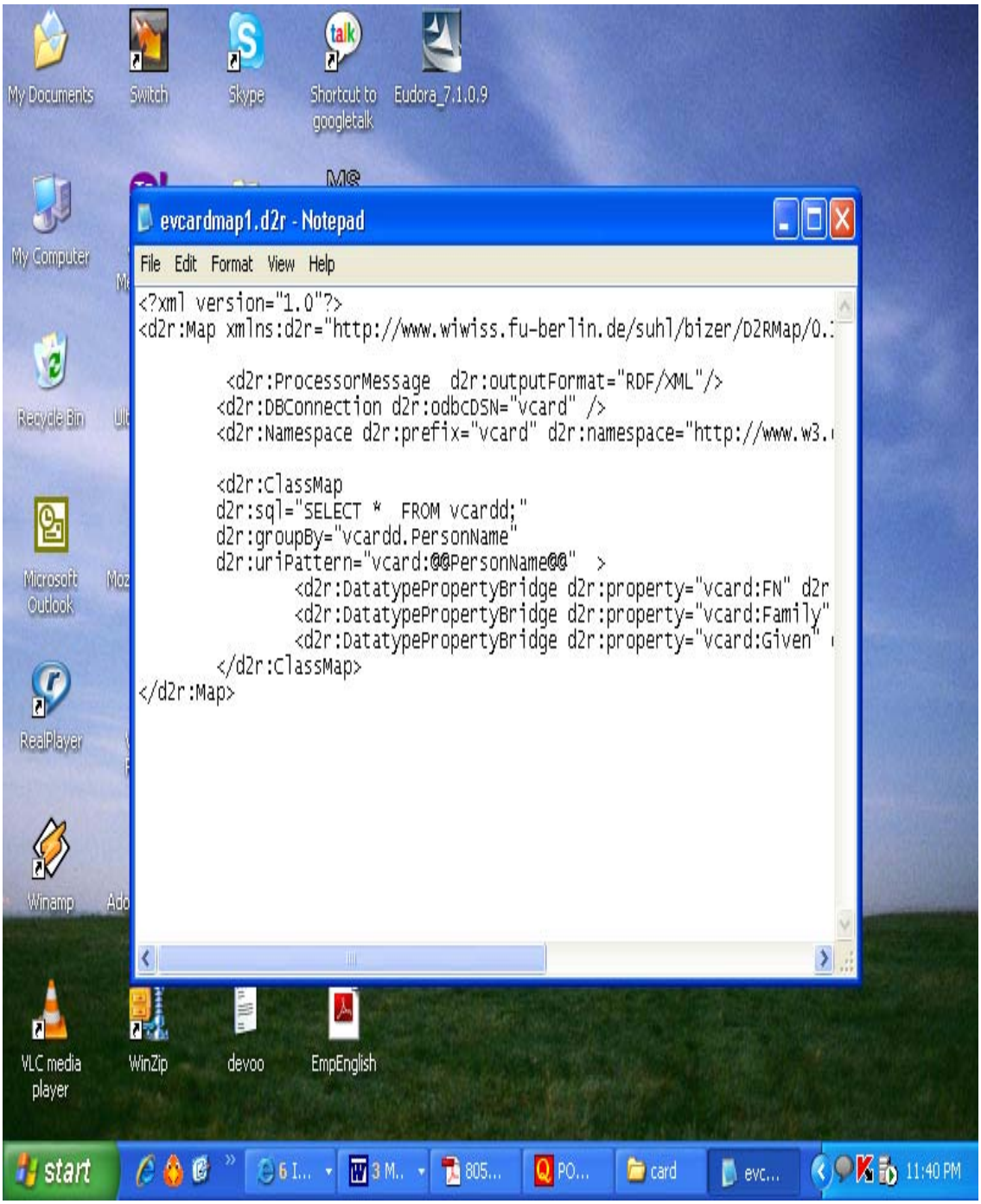

Figure 4.1: D2R Map Fragment

RDF Output:

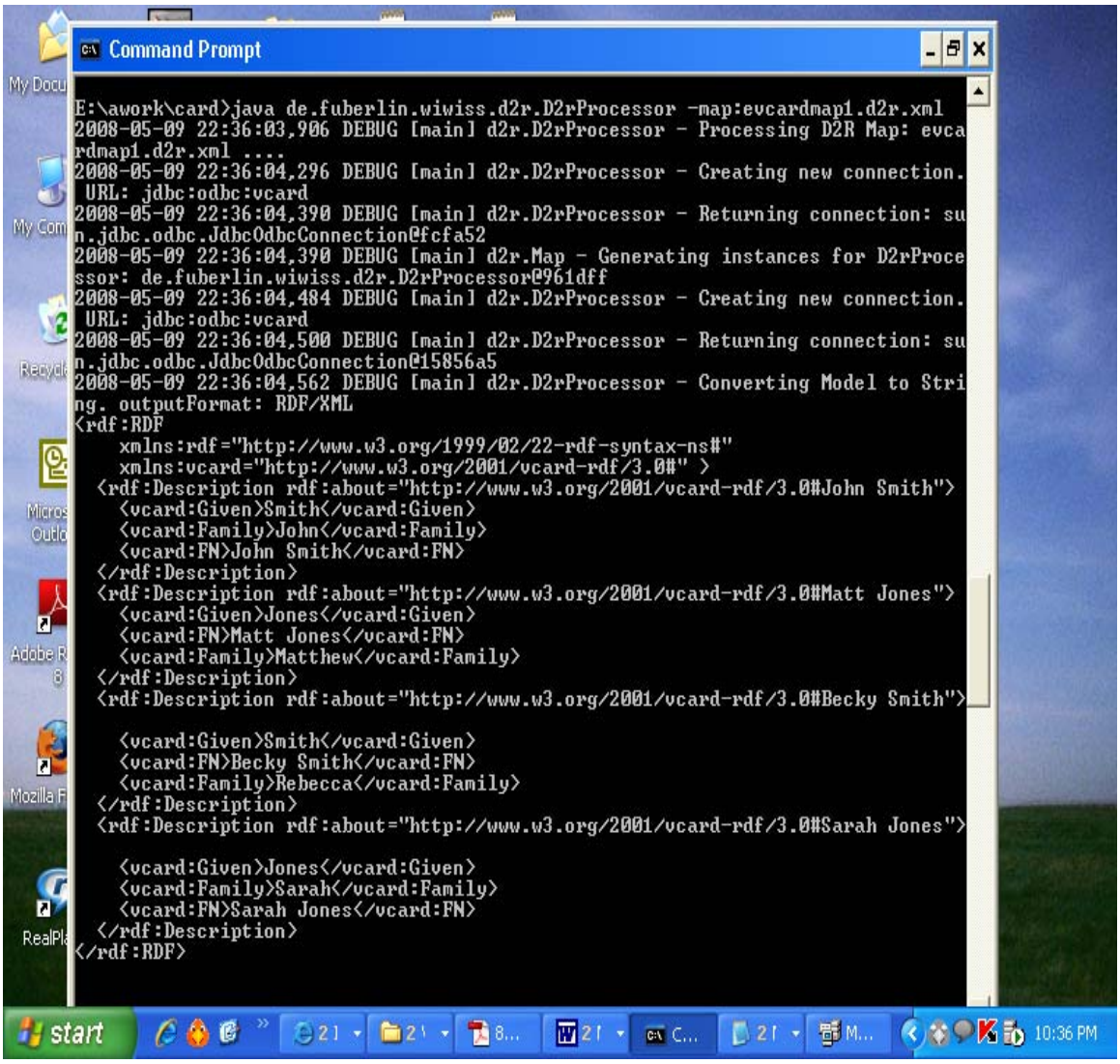

Figure 4.2: RDF output

4.2 Query RDF File using JENA

Jena rich API is used for manipulating RDF graphs. Jena is a Java class library, and is composed mainly of API (Application Programming Interface) and SPI (System Programming Interface. This API provides an interface for the Semantic Web application developer that makes it an ideal programming toolkit when we want to process the RDF file. In the RDF API, the key RDF package for the application developer is com.hp.hpl.jena.rdf.model. This package contains interfaces for representing models, resources, properties, literals, statements and all the other key concepts of RDF, and a ModelFactory for creating models.

Example: Query RDF file using JENA

query - Notepad

File Edit Format View Help

```
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.util.FileManager;
import com.hp.hpl.jena.vocabulary.*;

import java.io.*;

/** selecting the VCARD resources

 */
public class query extends Object {

    static final String inputFileName = "v1.rdf";

    public static void main (String args[]) {
        // create an empty model
        Model model = ModelFactory.createDefaultModel();

        // use the FileManager to find the input file
        InputStream in = FileManager.get().open(inputFileName);
        if (in == null) {
            throw new IllegalArgumentException( "File: " + inputFileName + " not found");
        }

        // read the RDF/XML file
        model.read( in, "");

        // select all the resources with a VCARD.FN property
        ResIterator iter = model.listSubjectsWithProperty(VCARD.FN);
        if (iter.hasNext()) {
            System.out.println("The database contains category for:");
            while (iter.hasNext()) {
                System.out.println("  " + iter.nextResource()
                                          .getRequiredProperty(VCARD.FN)
                                          .getString() );
            }
        } else {
            System.out.println("No vcards were found in the database");
        }
```

start 6 I... 3 M.. 805... PO... bin 3 N.. 11:46 PM

Figure 4.3: Jena querying RDF files

This program produce output similar to the following:

The database contains vcards for:

Sarah Jones

John Smith

Matt Jones

Becky Smith

4.3 Mapping between Relational Database tables in Semantic Web

In order to perform mapping between two tables in semantic web, first we create RDF files for tables using D2R mapping and then we map two RDF files in to one RDF file on the basis of primary key of their corresponding Relational database table.

PersonName	FirstName	LastName
John Smith	John	Smith
Sarah Jones	Sarah	Jones
Matt Jones	Matthew	Jones
Becky Smith	Rebecca	Smith

Table 4.2: vcardd

vcardd table RDF file:

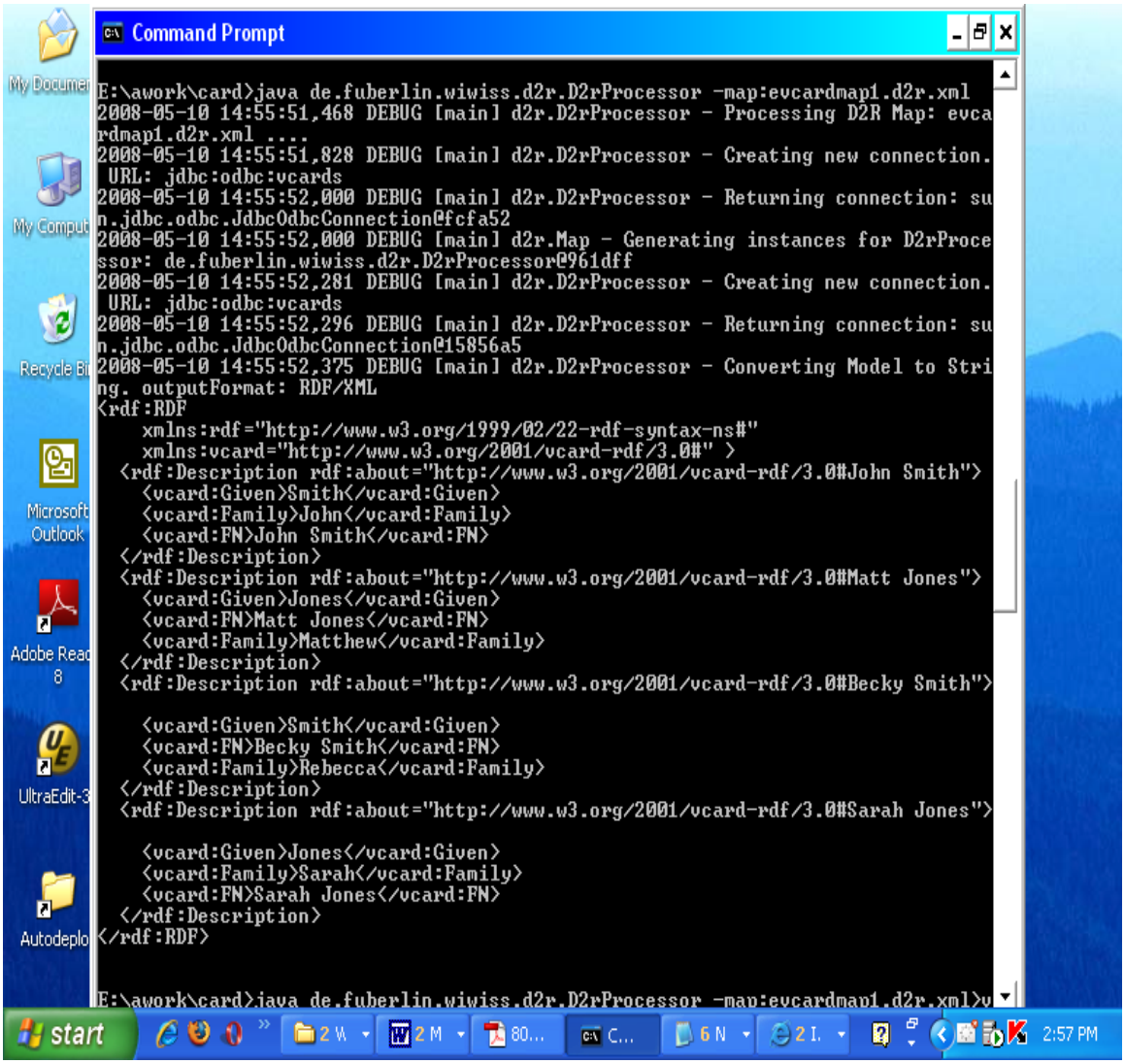

Figure 4.4: vcardd table RDF file

PersonName	AddressType	Address
John Smith	internet	john@yahoo.com
Sarah Jones	postal address	#320/7,ABC

Table 4.3: evcard

evcard table RDF file

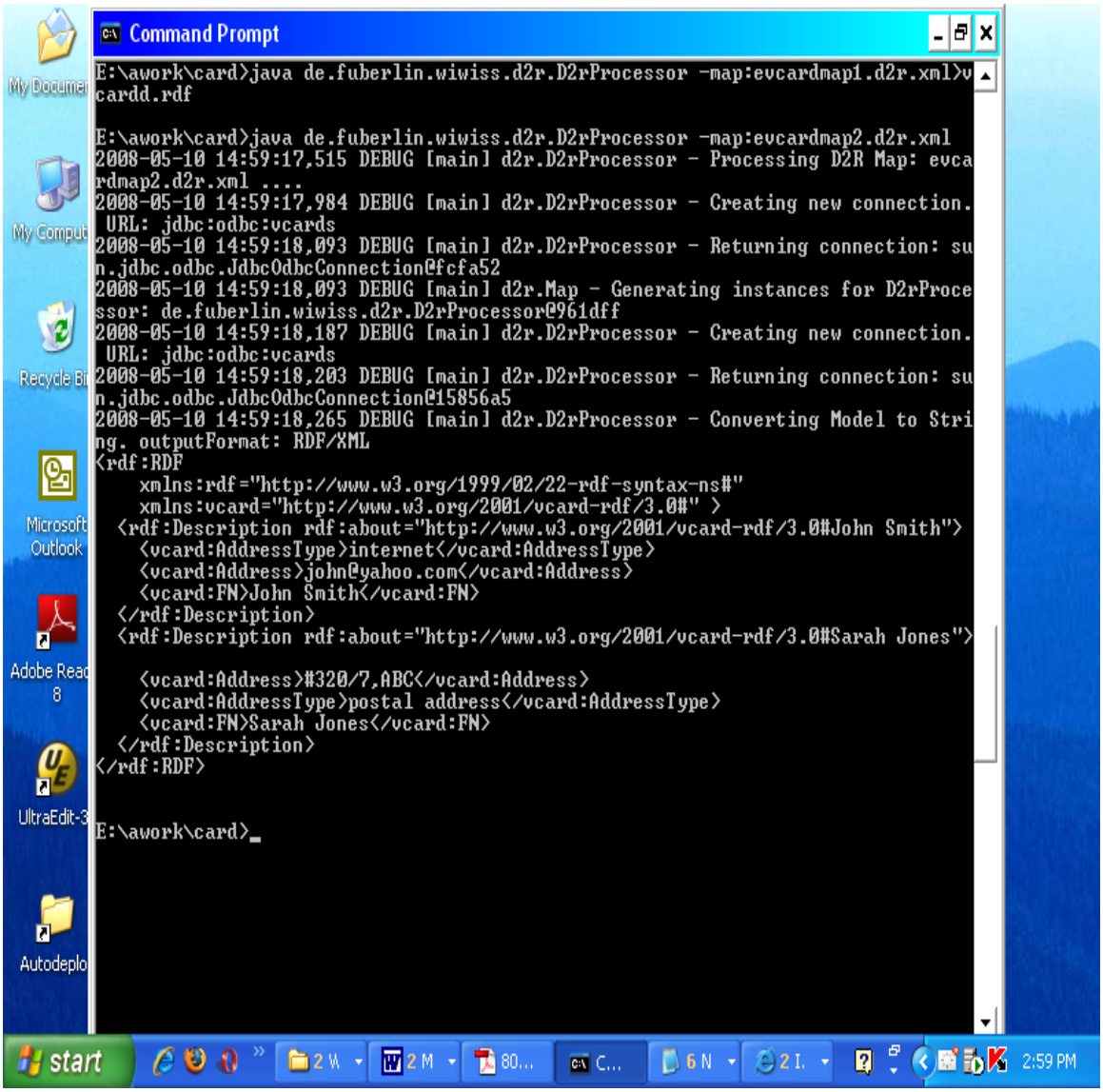

Figure 4.5: evcard table RDF file

Resulting mapping of two RDF file:

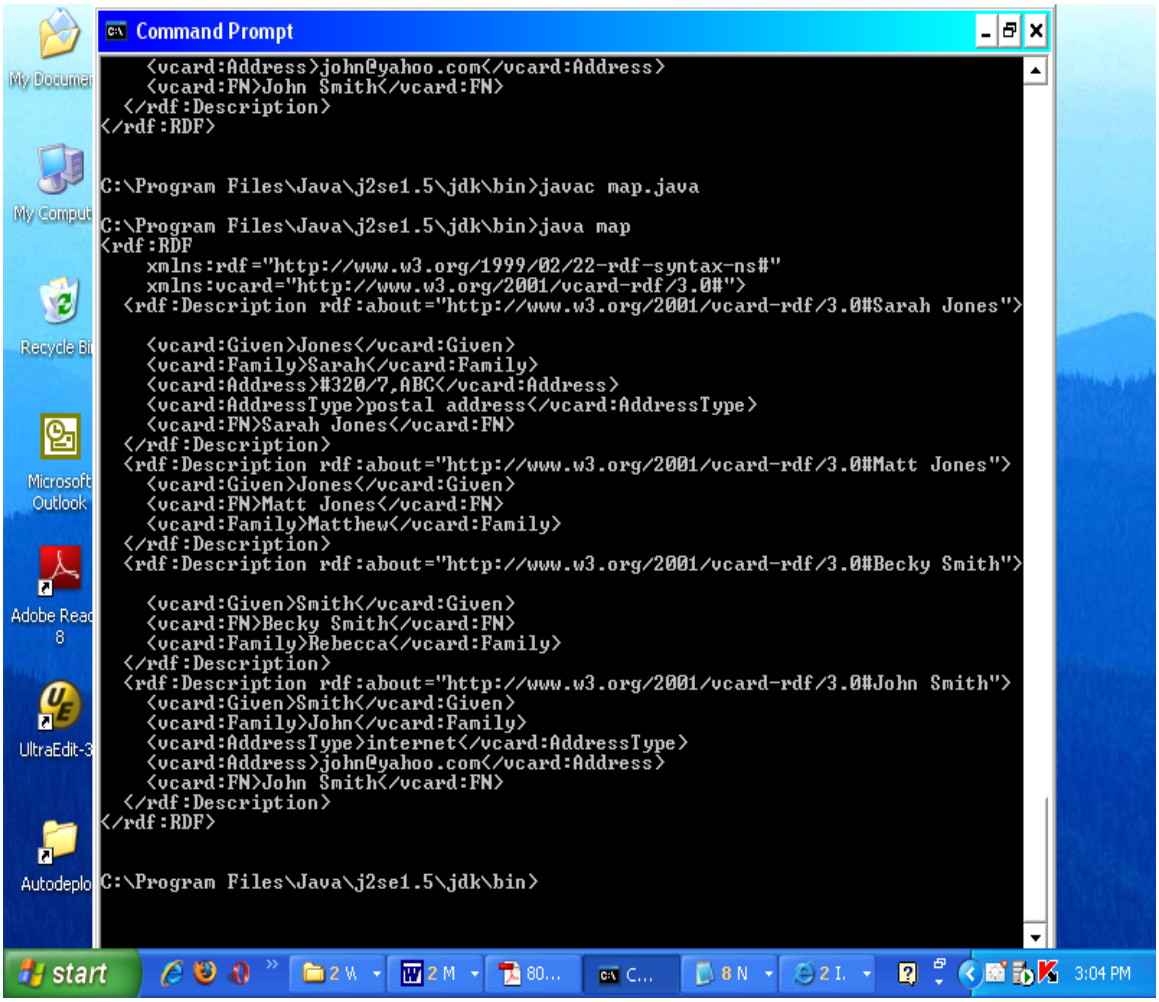

Figure 4.6: Mapped file

Chapter 5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

This work concludes that both D2R mapping and Jena API together enable semantic access to Relational database. It enables users to retrieve information solely by the semantic understanding of the applicable domain. Database schemas change over the time, even though the semantics of the data have not changed. In SQL, all these non-semantic changes imply that the existing pre-defined or pre-cached SQL query statements have to be modified accordingly. But in Semantic Web, it is not necessary to modify an application as long as there isn't any change in semantics.

Jena framework is an open source implementation of RDF, RDFS, and OWL (Web Ontology Language) and includes a rule-based inference engine. It provides a Java API for creating and manipulating RDF models. It is different from SQL in many ways. Although it is theoretically possible to query any RDF model with either language, but the use of JENA makes it easier to query RDF model because the JENA syntax makes virtually all join operations implicit, while SQL syntax usually makes them explicit. A consequence of this design decision is that the SQL expressions to answer typical questions will be much larger and somewhat more difficult to create as compared to be against RDF collections.

This work is based on the principle of the Software engineering, neglects the difference of the different ontology languages, sets from the RDF level of the ontology, and finally puts forward a more easy way to extract and process the RDF level ontology method which gives the corresponding implementation based on Jena. The method raised in this is easy, simple and useful and realizes the programmatic extracting and processing of the RDF and relational database.

5.2 Future Scope of Work

There is scope for extending the work further by query Relational Database in semantic web using Ontology as intermediate.

- D2R declarative language can be used to describe mappings between relational database schemas and OWL ontologies.
- JENA Ontology-level API will be used to manipulate Ontology.

The Ontology API contains classes that represent the different elements comprise ontology, such as OntModel, OntClass, OntProperty, Individual, Restriction (and its subclasses AllValuesFromRestriction, SomeValuesFromRestriction, MinCardinalityRestriction, etc), IntersectionClass, and UnionClass. The Jena2 ontology API is intended to support programmers who are working with ontology data based on RDF. Jena1 only provides the support for the DAML, but Jena2 provides support for OWL, DAML+OIL and RDFS with different class and interface.

REFERENCES

[1] T. Berners-Lee, "Weaving the Web", Harper, San Francisco, 1999.

[2] T. Berners-Lee, "Relational Databases on the Semantic Web", http://www.w3.org/DesignIssues/RDB-RDF.html.

[3] T. Berners-Lee. "Primer: Getting into RDF & Semantic Web using N3", 2004. http://www.w3.org/2000/10/swap/Primer.html.

[4] C. Bizer, "D2R MAP – A DB to RDF Mapping Language", 12th International World Wide Web Conference, Budapest, May 2003.

[5] Jeremy J. Carroll, Ian Dickinson, C. Dollin, Dave Reynolds, Andy Seaborne, and K. Wilkinson. "Jena: Implementing the semantic web recommendations." In Int. World Wide Web Conference, WWW'04 (Alternate track papers and posters), pages 74.83, 2004.

[6] J.Davidson, "Java API for XML Parsing", Version 1.0 Final Release. http://java.sun.com/aboutJava/communityprocess/final/jsr005/index.html.

[7] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann and I. Horrocks, "The Semantic Web: The Roles of XML and RDF", IEEE Internet Computing, Vol. 4(5), pp. 63-74, Sept/Oct 2000.

[8] James Hendler, Tim Berners-Lee, "Eric Miller. Integrating Applications on the Semantic Web.", Journal of the Institute of Electrical Engineers of Japan, Vol 122(10), October 2002, pp.676-680.

[9] I. Horrocks and J. Hendler, "The Semantic Web", in proceedings of First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002.

[10] A.J. Gerber, A. Barnard, and A.J. van der Merwe (South Africa), "Towards a Semantic Web Layered Architecture", From Proceeding (552), Software Engineering – 2007.

[11] F. Manola, E. Miller, "RDF Primer: W3C Proposed Recommendation", December, 2003. http://www.w3.org/TR/2003/PRrdfprimer-20031215.

[12] B. McBride, "Jena IEEE Internet Computing", July/August, 2002.

[13] Eric Miller, "An Introduction to the Resource Description Framework", D-Lib Magazine, May 1998.

[14] Wang Min, Ding Jianping, Xiang Yang, Xun Henxing, "The Research on the Jena-based Web Page Ontology Extracting andProcessing", Proceedings of the First International Conference on Semantics, Knowledge, and Grid.

[15] W. Nejdl, M. Wolpes, and C. Capelle, "The RDF schema" specification revisited. In Modellierung 2000.

[16] E. Prud'hommeaux and A. Seaborne. "SPARQL Query Language for RDF." http://www.w3.org/TR/2005/ WD-rdf-sparql-query-20051123/, 2005. W3C Working Draft.

[17] Joe Verzulli, "Using the Jena API to Process RDF" http://www.xml.com/pub/a/2001/05/23/jena.html.

[18] Erik Wilde, Katrina Rhoads Lindholm, "XML Foundations", September, 2006, http://dret.net/netdret/docs/wilde-elpub2006-xml.pdf.

[19] Jun Yuan, David H. Jones, "Enabling Semantic Access to Enterprise RDB Data" Position Paper for W3C workshop on RDF Access to Relational Databases.

[20] “From Data Federation Pyramid to the Semantic Web ‘Birthday Cake’”, http://www.mkbergman.com.

[21] “JENA – A Semantic Framework for Java”, http://jena.sourceforge.net/.

[22] “Resource Description Framework (RDF) Model and Syntax Specification’’, http://www.w3.org/TR/1999/REC-rdfsyntax- 19990222.

Appendix A
D2R MAPPING

```
<?xml version="1.0"?>
<d2r:Map xmlns:d2r="http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RMap/0.1#" >
        <d2r:ProcessorMessage  d2r:outputFormat="RDF/XML"/>
        <d2r:DBConnection d2r:odbcDSN="vcard" />
        <d2r:Namespace d2r:prefix="vcard"
        d2r:namespace="http://www.w3.org/2001/vcard-rdf/3.0#"/>
        <d2r:ClassMap
        d2r:sql="SELECT *  FROM vcardd;"
        d2r:groupBy="vcardd.PersonName"
        d2r:uriPattern="vcard:@@PersonName@@"  >
                <d2r:DatatypePropertyBridge d2r:property="vcard:FN"
                d2r:column="vcardd.PersonName" />
                <d2r:DatatypePropertyBridge d2r:property="vcard:Family"
                d2r:column="vcardd.FirstName" />
                <d2r:DatatypePropertyBridge d2r:property="vcard:Given"
                d2r:column="vcardd.LastName" />
        </d2r:ClassMap>
</d2r:Map>
```

RDF File:

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#" >
  <rdf:Description rdf:about="http://www.w3.org/2001/vcard-rdf/3.0#John Smith">
   <vcard:Given>Smith</vcard:Given>
   <vcard:Family>John</vcard:Family>
   <vcard:FN>John Smith</vcard:FN>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.w3.org/2001/vcard-rdf/3.0#Matt Jones">
```

```
    <vcard:Given>Jones</vcard:Given>
    <vcard:FN>Matt Jones</vcard:FN>
    <vcard:Family>Matthew</vcard:Family>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.w3.org/2001/vcard-rdf/3.0#Becky Smith">
    <vcard:Given>Smith</vcard:Given>
    <vcard:FN>Becky Smith</vcard:FN>
    <vcard:Family>Rebecca</vcard:Family>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.w3.org/2001/vcard-rdf/3.0#Sarah Jones">
    <vcard:Given>Jones</vcard:Given>
    <vcard:Family>Sarah</vcard:Family>
    <vcard:FN>Sarah Jones</vcard:FN>
  </rdf:Description>
</rdf:RDF>
```

Appendix B
JENA Query

```
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.util.FileManager;
import com.hp.hpl.jena.vocabulary.*;
import java.io.*;
/**  selecting the VCARD resources
 */
public class query extends Object {
     static final String inputFileName = "v1.rdf";
      public static void main (String args[]) {

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

      // use the FileManager to find the input file
      InputStream in = FileManager.get().open(inputFileName);
      if (in == null) {
                        throw new IllegalArgumentException( "File: " +
                        inputFileName + " not found");
                  }

      // read the RDF/XML file
      model.read( in, "");

      // select all the resources with a VCARD.FN property
      ResIterator iter = model.listSubjectsWithProperty(VCARD.FN);
      if (iter.hasNext()) {
         System.out.println("The database contains category for:");
         while (iter.hasNext()) {
            System.out.println("  " + iter.nextResource()
                              .getRequiredProperty(VCARD.FN)
```

```
                                    .getString() );
            }
        } else {
            System.out.println("No vcards were found in the database");
        }
    }
}
```

LIST OF PUBLICATIONS

- Ruby Sawhney and Shalini Batra, "*Querying relational database in Semantic Web*" at International Conference ICCDIT-2008, held at PCTE, Ludhiana (Punjab)**[Published]**.
- Ruby Sawhney and Shalini Batra, *"Querying Relational database in JENA"* at National Conference NCCN'08 held at SLIET, Longowal (Punjab) **[Presented & Published]**.
- Ruby Sawhney and Shalini Batra, "*From Relational data to RDF data*" in Apeejay Journal of Management and Technology at Apeejay Institute of Management, Jalandhar (Punjab)**[Communicated]**.