

An RDF-based Distributed Expert System

NAPAT PRAPAKORN*, SUPHAMIT CHITTAYASOTHORN**

Department of Computer Engineering
King Mongkut's Institute of Technology Ladkrabang
Faculty of Engineering, Bangkok 10520
THAILAND

Napat.Prapakorn@gmail.com*, suphamit@kmitl.ac.th**

ABSTRACT – An expert system or knowledge-based system comprises of a knowledge base and an inference engine in which their expertise knowledge is represented. The knowledge can be called upon when needed to solve a problem by the inference engine. In a large expert system, the knowledge base can be represented using frames. Hence they are called frame-based expert system or frame-based system. To avoid too many communication traffics during inferences, a distributed expert system, an expert system with an inference engine on its external knowledge base side is presented. It is an expert system which has an RDF external knowledge base for improved flexibility and mobility in knowledge sharing. This research presents a design and implementation of a Frame-based RDF expert system which has a RDF/XML database as its external knowledge base. The external knowledge base uses frame as its knowledge representation stored in RDF/XML format so that it can be placed on the WWW (World Wide Web) which is, ideally, accessible from anywhere. With this capability, an expert system will be enriched with flexibility and mobility in knowledge sharing.

Key-Words: - Frames, Expert system, Knowledge base, RDF, RDF/XML, Ontology

1 Introduction

Frames are widely used as the knowledge representation of large, complex expert systems [1]. Such expert system comprises internal knowledge base and inference engine in which their knowledge base is used to represent their expertise knowledge as frames. To avoid too many communication traffics during inferences, a database system is presented to store knowledge as an external knowledge base along with an inference engine of its own. With a tight coupling between an expert system and an external knowledge base, inferences can be performed on the external knowledge base then the inference result, and not only simple facts, are sent to a client expert system for further inferences [2]. In some distributed systems, mobile agents are used to reduce the communication problem [13]. However, the systems may not be very efficient in a resource-limited environment due to its lack of flexibility and mobility in knowledge sharing.

To improve flexibility and mobility, a Resource Description Framework (RDF) is presented. RDF is a standard for representing information about resources in the World Wide Web. RDF is intended for situations in which the information needs to be processed by application, rather than being displayed to people. Moreover, RDF also provides a

common framework for expressing information between applications without loss of meaning [3]. With RDF concepts, we could simply store our expertise knowledge on the WWW which is, ideally, accessible from anywhere. Thus, this research proposes a frame-based RDF expert system, an RDF-based distributed expert system, as an alternative expert system with flexibility and mobility, using a collection of RDF/XML databases as its knowledge base.

2 Resource Description Framework (RDF)

A Resource Description Framework (RDF) is a standard developed by W3C for representing information about resources in the World Wide Web. RDF uses a Uniform Resource Identifier Reference (URIref), a URI together with an optional fragment identifier at the end, as its mechanism for identifying resources. With the capability of URIref, not only network-accessible resources but things that are not network-accessible such as human beings, corporations or even abstract concepts that do not physically existed can also be identified by RDF.

A complete detail of RDF is available on the Website: <http://www.w3.org/RDF/>. In this section, we briefly review the RDF concepts.

2.1 RDF Statement

RDF is based on the idea that things being described have properties which have values. RDF describes things by making statements that specify those properties and values. RDF statement is divided into three parts; the part that identifies the thing being described is called the "Subject"; the part that identifies the property or characteristic of the subject is called the "Predicate"; the part that identifies the value of that property is called the "Object" [3]. For example:

`http://www.example.org/index.html` has a creator whose value is John Smith.

The statement can be divided into three parts:

"`http://www.example.org/index.html`" is the subject.
 "creator" is the predicate.

"John Smith" (who is identified by
`http://www.example.org/staffid/85740`) is the object.

2.2 RDF Conceptual Model

RDF models statements as nodes and arcs in a graph. In this notation, RDF statements are represented by:

- A node for the subject
- A node for the object
- An arc for the predicate, directed from the subject node to the object node

So the RDF statement above would be represented by the graph shown in Fig.1:

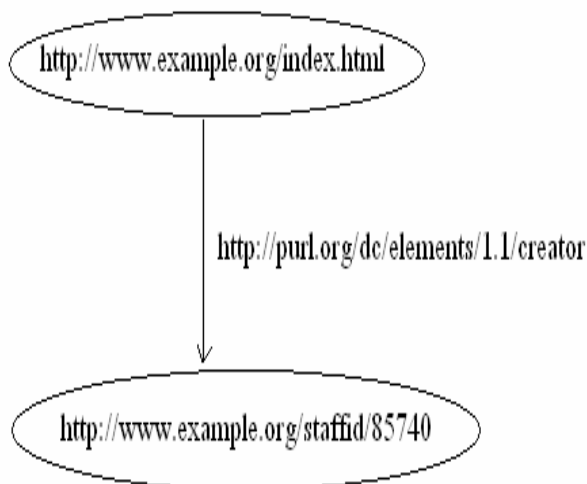


Fig.1 RDF graph [3]

Sometimes it is not convenient to draw graphs, so an alternative way of writing down the statements, called triples, is also used. In a triples notation, each statement in the graph is written as a simple triple of subject, predicate, object, respectively [3]. For example, the statement shown in Fig. 1 would be written in the tripples notation as:

```
<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/creator>
<http://www.example.org/staffid/85740>
```

2.3 An XML syntax for RDF: RDF/XML

RDF also provides an XML-based syntax, called RDF/XML, for recording and exchanging these graphs. Fig. 2 is an RDF in RDF/XML corresponding to the graph in Fig.1:

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:ex="http://www.example.org/"
3.     xmlns:dc=
4.         "http://purl.org/dc/elements/1.1"
5.         xmlns:exstaff=
6.         "http://www.example.org/staffid/" >
7.     <rdf:Description rdf:about=
8.         "http://www.example.org/index.html">
9.         <dc:creator>
10.            <rdf:Description>
11.                <exstaff:85740
12.                    rdf:resource=
13.                        "http://www.example.org/staffid/85740"/>
14.            </rdf:Description>
15.        </dc:creator>
16.    </rdf:Description>
17. </rdf:RDF>
```

(Line numbers are added to help in explaining the example.)

Fig. 2 RDF/XML Example

Line 1, `<?xml version="1.0"?>`, is the XML declaration, which indicates that the following content is XML, and the version of XML. Line 2 begins an `rdf:RDF` element. This indicates that the following XML content (starting here and ending with the `</rdf:RDF>` in line 12) is intended to represent RDF. Following the `rdf:RDF` on this same line is an XML namespace declaration, represented as an `xmlns` attribute of the `rdf:RDF` start-tag. This

declaration specifies that all tags in this content prefixed with `rdf:` are part of the namespace identified by the URIref `http://www.w3.org/1999/02/22-rdf-syntax-ns#`.

URIrefs beginning with the string `http://www.w3.org/1999/02/22-rdf-syntax-ns#` are used for terms from the RDF vocabulary. Line 3 and Line 4 specifies another XML namespace declaration. The ">" at the end of line 4 indicates the end of the `rdf:RDF` start-tag. Lines 1-4 are basically necessary to indicate that this is RDF/XML content, and to identify the namespaces being used within the RDF/XML content. Lines 5-11 provide the RDF/XML for the specific statement shown in Fig.1. An obvious way to talk about any RDF statement is to say it is a description, and that it is about the subject of the statement (in this case, about `http://www.example.org/index.html`), and this is the way RDF/XML represents the statement. 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) using the `rdf:about` attribute to specify the URIref of the subject resource. Line 6 provides a property element, with `dc:creator` as its tag, to represent the predicate and object of the statement. The content of this property element is the object of the statement, with `ex:staff:85740` as its tag using the `rdf:resource` attribute to specify the URIref of the object resource with no further sub properties. The property element is nested within the containing `rdf:Description` element, indicating that this property applies to the resource specified in the `rdf:about` attribute of the `rdf:Description` element. Line 11 indicates the end of the first `rdf:Description` element. Finally, Line 12 indicates the end of the `rdf:RDF` element started on line 2.

2.4 RDF Schema

RDF provides a way to express simple statements about resources, using named properties and values. However, RDF user communities also need the ability to define the vocabularies (terms) they intend to use in those statements, specifically, to indicate that they are describing specific kinds or classes of resources, and will use specific properties in describing those resources. For example, some applications might need to describe classes such as `ex:Person` and `ex:Company`, and properties such as `ex:age`, `ex:jobTitle`, and `ex:numberOfEmployees`. RDF itself cannot define such application-specific classes and properties. Instead, such classes and properties are described as an RDF vocabulary,

using extensions to RDF provided by the RDF Vocabulary Description Language, called RDF Schema [3]. The basic concept of RDF Schema and its property are illustrated in Fig. 3. More details of each element are described in Appendix A and Appendix B in Appendix section.

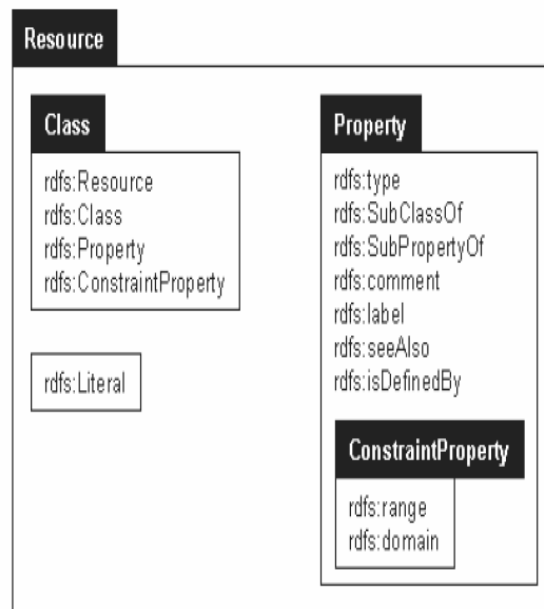


Fig. 3 Concept of RDF Schema

3 SPARQL Query Language for RDF

With an RDF approach, W3C also propose SPARQL as a standard query language for RDF. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF documents or viewed as RDF documents via a middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs and also can be serialized into XML format.

Most forms of SPARQL query contain a set of triple patterns called a basic graph pattern. Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable. A basic graph pattern matches a subgraph of the RDF data when RDF terms from that subgraph may be substituted for the variables and the result is RDF graph equivalent to the subgraph [11].

In this section, we briefly illustrate the SPARQL basics using the data source in Fig. 4. A complete detail of SPARQL Query Language for RDF can be

found on the Website: <http://www.w3.org/TR/rdf-sparql-query/>.

@prefix dc: <<http://purl.org/dc/elements/1.1/>>

@prefix book: <<http://example.org/book/>>

@prefix writer: <<http://example.org/writer/>>

book:book1	dc:title	"Harry Potter and the Deadly Hallows"
book:book2	dc:title	"The Lord of the Ring"
writer:ID001	dc:name	"J.K. Rowling"
writer:ID002	dc:name	"J.R.R. Tolkien"
book:book1	dc:auther	writer:ID001
book:book2	dc:auther	writer:ID002

Fig. 4 Data Source used for illustrating SPARQL basics

3.1 SPARQL "SELECT" Query

The SELECT query will result in variables binding results. The SELECT query is mainly used to find values of declared variables that matches RDF graphs in WHERE clause. An example of SELECT query will be illustrated in Fig. 5 and its result will be illustrated in Fig. 6 which can be serialized into XML format as illustrates in Fig. 7.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?myTitle

WHERE
{
    ?anyBook    dc:title    ?myTitle
}
```

Fig. 5 SPARQL "SELECT" Query Example

myTitle
"Harry Potter and the Deadly Hallows"
"The Lord of the Ring"

Fig. 6 "SELECT" Query's result

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3c.org/2005/sparql-
results#">
  <head>
    <variable name="myTitle"/>
  </head>
  <results>
    <result>
      <binding name="myTitle">
        <literal>Harry Potter and the Deadly
          Hollows</literal>
      </binding>
    </result>
    <result>
      <binding name="myTitle">
        <literal>The Lord of the Ring</literal>
      </binding>
    </result>
  </results>
</sparql>
```

Fig. 7 "SELECT" Query result in XML Format

3.2 SPARQL "CONSTRUCT" Query

The CONSTRUCT query will result in RDF graphs. The CONSTRUCT query is mainly use to build graph based on a template which is used to generate RDF triples based on the results of matching the graph pattern in WHERE clause. An example of CONSTRUCT query will be illustrated in Fig.8 and its result will be illustrated in Fig. 9 which can also be serialized into XML format as illustrates in Fig. 10.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>

CONSTRUCT ?myAuther dc:write ?myTitle

WHERE
{
    ?myBook    dc:title    ?myTitle
    ?myBook    dc:auther   ?myAuther
}
```

Fig. 8 SPARQL "CONSTRUCT" Query Example

@prefix dc: <http://purl.org/dc/elements/1.1/>

@prefix writer: <http://example.org/writer/>

writer.ID001	dc:write	"Harry Potter and the Deadly Hallows"
writer.ID002	dc:write	"The Lord of the Ring"

Fig. 9 SPARQL "CONSTRUCT" Query's result

```
<rdf:RDF xmlns:rdf=
"http://www.w3c.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc=
"http://purl.org/dc/elements/1.1/"
    xmlns:writer=
"http://example.org/writer/">
  <rdf:Description rdf:about=
"http://example.org/writer/ID001">
    <dc:write>Harry Potter and the Deadly
Hollows</dc:write>
  </rdf:Description>
  <rdf:Description rdf:about=
"http://example.org/writer/ID002">
    <dc:write>The Lord of the
Ring</dc:write>
  </rdf:Description>
</rdf:RDF>
```

Fig. 10 "CONSTRUCT" Query result in XML Format

3.3 SPARQL "ASK" Query

The ASK query will result in a boolean value which can also be serialized into XML format. The main purpose of this type of query is to test whether or not a graph that match a specified graph pattern exists. An example of ASK query will be illustrated in Fig.11 and its result in XML format will be illustrated in Fig. 12

PREFIX dc: <http://purl.org/dc/elements/1.1/>

ASK { ?anyBook dc:title "The Lord of the Ring" }

Fig. 11 SPARQL "ASK" Query Example

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3c.org/2005/sparql-results#">
  <head></head>
  <results>
    <boolean>true</boolean>
  </results>
</sparql>
```

Fig. 12 "ASK" Query result in XML Format

4 SPARQL Protocol for RDF

The SPARQL Protocol for RDF uses Web Service Description Language 2.0 (WSDL2) to describe how to convey SPARQL queries to a SPARQL query processing service and returning the query results to the entity that requested them. SPARQL Protocol can be described in two ways: first, as an abstract interface independent of any concrete realization, implementation, or binding to another protocol; second, as HTTP and SOAP bindings of this interface.

The proposed system architecture will implement this protocol using HTTP Binding technique. SPARQL Protocol contains one interface, SparqlQuery, which in turn contains one operation, query. SPARQL Protocol is described abstractly with WSDL2 in terms of a web service that implements its interface, types, faults, and operations, as well as by HTTP bindings [11].

5 Frames

Frame was introduced by Marvin Minsky in 1974 [4]. It is a knowledge representation which has both data structure and inference capability. Frame is suitable for representing knowledge with concepts and classifications or a taxonomy hierarchy [5], [6].

A frame comprises a frame name, slots or attributes of the frame and facets [7], [8]. For clarification, frames can be classified into class frames and instance frames. Class frames are used to describe groups of objects or classes of objects and can also be organized into taxonomy. A class frame therefore has parent and children as common slots. Slots from a parent frame can be inherited to its children. Instance frames, on the other hand, describe particular object instances. They are the leaf node of the taxonomy and have no children [2], [6]-[10].

Facets are used to control slot values and corresponding operations. It can be used to establish initial slot value, slot data type, possible value range and next activity to be performed. Validation rules, trigger operations and derivation rules are common facet as well [2].

With the RDF concept of describing things by identifying its properties and properties' value, frames can simply be represented on the World Wide Web by RDF/XML.

6 System Architecture

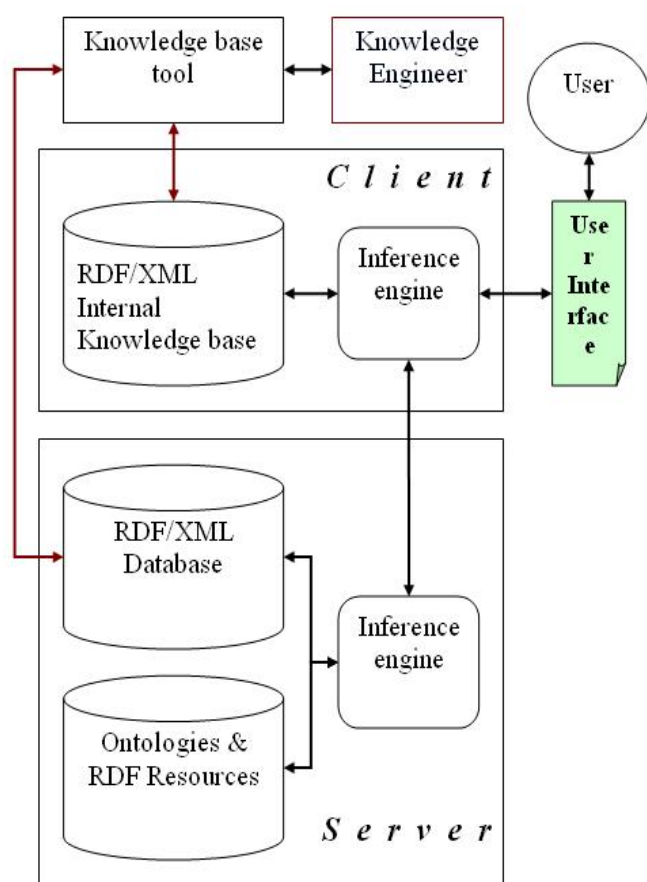


Fig.13 FRDFX System Architecture

Fig.13 shows the architecture of the FRDFX (Frame-based Resource Description Framework Expert System). This architecture allows the knowledge base side (server side) to represent their expertise knowledge with frames and simply share them on the World Wide Web as desired. Moreover, both of the knowledge base side and the expert system side (the client side) has their own inference engine to avoid a lot of communication traffic during inferences. The frames on the expert system side are those which involve user's interaction and acquire fact. Inference on this side can lead to a reference to the facts on the external knowledge base which can be inferred from other frames on server side [2].

The client-sided expert system comprises user interfaces for user and expert, the internal knowledge base and the client-sided inference engine. The external knowledge server comprises the server-sided inference engine, the RDF/XML external knowledge base and also ontologies and other RDF resources.

The main purpose of client-sided inference is to gather current information on the subject matter by gathering facts from the user during a consultation session and inference. The inference rule in the first frame will choose the most suitable next frame to go to. Several frames may be visited on this client side before the client-sided expert system gathers enough information which is to be sent to the knowledge base system on the server side for further inferences [2] using an implemented web service technology [14] including a SPARQL Protocol for RDF along with a SPARQL Query Language for RDF [11], [12].

On the other hand, server-sided inference mainly focuses on using existing facts which are already stored in the knowledge base and other relevant knowledge from ontologies and other RDF resources. The result of this server-sided inference will be sent back to the client-sided expert system and finally to the user [2].

7 Implementation

Frames on the knowledge base are kept in RDF/XML documents. Fig.14 shows RDF graph which representing frame's infrastructure.

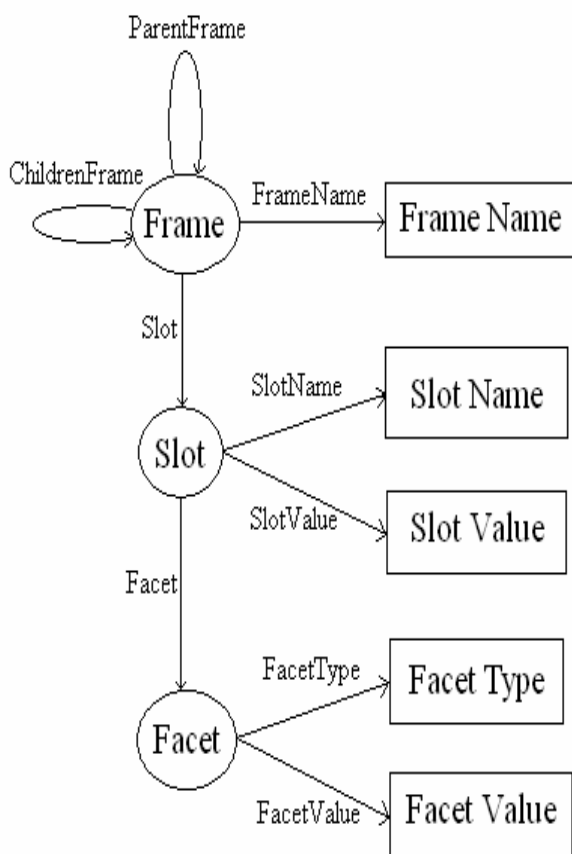


Fig.14 RDF graph representing Frame

From RDF graph representing Frame in Fig.14, we can create RDF/XML which represents a prototype frame as shown in Appendix C.

8 Conclusions

This paper presents an alternative architecture of frame-based knowledge base and expert system with improved flexibility and mobility. Storing frames in RDF/XML format provides the capability of knowledge sharing on the World Wide Web which is, ideally, accessible from anywhere. The architecture uses web service technology implementing a SPARQL Protocol for RDF and SPARQL Query Language for RDF to communicate between servers and clients. For further research, RDF/XML knowledge base can be migrated to mobile server to improve even more flexibility and mobility in knowledge sharing.

References:

- [1] Peter D. Karp, *The Design Space of Frame Knowledge Representation System.*, SRI AI Center Technical, Note#520, 1993.
- [2] Suphamit Chittayasothorn and Chuleerat Rattanaprathet, *Expert Database System Architecture and Implementation on Object Relational Databases*, WSEAS Transactions on Computers Issue 3, Vol.5, March 2006.
- [3] *RDF Primer*, W3C Recommendation 10 February 2004
- [4] Marvin Minsky, *A Framework for Representing Knowledge*, Reprinted in *The Psychology of Computer Vision*, P. Winston (Ed.), McGrawHill, 1975.
- [5] Negnevitsky, M., *Artificial Intelligence: A Guide to Intelligent Systems*, Addison Wesley, Harlow, England, 2002
- [6] Richard Fikes and Tom Kehler, *The Role of Frame-based representation in reasoning*, *Communications of the ACM*, 28(9), 1985
- [7] Natalya F. Noy, Mark A. Musen, Jose L. V. Mejjino, Cornelius Rosses, *Pushing the Envelope: Challenges in Frame-Based Representation of Human Anatomy*, *Data & Knowledge Engineering*, Volume 48 Issue 3 (ACM), March 2004
- [8] Durkin J., *Expert Systems: Design and Development*, Macmillan Inc., 1994.
- [9] Benjamin Kuipers, *Algernon for expert system*, Draft document in Computer Science Department University of Texas at Austin, 18 January 1994.
- [10] Kamran Pasaye, *Expert systems for experts*, John Wiley & Sons, Inc., 1988.
- [11] *SPARQL Protocol for RDF*, W3C Recommendation 15 January 2008
- [12] *SPARQL Query Language for RDF*, W3C Recommendation 15 January 2008
- [13] Nutlada Rattanavijai and Suphamit Chittayasothorn, *Mobile Agents Based Distributed Database Statistics Collection*, WSEAS Transactions on Computers, Vol.6, June 2007.
- [14] Saeed Araban and Leon Sterling, *Quality of service for web services*, WSEAS Transactions on Computers, Vol. 3, October 2004

Appendix A: RDF classes

<u>Class name</u>	<u>comment</u>
rdfs:Resource	The class resource, everything.
rdfs:Literal	The class of literal values, e.g. textual strings and integers.
rdf:XMLLiteral	The class of XML literals values.
rdfs:Class	The class of classes.
rdf:Property	The class of RDF properties.
rdfs:Datatype	The class of RDF datatypes.
rdf:Statement	The class of RDF statements.
rdf:Bag	The class of unordered containers.
rdf:Seq	The class of ordered containers.
rdf:Alt	The class of containers of alternatives.
rdfs:Container	The class of RDF containers.
rdfs:ContainerMembershipProperty	The class of container membership properties, rdf:_1, rdf:_2, ..., all of which are sub-properties of 'member'.
rdf:List	The class of RDF Lists.

Appendix B: RDF properties

<u>Property name</u>	<u>comment</u>	<u>domain</u>	<u>range</u>
rdf:type	The subject is an instance of a class.	rdfs:Resource	rdfs:Class
rdfs:subClassOf	The subject is a subclass of a class.	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	The subject is a subproperty of a property.	rdf:Property	rdf:Property
rdfs:domain	A domain of the subject property.	rdf:Property	rdfs:Class
rdfs:range	A range of the subject property.	rdf:Property	rdfs:Class
rdfs:label	A human-readable name for the subject.	rdfs:Resource	rdfs:Literal
rdfs:comment	A description of the subject resource.	rdfs:Resource	rdfs:Literal
rdfs:member	A member of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:first	The first item in the subject RDF list.	rdf:List	rdfs:Resource
rdf:rest	The rest of the subject RDF list after the first item.	rdf:List	rdf:List
rdfs:seeAlso	Further information about the subject resource.	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	The definition of the subject resource.	rdfs:Resource	rdfs:Resource
rdf:value	Idiomatic property used for structured values.	rdfs:Resource	rdfs:Resource
rdf:subject	The subject of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:predicate	The predicate of the subject RDF statement.	rdf:Statement	rdfs:Resource
rdf:object	The object of the subject RDF statement.	rdf:Statement	rdfs:Resource

Appendix C: RDF/XML Frame Representation

```

<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
  <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
  <!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1233987595.owl#"
  xml:base="http://www.owl-ontologies.com/Ontology1233987595.owl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
  xmlns:swrl="http://www.w3.org/2003/11/swrl#"
  xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
  xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Ontology rdf:about="Frame"/>
  <owl:Class rdf:ID="Facets">
    <rdfs:subClassOf rdf:resource="#owl;Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#FacetValue"/>
        <owl:cardinality rdf:datatype="#xsd:int">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:DatatypeProperty rdf:ID="FacetType">
    <rdfs:domain rdf:resource="#Facets"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="FacetValue">
    <rdfs:domain rdf:resource="#Facets"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="FrameName">
    <rdfs:domain rdf:resource="#Frames"/>
    <rdfs:range rdf:resource="#xsd:string"/>
  </owl:DatatypeProperty>
  <owl:Class rdf:ID="Frames">
    <rdfs:subClassOf>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChildren"/>
            <owl:someValuesFrom rdf:resource="#Frames"/>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasParent"/>
            <owl:someValuesFrom rdf:resource="#Frames"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:Class>
    </rdfs:subClassOf>
  </owl:Class>

```

```

    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasSlot"/>
      <owl:someValuesFrom rdf:resource="#Slots"/>
    </owl:Restriction>
  </owl:unionOf>
</owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#FrameName"/>
    <owl:cardinality rdf:datatype="&xsd:int">1</owl:cardinality>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasChildren">
  <rdfs:domain rdf:resource="#Frames"/>
  <owl:inverseOf rdf:resource="#hasParent"/>
  <rdfs:range rdf:resource="#Frames"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasFacet">
  <rdfs:domain rdf:resource="#Slots"/>
  <owl:inverseOf rdf:resource="#isFacetOf"/>
  <rdfs:range rdf:resource="#Facets"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasParent">
  <rdfs:domain rdf:resource="#Frames"/>
  <owl:inverseOf rdf:resource="#hasChildren"/>
  <rdfs:range rdf:resource="#Frames"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasSlot">
  <rdfs:domain rdf:resource="#Frames"/>
  <owl:inverseOf rdf:resource="#isInFrame"/>
  <rdfs:range rdf:resource="#Slots"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isFacetOf">
  <rdfs:domain rdf:resource="#Facets"/>
  <owl:inverseOf rdf:resource="#hasFacet"/>
  <rdfs:range rdf:resource="#Slots"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isInFrame">
  <rdfs:domain rdf:resource="#Slots"/>
  <owl:inverseOf rdf:resource="#hasSlot"/>
  <rdfs:range rdf:resource="#Frames"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="SlotName">
  <rdfs:domain rdf:resource="#Slots"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="Slots">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#SlotValue"/>
      <owl:maxCardinality
rdf:datatype="&xsd:int">1</owl:maxCardinality>

```

```

    </owl:Restriction>
  </rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#SlotName"/>
    <owl:cardinality rdf:datatype="&xsd:int">1</owl:cardinality>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf
rdf:resource="&owl;Thing"/>
</owl:Class>
<owl:DatatypeProperty rdf:ID="SlotValue">
  <rdfs:domain rdf:resource="#Slots"/>
</owl:DatatypeProperty>
</rdf:RDF>

```