RDF Objects

A. Barnell

Semantic Web Applications Group, Hewlett Packard Laboratories
Bristol, Avon, England

The Semantic Web is growing in both size and complexity. The number of RDF vocabularies and data sources are both increasing. This paper shows how large RDF databases can be accessed in smaller and more manageable chunks called RDF Objects. RDF Objects encapsulate compound data structures to give applications a more granular view of a database. Applications can control the size and structure of Objects by specifying the Object extraction algorithm and filtering the returned data by vocabulary. Links between Objects, analogous to hypertext links, allow applications to navigate across distributed data sources and leave link-trails for other applications to follow. Information can be aggregated from multiple data sources by recursively following Object links. Resources without URIs in different databases are cross-matched using DAML and OWL ontologies. An HTTP implementation of an RDF Object Server has been developed and used to develop tools for W3C Working Groups.

1. CONTEXT AND BACKGROUND

1.1 RDF and the Semantic Web

The Semantic Web is the part of the Internet that machines can understand. Documents on the Semantic Web are written in machine-readable languages such as RDF [1], the Resource Description Framework. A "machine-readable" document can be understood by a machine without ambiguity. In contrast the World Wide Web is not understood by machines because it is written in languages that machines don't understand, like English. RDF is the core language of the Semantic Web. RDF graphs represent collections of statements. Nodes in an RDF graph represent things, and arrows between nodes represent the relationships between things. Things and relationships can be labelled with URIs [20]. RDF's original purpose was to allow metadata to be attached to web pages, but it is powerful enough to describe complex real world entities like people and organisations. RDF is even flexible enough to be used as the foundation for a number of more powerful languages such as DAML+OIL [2] and its successor-to-be, OWL [3], the Web Ontology Language.

1.2 The World Wide Web Consortium

The W3C [4] is a forum for the development of Internet technologies. Initially created to promote compatibility amongst Web browsers, the Consortium now concentrates on more advanced technologies such as the Semantic Web and Web Services. The Consortium is made up of a number of Working Groups, each tackling the issues involved with a particular technology. A Working Group's goal is to produce specifications that allow independent developers to create implementations of a technology that are compatible with each other. For example, the RDFCore Working Group [5] develops specifications for the RDF language. Working Groups use Internet mailing lists as their main method of communication. The lists are used for general discussion and to report new developments. For example, RDFCore members use the RDFCore Mailing List [6] to inform the group of completed actions. Groups hold regular teleconferences in which formal decisions are made and actions are assigned to each member of the group. The W3C process is formally described in the W3C Process Document [7].

1.3 Semantic Web Applications at Hewlett Packard

The Semantic Web research team at HP Labs, Bristol [8] is actively developing Jena [9], a Java API for manipulating RDF. Jena provides a rich Model interface giving a set of resource-centric operations on RDF data. Some members of the Jena team are also members of RDFCore, the W3C Working Group for RDF. The Sweb-Apps team at Hewlett Packard was formed in April 2002 to investigate the issues involved in developing Semantic Web Applications using the Jena Toolkit:

- What does it mean to be a Semantic Web Application?
- What tools do developers need to create Semantic Web Applications?
- Are there any recurring patterns involved in designing Semantic Web Applications?
- To what extent does Jena meet the needs of the Semantic Web application developer?

To address these issues the Sweb-Apps team have been developing a suite of Semantic Web tools to be used by W3C Working Groups to automate and assist the Working Group process.

1.4 Tools for Working Groups

After discussions with members of the RDFCore and WebOnt Working Groups, the Sweb-Apps team identified three key Semantic Web Tools that could be useful to Working Groups.

Action and Issue Manager - Actions are created at meetings and assigned to members of the group. Actions remain active until they are either completed or discontinued. Issues have a similar lifecycle but are resolved by the entire group rather than an individual. The Action and Issue lifecycles can be modelled as finite state machines . An Action and Issue manager could use RDF to describe and link actions and issues to the the documents, people, places and technologies that they refer to.

Teleconference Meeting Assistant - Working Groups use IRC (Internet Relay Chat) during teleconferences to log the decisions made in the meeting. A Meeting Assistant could use an RDF-based agenda to guide the group through the meeting, record the decisions made and automatically produce the minutes of the meeting. The assistant could use the Action and Issue Manager to automatically include recently completed actions and unresolved issues in the agenda for the next meeting.

Mailing List Assistant - Working Groups use mailing lists to discuss issues and for administrative purposes. For example, a member can inform the group that they cannot attend the next meeting or that they have completed an action. A mailing list assistant could detect these messages and use the Action Manager to change an action or use the Meeting Assistant to change the attendance list in the next agenda.

FIG 1: An Action represented using RDF

Working Groups, people and actions can all be described using compound data structures in RDF. Figure 1 shows an action described using RDF. The action is represented by the highlighted resource and the action details are

given by the rest of the graph structure: whom it is assigned to, that person's email address, the ID of the action, and so on. It is the *entire* collection of statements in the graph that describe the action, not just the action resource's immediate properties. Many other entities are also described in RDF using compound data structures, for example VCards [10], RSS News Feeds [11], and vCal Calendar information [12]. RDF Objects encompass this concept of compound data structures, enabling them to be extracted from databases and treated as atomic entities using an API on top of Jena.

2. RDF DATABASES

Although there is a single specification for the RDF language [13], there is no such standardisation for methods of accessing remote RDF databases. A number of proposals have been made by developers, each with different query languages, client-server protocols and client APIs. This section describes Joseki [14] and Jena SQL Models [15] and investigates the issues involved in database access from the point of view of the application.

2.1 Use Cases

All Semantic Web Applications can be split into two categories:

1. Applications that access RDF from a single database
2. Applications that access RDF from multiple databases

In the first case, all the RDF that the application uses is from a single graph. An example would be a Personal Information Manager that uses RDF to store user bookmarks in a single database.

In the second case, applications will often need to merge data from different databases. For example, a workflow application might want to find the contact details (phone, email etc) of a Project Manager. It could achieve this by combining information about the project from the project database with information about the manager from the employee database. Without the ability to combine information from different sources, the Semantic Web is no longer a Web but a collection of discrete data sources. Therefore it is important to make using multiple databases a simple task for the application, whilst still giving the application control over how the data is combined.

2.2 Existing RDF Database Systems

2.2.1 Joseki

Joseki [14] is an RDF database server that allows databases to be queried using the RDQL query language [18]. The Joseki server is written in java using Jena and allows the databases to be stored in SQL servers. There is a java client API that works with Jena. Clients send requests to the Joseki server using HTTP GET. RDQL queries are graph patterns with any number of variables. The query below finds all the actions in a database, the people assigned to those actions, and the email addresses of those people. A question mark in the query indicates that the term is a variable.

```
SELECT ?actionID, ?person, ?email WHERE
(?action, <rdf:type>, <action:Action>),
(?action, <action:ID>, ?actionID),
(?action, <action:assignedTo> ?person),
(?person, <rdf:type>, <person:Person>),
(?person, <person:email>, ?email)
```

The result is a table of variable bindings with the columns being the variables and the rows being the results.

?actionID	?person	?email
"REVIEW-DATATYPES-DOCUMENT"	(blank)	<mailto:brian@w3.org>
"ATTEND-MEETING"	(blank)	<mailto:jeremy@w3.org>
"ATTEND-MEETING"	(blank)	<mailto:patrick@w3.org>

In this particular result set all the bindings of ?actionID are literals, all the bindings of ?person are blank nodes and all the bindings of ?email are named resources (URIs). Clients can make changes to the database by adding and removing sets of statements. The Joseki client-server protocol uses internal blank node identifiers to allow a client to add or remove a statement involving a particular blank node.

2.2.2 Jena SQL Models

The Jena toolkit comes with relational database classes [15] that allows RDF to be stored in SQL databases and accessed through the Model interface. This allows remote RDF to be accessed with the same API as local RDF data as long as remote JDBC connections can be made. Using remote SQL models involved three types of overhead:

- Initial SQL connection overhead
- At least one remote JDBC call is needed for every Jena function call
- On the SQL server, the database needs to be accessed for every triple operation

Other RDF Databases include Sesame [16] and rdftp [17]. Sesame allows databases to be queried using the RQL query language, which has explicit support for RDFS schemas [21], but not DAML or OWL ontologies. rdftp uses a relational database to store RDF and allows databases to be queried using property-value pairs.

2.3 Issues in RDF Database Design

Many real-world entities like people and Working Groups do not have URIs and are represented in RDF using blank nodes. Instead of being identified by a URI they can be identified by their unambiguous properties. These properties can be determined from the relevant OWL and DAML ontologies. An application wanting to query a database for information about a person will need to identify that person by their unambiguous properties, for example their email address or social security number. This is problematic when using RDQL because there might be many such properties that could be used and the application may not know in advance which ones are actually used in the database. And if the application provides too much identifying information the entire query might fail because RDQL does not allow partial matches. RDF Objects have been designed to allow clients to query databases without knowing precisely how resources are identified in the database. This is achieved by both the client and server making use of DAML and OWL ontologies to dynamically determine the identifying properties of a resource at run-time and allowing for clients to over-identify the resource. This is comparable to the "Reference by Description" of TAP [19]. Whereas with Joseki and SQL Models it is the responsibility of the application to identify resources, with RDF Objects the client library and server do the identification automatically, simplifying the development of applications.

3. REQUIREMENTS FOR RDF OBJECTS

3.1 RDF Objects

An RDF Object is a compound data structure extracted from a database. Different Objects can be obtained by specifying the Object extraction algorithm and its parameters. In particular it should be possible to extract an Object centred around a particular resource in the database, for example an action or a person. It should be possible to discover more information about a resource in an RDF Object either from the database the Object was extracted from or a different database. It should be possible to search a database and receive the results as an RDF Object.

3.2 Defining and Describing RDF Objects

RDF Objects should be defined and described using RDF. Object Definitions should provide enough information for an agent to retrieve the correct Object from the correct Object Server. Encoding the definitions in RDF allows Object Descriptions to be embedded in Object data, allowing links between Objects (see below).

3.3 RDF Object Server

An RDF Object Server provides access to many RDF Object Databases. The Server should be able to accept an RDF Object definition from a client, extract the Object from the specified database and return the Object to the client. The Server should be able to store data in SQL databases and on the local file system. The Server should have a modular design allowing additional storage modules and extraction algorithms to be added.

3.4 Editing and Updating RDF Objects

Client Applications should use the Jena Model API to manipulate the local cache of a retrieved RDF Object. Clients make permanent changes to an RDF Object by calling an update method that sends the changes to the RDF Object Server. This will allow changes to Action and Working Group Objects to be atomic operations and ensure the database is in a consistent state (for example, an action is either completely added or not added at all, but never partially added).

3.5 Linking and Aggregating RDF Objects

The RDF Object client API should provide extra functionality for creating links between different objects, which could be from different servers. The client API should allow links between Objects to be traversed even when the Objects are on different servers. It should be possible to aggregate data from different databases by making use of Object links. This will allow Working Groups, People and Action Objects to be spread over multiple databases on multiple servers but still be used together.

4. RDF OBJECTS

4.1 Definition of RDF Objects

An RDF Object is a compound data structure that is extracted from a database. More technically, it is a subgraph of the RDF stored in the database. The extraction of the subgraph is performed by a particular extraction algorithm. Each algorithm can have its own parameters. One parameter is common amongst many algorithms is the "focus resource". This is the resource in the database that acts as the central point of the extracted subgraph. The algorithm will start at this resource in the database and recurse through the graph using some path algorithm to extract the RDF Object.

4.2 Object Descriptions

An RDF Object can be represented by a resource in an RDF graph. The RDF Object has properties that describe the Object sufficiently enough for an agent to be able to retrieve the Object. These include:

- The RDF Object Server that provides the RDF Object
- The source database on that server
- The extraction algorithm, focus resource, and other parameters

An example of an RDF Object Description is given in Figure 2. The highlighted resource represents the RDF Object being described. The Object is extracted from the `urn:w3c:members` database provided by the `rdfojects.w3.org` Object Server using the BasicCut extraction algorithm, and `urn:people:tom` as the focus resource. The extracted Object would contain the information about Tom in the database. More information about the BasicCut extraction algorithm is given in section 4.3.1. Note that this graph is not the RDF Object itself, but a description of the Object used to retrieve the Object.

FIG 2: An RDF Object Description

4.3 Extraction Algorithms

Different RDF Objects can be extracted from a database by using different subgraph extraction algorithms. Extraction algorithms are represented by RDFS classes [21] that are subclasses of *ExtractionAlgorithm*. Instances of these classes represent a particular extraction, and have associated properties that are used as parameters to the algorithm. The input database and output object are included as parameters to all algorithms. Figure 2 shows an example of how the parameters are specified in the Object Description.

Two specific extraction algorithms have been developed in order to meet the feature requirements of section 3. The first algorithm is the "Basic Cut" algorithm, which allows information about a particular resource to be retrieved from a database. The depth and breadth of the extracted information can be varied by altering the recursion depth of the algorithm, and by filtering the subgraph by vocabulary. The second algorithm is the "RDQL Search" algorithm, which allows an entire database to be searched using the RDQL query language, and the results returned as an RDF Object.

4.3.1 Basic Cut Algorithm

The Basic Cut algorithm returns information about a resource without the client needing to know what vocabularies are used in the database. The Basic Cut algorithm takes three arguments:

1. The focus resource
2. The recursion depth
3. A optional list of vocabularies to filter by (either an inclusion or exclusion list)

The focus resource can either be a URI, or a blank node with identifying properties. The RDF Object is extracted by first locating the focus resource in the database, and then by recursing in all directions from this resource through the graph to the specified depth. In traversing arcs between resources, the direction of the arrow is

unimportant; both forward and backward arrows are traversed. For example, the statement "Scott isAuthorOf Waverley" is information about both the person Scott and the novel Waverley, and the statement could reasonably be part of both the Scott Object and the Waverley Object.

If a list of vocabularies to include is specified, an arc will only be traversed if it is a property from a vocabulary that is in the inclusion list. If a list of vocabularies to exclude is specified, an arc will only be traversed if it isn't a property from a vocabulary that is in the exclusion list. The algorithm will always traverse RDFS type and label properties of resources even if they would normally be outside of the recursion depth. This is as a convenience to client applications which will in many cases need this information to make sense of the Object, and saves them from having to make an extra query to the database to obtain this information. The Basic Cut algorithm also includes the identifying properties of each blank node (if any). The client API can make use of this information to locate the same resource in a different database by using the identifying properties as a query pattern in the focus resource of the new Object.

Figure 3 shows a Working Group Database, and Figure 4 shows an RDF Object extracted from the database, using the Basic Cut algorithm with Tom as the focus (Tom is the highlighted resource).

FIG 3: A Working Group Database

FIG 4: RDF Object focussed on Tom

4.3.2 RDQL Search Algorithm

The RDQL Search Algorithm allows a database to be queried using RDQL and returns the results as an RDF Object. The algorithm takes an RDQL query string as a single parameter. The algorithm executes this query over the database and converts the query results table into an RDF Object that contains the result data. The Object graph is the minimal complete subgraph for the query, i.e. when the Object is queried with the query string, the results will be the same as if the query was performed against the database. The RDQL Search algorithm, like the Basic Cut algorithm, will include the identifying properties of each blank node in the result set, even if those properties were not queried for. This saves the client from having to manually modify queries to obtain the identifying properties, which is problematic when the client knows little about the structure of the data being queried (see section 2.3). This is the biggest difference between the RDQL Search algorithm and Joseki. The algorithm uses RDQL to locate the results but not as the extraction rule, whereas Joseki uses it for both location and extraction, which is less flexible.

4.4 Links between Objects

Linking is the act of creating a property arc between two resources in different Objects. There are three possible ways of linking resources from different Objects which we call Simple, Identity, and Cross-Linking.

4.4.1 Simple Linking

Simple Linking is the form of linking already possible using Jena's `addProperty()` method of Resources. For example, `x.addProperty(p, y)` creates a link with property `p` between resource `x` and `y`, where `x` and `y` could be from different Objects. A problem occurs when resource `y` is a blank node. The identifying properties of `y` are not copied by Jena into the Object containing resource `x`, making it impossible to determine what resource the blank node refers to. This is demonstrated in figure 5. A link is created between a Working Group in Object A and a person in Object B. The person is identified in Object B using his email address, but this crucial identifying information is lost in Object A when the link is made because Jena does not know to copy the identifying properties of the blank node.

FIG 5: Simple Linking

4.4.2 Identity Linking

Identity linking is an extension to Jena's `addProperty()` that does copy the identifying properties of blank nodes into the other Object. An example is given in figure 6, using the same scenario as in figure 5 but this time creating an identifying link between the Objects. Figure 6 shows Object A after the link is made. The RDF Object API consulted the person schema and determined that the email property is an unambiguous property, and therefore should be copied into Object A. Identity linking is performed using the RDFObject client API: `link(Resource x, Property p, Resource y)`

FIG 6: Identity Linking

4.4.3 Cross-Linking

Cross-Linking is short for Cross-Reference Linking. It extends Identity Linking by adding Object B's Object Description to Object A, connecting the databases as well as the resources. The origins of a resource can now be determined. Applications that cross-link resources from different databases leave trails for themselves and other applications to follow to discover new and relevant data sources. This only becomes possible with Cross-Linking; in Identity Linking the origin of linked-to resource is lost. Figure 7 shows the same scenario again, but this time showing Object A after a cross-link is made.

FIG 7: Cross-Linking

A Cross-Link is analogous to an HTML Hyperlink. Cross-links do for Semantic Web agents what hyperlinks do for Web users. Because hyperlinks are transitive, users can repeatedly follow links to "surf the web". Cross-links are similarly transitive, allowing applications and agents to recursively follow links to find more information about resources, which can be aggregated to form a new Object. Cross-linking is performed using the RDFObject client API: `crossLink(Resource x, Property p, Resource y)`

5. RESULTS AND CONCLUSION

An RDF Object Server was deployed in HP Labs, and the RDF Object client API was used to develop a Mailing List Assistant. Different Objects were stored in different databases: People were stored in a people database, and several Working Groups representing the teams in HP Labs were each stored in their own database. Cross-links were used to connect Working Groups to their members. An Mailing List Assistant was set up to monitor the mailing lists of each group. It was configured using an RDF Object and used cross-links to connect the Working Groups it was to monitor to the configuration object. The following of these links allowed the assistant to access the correct database of each working group and to create new Action Objects in the correct location. A GUI for editing and linking RDF Objects was created to allow the assistant to be re-configured to work with different Working Groups.

RDF Objects allow large RDF databases to be accessed in smaller, more manageable chunks. Compound data structures are encapsulated by RDF Objects, giving applications a simpler conceptual view of the world than if the individual statement was the atomic unit of information. An ontology-aware server and client library allows data to be modelled using blank nodes rather than URIs, which is more convenient for representing real world entities like people and Working Groups. Links between Objects allow clients to discover and aggregate data from different sources. The linking API encourages applications to create links between data sources, and in doing so it provides the glue necessary for the Semantic Web.

REFERENCES

- [1] Resource Description Framework: <http://www.w3.org/RDF>
- [2] DAML+OIL: <http://www.daml.org>
- [3] OWL: <http://www.w3.org/2001/sw/WebOnt/>
- [4] W3C: <http://www.w3.org>
- [5] RDFCore Working Group: <http://www.w3.org/2001/sw/RDFCore/>
- [6] RDFCore Mailing List <http://lists.w3.org/Archives/Public/w3c-rdfcore-wg/>
- [7] W3C Process Document: <http://www.w3.org/Consortium/Process-20010719/>
- [8] HP Labs Semantic Web Research: <http://www.hpl.hp.com/semweb/>
- [9] Jena: <http://www.hpl.hp.com/semweb/jena-top.html>
- [10] VCards in RDF: <http://www.w3.org/TR/vcard-rdf>
- [11] RSS: <http://www.purl.org/rss/1.0/>
- [12] RDF Calendar Taskforce: <http://ilrt.org/discovery/2001/04/calendar/>
- [13] RDF Model and Syntax Specification: <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>
- [14] Joseki: <http://www-uk.hpl.hp.com/people/afs/Joseki/>
- [15] Jena RDB Support:
http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/*checkout*/jena/jena/doc/RDB/rdb-intro.html?rev=HEAD
- [16] Sesame: <http://sesame.aidministrator.nl>
- [17] rdftp: <http://www.semanticweb.gr/rdftp/>
- [18] RDQL: <http://www.hpl.hp.com/semweb/rdql.html>
- [19] Reference by Description in TAP: <http://tap.stanford.edu/tap/rbd.html>
- [20] W3C URI Home Page: <http://www.w3.org/Addressing/>
- [21] RDF Vocabulary Description Language: <http://www.w3.org/TR/rdf-schema/>

Alex Barnell is the author of RDF Objects: aeb99@doc.ic.ac.uk

© Hewlett-Packard Company, 2002