RDF Objects[1]

Alex Barnell
Information Infrastructure Laboratory
HP Laboratories Bristol
HPL-2002-315
November 27th , 2002*

E-mail: Andy_Seaborne@hpl.hp.com

RDF, semantic web, ontology, object-oriented datastructures

The Semantic Web is growing in both size and complexity, with the number of data sources and vocabularies both increasing. This report shows how large RDF databases can be accessed in smaller, more manageable chunks, known as RDF Objects. RDF Objects encapsulate compound data structures, giving applications a more granular view of a database. Applications can control the size and structure of RDF Objects by altering the data extraction rule and by filtering the returned data by vocabulary. Links between RDF Objects, analogous to hypertext links, enable applications to connect and navigate data across different databases. OWL and DAML ontologies are used to discover the identifying properties of resources, allowing information to be aggregated from multiple databases without the need for constant identifiers. An HTTP implementation of an RDF Object Server has been developed and used to develop tools for W3C Working Groups.

* Internal Accession Date Only Approved for External Publication
[1] Enquiries concerning this report should be directed to Andy Seaborne email: Andy_Seaborne@hpl.hp.com

RDF Objects

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

Abstract

The Semantic Web is growing in both size and complexity, with the number of data sources and vocabularies both increasing. This report shows how large RDF databases can be accessed in smaller, more manageable chunks, known as RDF Objects. RDF Objects encapsulate compound data structures, giving applications a more granular view of a database. Applications can control the size and structure of RDF Objects by altering the data extraction rule and by filtering the returned data by vocabulary. Links between RDF Objects, analogous to hypertext links, enable applications to connect and navigate data across different databases. OWL and DAML ontologies are used to discover the identifying properties of resources, allowing information to be aggregated from multiple databases without the need for constant identifiers. 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. It consists of data written in machine-readable languages, such as RDF [1], the Resource Description Framework. To be "machine-readable" means much more than just to be in electronic form. To be machine-readable, the machine must be able to understand the document without ambiguity. This is in contrast to the World Wide Web whose documents are written using languages that machines do not understand, such as English. RDF is the main language of the Semantic Web, a simple language that uses graphs to represent collections of statements. Nodes in the graph represent things, and arrows represent relationships between the things they connect. RDF's original purpose was to allow metadata to be attached to web pages, but it is powerful enough to describe more complex 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] works to develop 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. Actions are assigned to each member of the group in order to create progress. A Group produces specifications and other documents group its course, and once all of the outstanding issues have been resolved, those specifications can become recommendations. 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 [8] at HP Labs, Bristol, is actively developing Jena [9], a Java API for manipulating RDF. Jena provides a rich Model interface giving a resource-centric set of 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 and has been investigating the issues involved in developing a Semantic Web Application using the Jena toolkit. Specifically, they have been tasked with creating a suite of Semantic Web tools to be used by

W3C Working Groups to automate and assist the Working Group process. The field of Semantic Web application development is still in its infancy, and many questions are still to be answered:

- 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?

1.4 Tools for Working Groups

Discussion with members of the RDFCore and WebOnt Working Groups allowed the sweb-apps team to identify three areas in which Semantic Tools would be useful for Working Groups.

Action and Issue Manager - Actions are tasks to be completed, and go through a number of phases that form the action's lifecycle. Actions are created during Working Group meetings, where they are given a description, for example to review a certain document, and are assigned to one or more members of the group. Actions are then active until they are either completed or discontinued. Issues also have a lifecycle, being created then active till either resolved or abandoned. Issues are different to actions in that it is the responsibility of the entire group to resolve an issue, not just a particular individual. Actions and Issues can be modelled as finite state machines, having states corresponding to their phases.

Teleconference Meeting Assistant - Many Working Groups use IRC (Internet Relay Chat) in conjunction with telephone conferences to log the decisions made in the meeting. An IRC robot could assist with this process by automatically walking the group through the agenda items, recording the decisions made, and by automatically producing the minutes for the meeting based on the decisions made. The agenda for the next meeting could also be partially automated by including the recently completed actions as agenda items.

Mailing List Assistant - Working Group members use mailing lists to circulate news and views to the rest of the group. For example, members can send their regrets that they cannot attend the next meeting, or can inform the group that they have completed one of their actions. A mailing list assistant could automatically detect these messages and take the appropriate action.

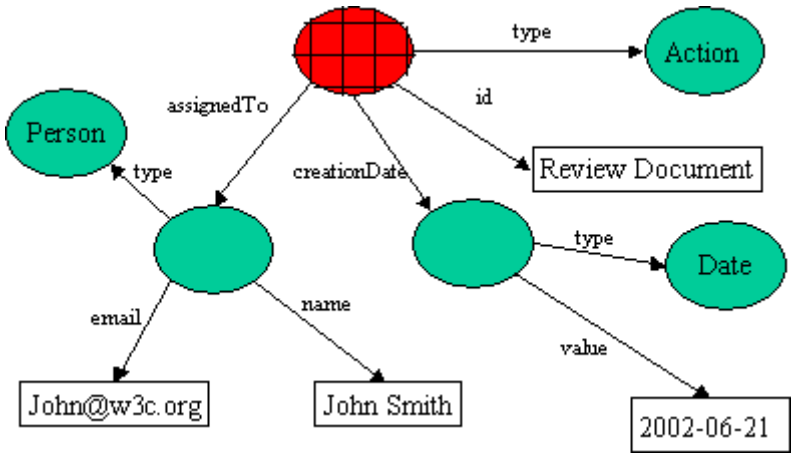

FIGURE 1: An Action represented using RDF

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 rest of the graph structure gives the details of the action: 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 using compound RDF data structures, for example VCards [10], RSS News Feeds [11], and vCal Calendar information [12]. RDF Objects have been designed to encapsulate the concept of compound data structures, enabling them to be extracted from databases and treated as atomic entities. The RDF Object API provides higher-level access to RDF than the Jena API; RDF Objects have a larger granularity than individual statements, which simplifies the development of applications that deal with complex RDF data structures.

2. RDF Databases

Although there is a single specification for the RDF language [13], there is currently no such standardisation for methods of accessing remote RDF databases. A number of proposals have been made by developers, each with different client-server protocols and different client APIs. Joseki [14] and Jena SQL Models [15] are two of the proposals, both with usable implementations. This section describes the two systems 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, an Enterprise 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. It is therefore 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. 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, which also 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. For example, the results might be:

```
?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 creates blank node identifiers so that a client can refer to an existing blank node when adding or removing a statement.

2.2.2 Jena SQL Models

The Jena toolkit comes with a 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, but not DAML or OWL ontologies. rdftp uses a relational database to stored 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 modelled 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 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 a problem in RDQL, because there might be many properties that could be used, but if even one fails then the entire query will fail, because RDQL does not allow partial matches. RDF Objects have been designed to allow clients to query databases without knowing exactly how resources are identified in the database. This is achieved by both the client and server making use of DAML and OWL ontologies behind the scenes, so whereas with Joseki and SQL Models it was the responsibility of the application to identify resources, with RDF Objects, the client library and server do the identification automatically, allowing applications to treat blank nodes like any other resource, simplifying the development process.

3. Requirements for RDF Objects

3.1 RDF Objects

An RDF Object is a piece of information from a database. Different types of information from a database can be retrieved as an RDF Object. In particular, it should be possible to retrieve information about a particular resource in the database, forming a compound data structure. This should be suitable for retrieving actions, people and working groups stored in RDF. It should also be possible to search a database and retrieve the results as an RDF Object. It should be possible to find more information about a resource in an RDF Object, either in the same database or a different one. In the remainder of this report, the terms "RDF Object" and "Object" are used interchangeably.

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. This will allow Object Descriptions to be embedded in Object data, allowing links between Objects (see below).

3.3 RDF Object Server

An RDF Object Server should be able to accept an RDF Object definition from a client and return the generated RDF Object to the client. An RDF Object 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 data from a returned RDF Object. Clients should be able to make permanent changes to an RDF Object by sending the changes it has made to the Object to the RDF Object Server. This will allow changes to Action and Working Group Objects to be atomic operations, and leave the database in a consistent state.

3.5 Linking and Aggregating RDF Objects

A separate RDF Object client API should provide extra functionality for creating links between different objects, which could possibly be on 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 but still be used together.

4. RDF Objects

4.1 Definition of RDF Objects

An RDF Object is a chunk of RDF 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 can have properties that describe itself sufficiently enough for an agent to be able to retrieve the Object. In particular, the properties that will need to be included in the description are:

- 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 patterned resource denotes the RDF Object being described. Note that this graph is not the RDF Object itself, but a description of the Object. The Object is extracted from the `urn:w3c:members` database, using the BasicCut extraction algorithm, and `urn:people:tom` as the focus resource. The resulting Object would contain the information about Tom in the database. More information about the BasicCut extraction algorithm is given in section 4.3.1.

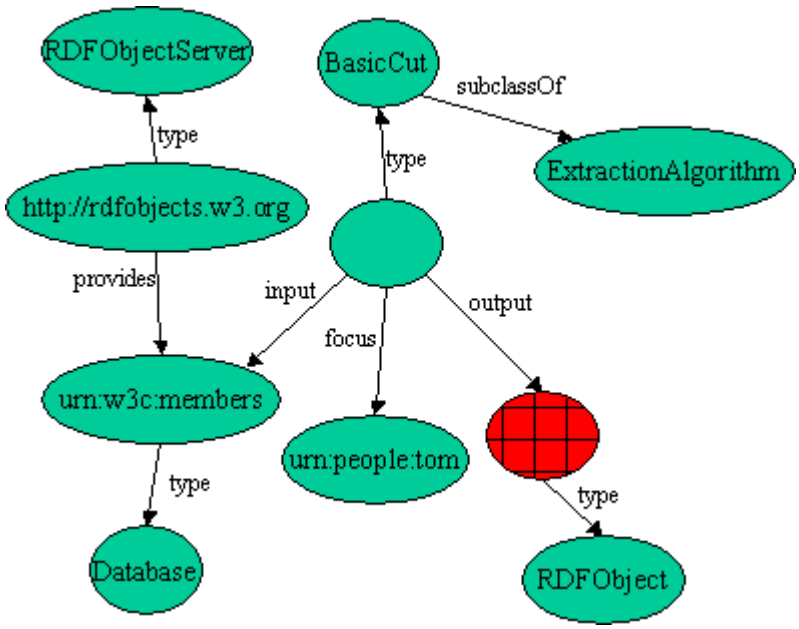

FIGURE 2: An RDF Object Description

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 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.

Two specific extraction algorithms have been developed in order to meet the feature requirements. 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 kind of data is stored about that resource 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; they are treated as non-directional. This is because an arrow that points to a resource is often important information about that resource. For example, the statement "Scott isAuthorOf Waverley" is information about both the person Scott and the novel Waverley, and the statement should 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 includes any 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. 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 patterned resource).

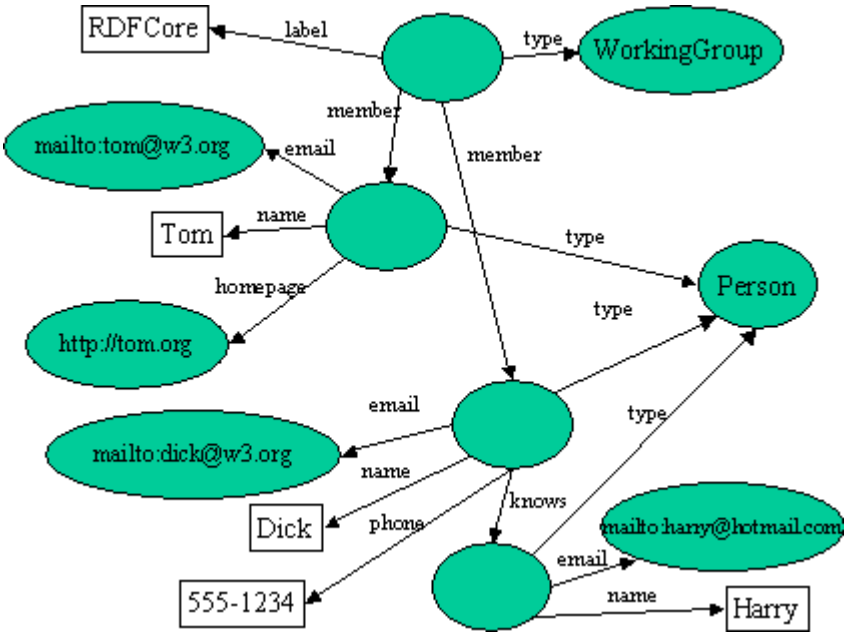

FIGURE 3: A Working Group Database

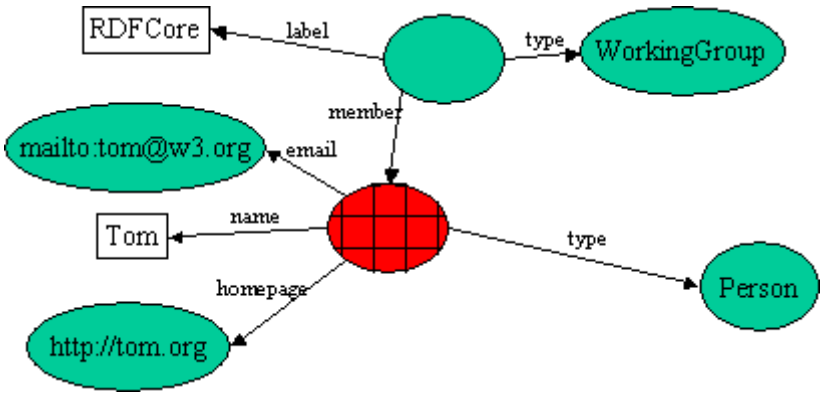

FIGURE 4: RDF Object focussed on Tom

4.3.2 RDQL Search Algorithm

The RDQL Search 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 it 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 where 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, since a partial failure in a query fails the entire query. This is the biggest difference between this implementation of RDQL querying and Joseki. The RDF Object Server uses RDQL to find the results, but not as the extraction rule.

4.4 Links between Objects

4.4.1 Use Cases

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

4.4.2 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 information is lost in Object A when the link is made, because Jena does not copy the identifying properties of the blank node.

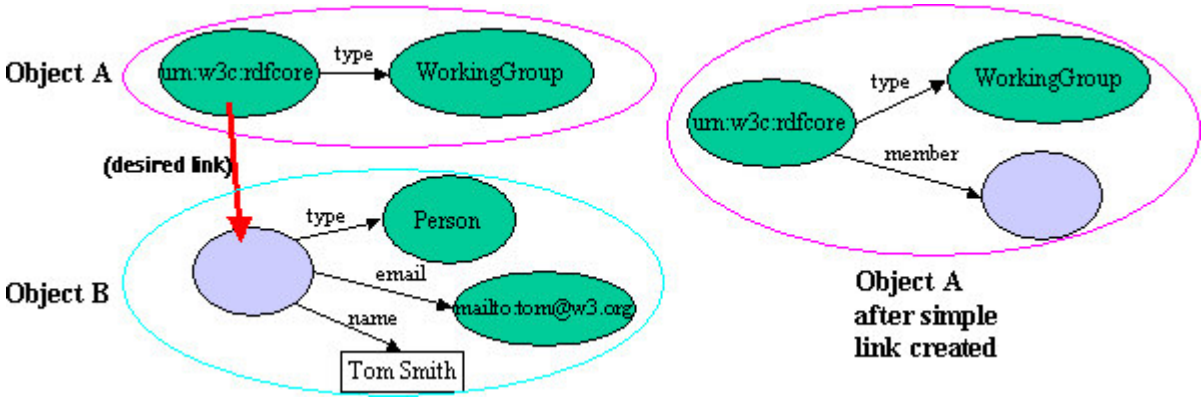

FIGURE 5: Simple Linking

4.4.3 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 using an identifying link to copy the identifying properties of the person. Identity linking is performed using the RDFObject client API: `link(Resource x, Property p, Resource y)`

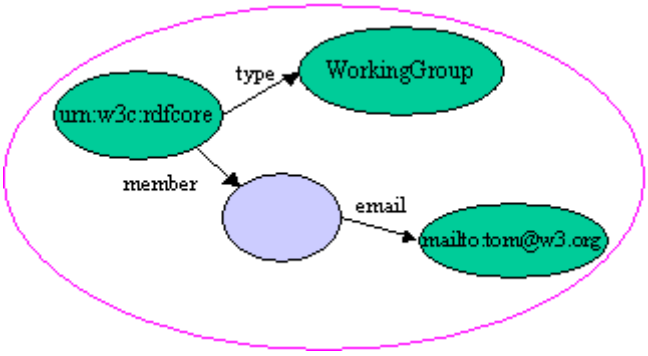

FIGURE 6: Identity Linking

4.4.4 Cross-Linking

Cross-Linking is short for Cross-Reference Linking. It extends Identity Linking by adding information about the origin of the linked-to resource to the object of the linked-from resource. This allows an agent who subsequently accesses Object A to discover information about the linked-to in Object B without previously knowing about Object B. 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 using a cross-link.

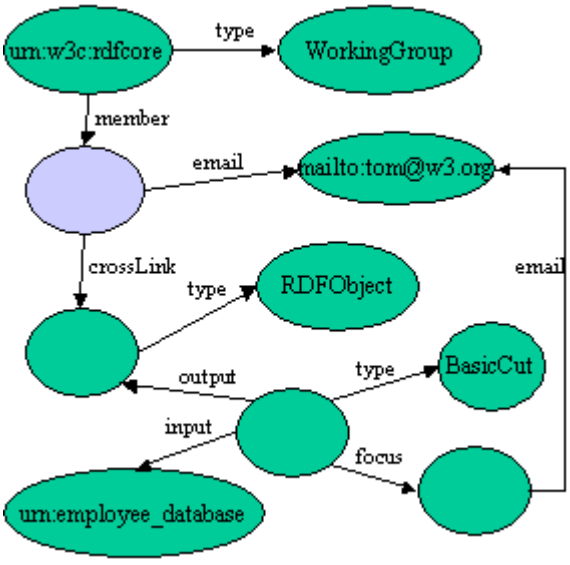

FIGURE 7: Cross-Linking

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". In the same way, cross-links are also transitive, allowing applications and agents to recursively follow links to find more information about resources, which can be aggregated into 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 their own database. Cross-links were used to connect Working Groups to their members. An email bot 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 bot 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 bot to be re-configured to assist 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 non-web 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/

Alex Barnell is the author of RDF Objects, and can be contacted at aeb99@doc.ic.ac.uk