Ontological Modeling: Part 1

Terry Halpin
LogicBlox

This is the first in a series of articles on ontology-based approaches to modeling. The main focus is on popular ontology languages proposed for the Semantic Web, such as the Resource Description Framework (RDF), RDF Schema (RDFS), and the Web Ontology Language (OWL). OWL is based on description logic. A later series of articles will explore other logic-based languages such as Datalog. This first article provides a brief introduction to ontologies, the Semantic Web, and RDF.

Ontology and Ontologies

The term *ontologia*, Latin for "ontology", was coined independently in 1613 by the scholastic philosophers Rudolf Göckel (Goclenius) and Jacob Lorhard [5]. Etymologically, the term "ontology" (Greek: *ontos* = being; *logos* = reasoned discourse) means the study of existence or reality, and this term was traditionally used in philosophy for that branch of metaphysics that seeks answers to fundamental questions about the nature of being, such as "Do nonphysical things exist?", and "Can things that undergo change maintain their identity?". Some more recent philosophers (e.g. Heidegger and Quine) have used the term for related but less ambitious activities [3].

More recently[1], the informatics community adopted the term "ontology" for a different purpose. Nowadays, the information systems community typically uses "ontology" for a conceptual model of some business domain, where the model is expressed in a way to facilitate sharing information about that domain by conforming to some standard set of constructs. At the low end of the scale, simple taxonomies (e.g. botanical classification schemes) are sometimes spoken of as ontologies. However, a full-fledged ontology ought to include a declaration of relationships between the objects of interest, as well as rules about them, including constraints on what is allowed and derivation rules for inferring new facts about them.

In contrast to the philosophical study, an ontology used in information systems is typically restricted to a single business domain. So you can have different ontologies for different domains (banking, botany, etc.). The main thing that distinguishes ontologies from some conceptual models is that they should be designed from the ground up to be shared with others. For examples, all terms (qualified with their context if necessary) should have a standard interpretation among all users.

The Semantic Web

Although millions of documents are readily available on the Internet, they were created by many different people, typically with little thought as to how their content might be shared or combined with other documents. As a result, the meaning of the vast majority of these documents typically cannot be automatically extracted. In 2001, the term "Semantic Web" was coined by Sir Tim Berners-Lee (founder of the World Wide Web), Jim Hendler, and Ora Lassila in an article they wrote for *Scientific American* to address this problem [1].

In essence, the Semantic Web proposal recommends enriching Web documents with global identifiers and structure, for example using Uniform Resource Identifiers (URIs) and embedded tags, to reveal the semantics of what is to be shared in a way that is accessible to automated agents.

[1] According to Smith [5], its first use in the informatics literature was in 1967 by Mealy in a work on the foundations of data modeling [4].

In 1999, the World Wide Web Consortium (W3C) defined version 1 of the *Resource Description Framework (RDF)* as a standard on top of the eXtensible Markup language (XML) to capture metadata for Web resources (e.g. listing the authors of the document, and the topics addressed by the document). Concurrently, the U.S. Defense Advanced Research Products Agency (DARPA) produced the DARPA Agent Markup Language (DAML) to add semantics via embedded tags. For similar purposes, the European Union (EU) developed the Ontology Inference Layer (OIL). In 2001 the US and EU collaborated to develop the DAML+OIL language, which incorporated many OIL constructs into DAML. DAML included DAML-ONT for ontology specification and DAML-LOGIC for inference. DAML+OIL drew mainly on DAML-ONT plus OIL. In 2002, DAML+OIL was submitted as a standard to the W3C.

In the same time frame, RDF was extended by a simple ontology language *RDF Schema (RDFS)*, and this was adopted in 2004 by the W3C. Also in 2004, the W3C formally recommended the *Web Ontology Language (OWL)*. OWL incorporates many aspects of DAML+OIL and RDFS, has a formal semantics based mainly on description logics, and is now the most popular textual language for developing ontologies for the Semantic Web.

Figure 1 depicts the layered foundation proposed for the Semantic Web. Here upper layers depend on lower layers. The bottom layer is expressed as strings of Unicode characters, including URI references (URIrefs), which are explained later. The next layer is expressed in XML, using the XML Schema Definition language (XSD) data types. RDF and RDFS are built on the XML layer, and OWL is built on the RDF layer. Finally, actual web applications may be built using OWL.

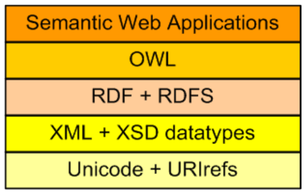

Figure 1 Foundational layers for the Semantic Web.

Resource Description Framework (RDF) Basics

Full technical details as well as a primer on RDF may be found at the Website www.w3.org/RDF/. An *RDF model* may be visualized as a directed graph, composed of labeled nodes (also known as vertices) and labeled, directed arcs (also known as edges or links), with each arc denoting a *binary relationship* between two nodes. Figure 2 shows a simple example of a directed graph of binary relationships with simple labels (names) for the nodes (*a*, *b*, *c*, and *d*) and edges (e_1 .. e_6).

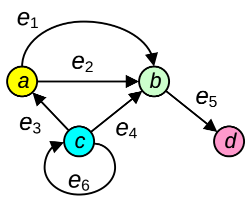

Figure 2 A labeled, directed graph of binary relationships.

Each RDF node is either a resource or a literal, but not both. A *resource* is anything that is identified using a *Uniform Resource Identifier* (*URI*), and is basically an entity instance or a logical predicate. A URI is either a *Uniform Resource Locator* (*URL*) or a *Uniform Resource Name* (*URN*). Using "::=" for "is defined as" and "|" for exclusive-or, this may be stated in Backus Naur Form (BNF) thus:

```
URI      ::=   URL | URN
```

A URL (e.g. http://www.ORM.net) includes a network access type (e.g. http, for hypertext transfer protocol) and a network address to locate the resource (e.g. www.ORM.net). A URN (e.g. urn:isbn:978-0-12-373568-3) includes a namespace (e.g. isbn, for International Standard Book Number), and a name (e.g. 978-0-12-373568-3) that identifies the resource within the context of that namespace. Unlike a URL, a URN does not provide a link to locate the resource on the Web. A specific URL or URN is a global identifier, so always refers to the same resource even when used in different documents.

More precisely, RDF identifies a resource by means of a *URI reference* (*URIref*), which optionally appends "#" followed by a local fragment identifier to a URI. Using square brackets "[ ]" to enclose an optional item, this may be summarized in BNF thus:

```
URIref    ::=    URI ['#' fragmentId]
```

so

```
URIref    ::=    (URL | URN) ['#' fragmentId]
```

For example, the URIref http://www.w3.org/TR/rdf-primer/#basicconcepts is comprised of the URI http://www.w3.org/TR/rdf-primer/ followed by a # separator and the fragment identifier basicconcepts. If you enter that URIref in your Web browser, you will access Section 2.1 "Basic Concepts" of the RDF primer on that site. The character strings used for URIrefs are composed of Unicode characters.

While RDF has no standard graphical notation, small RDF models may be conveniently depicted as directed graphs, using URIrefs to label the resources. To help distinguish resource nodes from literal nodes, I'll use soft rectangles (with rounded corners) for resource nodes and hard rectangles for literal nodes. Figure 3 shows a simple example taken from [2] of an RDF model that corresponds to the following fact:

The Book identified by ISBN 1-55860-672-6
is authored by
a Person
who has the name "Terry Halpin".

In Figure 3, the resource node labeled "urn:isbn:1-55860-672-6" identifies a specific book, and the resource node labeled "http://www.orm.net/People.contact#th" provides a Web address that identifies a specific person. So these two nodes denote object *instances*. In contrast, the resource nodes labeled "http://www.eg.net/concept#Book" and "http://www.eg.net/concept#Person" respectively identify the object *types* Book and Person. So RDF models treat instances and types in basically the same way—they are just resources. It should already be clear that RDF models are fundamentally different from data models in ORM, ER, or UML, where instances and types are treated very differently.

In Figure 3, the literal node labeled "Terry Halpin" denotes the untyped literal "Terry Halpin". Textually, an RDF model is a set of simple *statements*, where each statement may be represented as a (*subject*, *predicate*, *object*) *triple*. Figure 3 displays four triples in graphical format. Each of the four labeled arrows depicts a binary relationship of the form (subject, predicate, object), with the arrow pointing from the subject to the object.

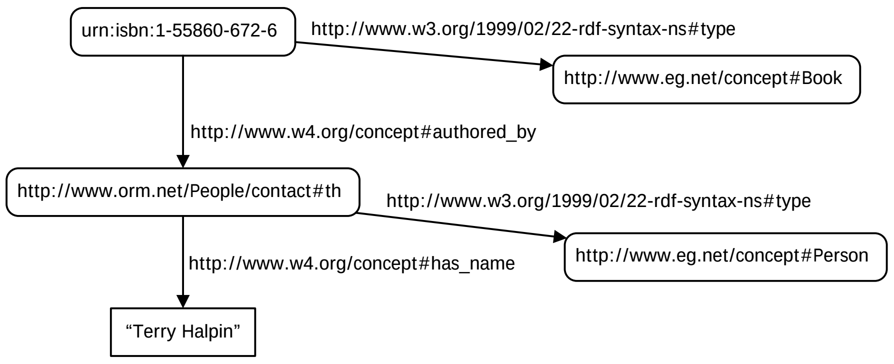
urn:isbn:1-55860-672-6
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.eg.net/concept#Book
http://www.w4.org/concept#authored_by
http://www.orm.net/People/contact#th
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.eg.net/concept#Person
http://www.w4.org/concept#has_name
"Terry Halpin"

Figure 3. A simple RDF model depicted as a labeled, directed graph.

Here “predicate” means logical predicate or verb phrase usage, rather than grammatical predicate (which would include the object). Rather unfortunately, RDF also refers to a predicate as a “property” of the subject. The label beside the arrow provides a URIref (and hence a globally identifying name) for the predicate. In general, subjects and predicates are resources, so must be identified by URIrefs. Objects must be either resources or literals.

The predicate labeled “http://www/w4.org/concept#authored_by” may be thought of as the predicate “is authored by”. But there’s more to it than that. There could be hundreds of documents or models containing a predicate called “is authored by”, and some of those might differ in meaning. By including a full URIref, we are able to unambiguously specify exactly which of those occurrences of “is authored by” is intended. Similarly for the “has_name” predicate.

As a more obvious example where such precision is important, consider the fact types Person runs Barbershop and Horse runs Race. Figure 4(a) shows this in ORM, and Figure 4(b) shows it in UML. Constraints such as uniqueness and multiplicity constraints are omitted since these are not relevant to the RDF example. Here the occurrences of “runs” have different semantics and hence denote different predicates. Because different predicates may have the same “short” predicate reading, some way of distinguishing them must be found. ORM, ER, and UML typically use an extended reading (such as a full fact type reading, or a reading supplemented by role names) to make a natural distinction, while using internal identifiers for implementation. RDF instead solves this problem by treating predicates as resources, and thus using different “long” predicate readings provided by different URIrefs, as shown in Figure 4 (c).

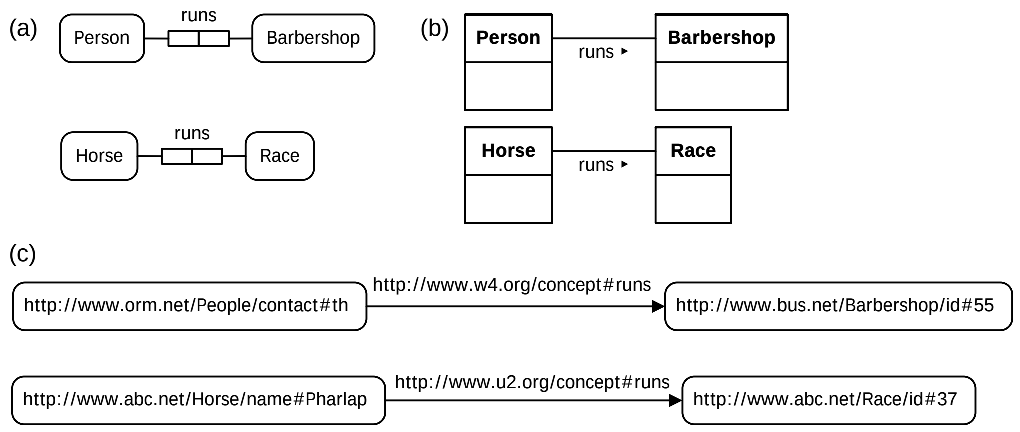
(a)
Person
runs
Barbershop
Horse
runs
Race
(b)
Person
runs ▸
Barbershop
Horse
runs ▸
Race
(c)
http://www.orm.net/People/contact#th
http://www.w4.org/concept#runs
http://www.bus.net/Barbershop/id#55
http://www.abc.net/Horse/name#Pharlap
http://www.u2.org/concept#runs
http://www.abc.net/Race/id#37

Figure 4. Two different “runs” predicates in (a) ORM, (b) UML, and (c) RDF.

In one way, RDF’s approach is more general, since it also provides a way to indicate when different predicate occurrences denote the same predicate. For example, if we add the fact type Person runs Race to the above example, this new occurrence of “runs” has the same meaning as the “runs” in Horse runs Race. In RDF, we can show this by using the same URIref for it (just as we did with the “is of type” predicate in Figure 3).

Note that the authored_by and has_name predicates in Figure 3 involve *instance-to-instance relationships*, which is normal for conceptual schemas in ORM, ER, or UML. In contrast, the “http://www.w3.org/1999/02/22-rdf-syntax-ns#type” predicate is a predefined *instance-to-type relationship*. Here “type” really means “is of type”. The RDF model in Figure 3 is roughly equivalent to the ORM and UML models in Figure 5. Again, constraints are omitted as they are not relevant to the RDF model. I use the term “model” to mean the schema (structure) plus the population (instances).

In the ORM model in Figure 5(a), object and fact instances are represented by their identifying values in fact tables below the relevant object type and predicate shapes. Each predicate role (depicted as a box) is associated with a column of a fact table. In this example, persons are identified by their initials (e.g. “th”) to conform to the RDF model. Book and Person are entity types, and PersonName is a value type.

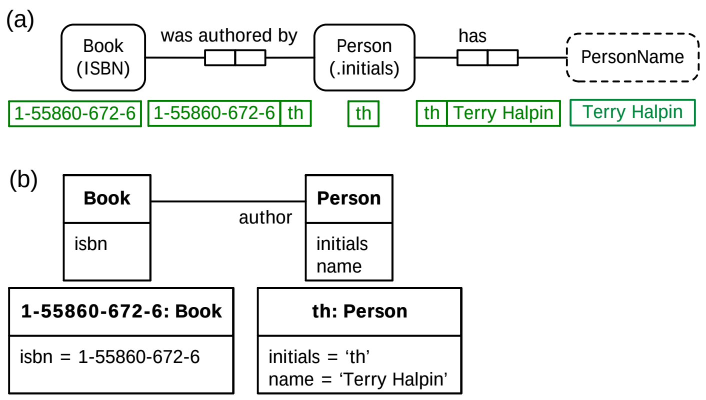

Figure 5. (a) ORM model and (b) UML model for the RDF example in Figure 3.

Figure 5(b) shows a UML class diagram for the schema, supplemented by object diagrams to capture the population of the Book and Person classes, including their attribute values. UML doesn't provide a graphical way to depict populations of associations. In this sample population, there is only one person and only one book, so this doesn't matter much for this example. However, once the classes are populated with many instances, we lose the ability to see graphically in UML which is associated with which.

In Figure 5, the authorship association is depicted attached to Book and Person, which are types or classes. However the underlying "is authored by" predicate is understood to apply to instances of these types, not the types themselves. A book instance may be authored by a person instance, but it makes no sense to say that the Book type is authored by the Person type. When we talk about facts, relationships, or associations in ORM, ER, or UML we typically mean an instance-to-instance relationship. Instance to type relationships are declared in the population, not in the schema. Type to type relationships (e.g. Woman is a subtype of Person) may be declared in the schema but these are treated as constraints, not as ground facts (which apply a predicate to individuals).

In contrast to this careful separation of instances and types, RDF treats all three of the following kinds of relationships in the same way: instance-to-type; instance-to-instance; type-to-type. Figure 6 illustrates this idea graphically, using a soft rectangle for a type or class, and a small dot for an instance (individual).

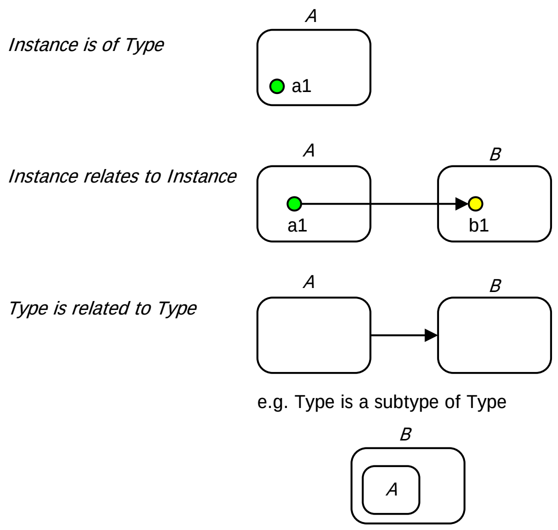

Figure 6. RDF treats all of these three kinds of relationships in the same way.

RDF even allows one to declare that a type (or class) is an instance of itself, as shown in Figure 7. For example, you could say that Class is an instance of Class by using the same URIref for Class as both subject and object, and using the “is type of” predicate mentioned earlier. This extreme freedom can lead to formal problems, such as Russell’s Paradox, and is best avoided. For example, by introducing different levels in which things are treated as individuals, one may treat Class as an instance of MetaClass (which is an instance only at a higher level).

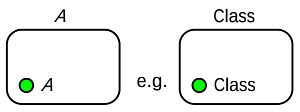

Figure 7. RDF allows you to declare that a type is an instance of itself.

Not only does RDF not introduce levels, but it assigns no special semantics to any predicate. For example, “is_an_instance_of” and “is_a_subytpe_of” are treated just like “has_name”, except for having different character strings. Consequently, no inferences may be performed based on known properties of special predicates (e.g. given that *A* is a subtype of *B*, and that *B* is a subtype of *C*, we can’t use transitivity of subtypehood to deduce that *A* is a subtype of *C*). As we’ll see in later articles, other ontology languages such as OWL do provide such inferencing capabilities.

RDF literals may be untyped (e.g. “US”, “3”) or typed. A *typed literal* pairs a value with a data type, where the data type is identified by a URIref to a standard XML data type, e.g. 3: xsd;nonNegativeInteger. RDF also supports collections, such as sets and lists. RDF graphs may be serialized into XML using RDF/XML. Some commercial database management systems (e.g. Oracle) have built-in support for storing RDF triples.

Conclusion

This article provided a simple introduction to ontologies and the Semantic Web, and then covered most of the basic concepts in the Resource Description Framework (RDF), contrasting them with other data modeling approaches. The next article will complete the coverage of RDF, and examine RDF Schema (RDFS). Later articles in this series will discuss the different flavors of the Web Ontology language (OWL).

References

1. Berners-Lee, T., Hendler, J. & Lassila, O. 2001, ‘The Semantic Web’, *Scientific American*, May 2001.
2. Halpin, T. & Morgan, T. 2008, *Information Modeling and Relational Databases, 2nd edition*, Morgan Kaufmann, San Francisco.
3. Macintyre, A. 1967, ‘Ontology’, *The Encyclopedia of Philosophy*, (Ed.) P. Edwards, Macmillan, New York.
4. Mealy, G. 1967, ‘Another Look at Data’, *AFIPS Conference Proceedings, vol. 31*, Academic Press, London, pp. 525-534.
5. Smith, B. 2004, ‘Ontology and Information Systems’, Online at http://ontology.buffalo.edu/ontology(PIC).pdf.