The Data Administration Newsletter
Since 1997

Data Modeling, RDF, & OWL - Part One: An Introduction To Ontologies

by David C. Hay
Published: April 1, 2006

(Article URL: http://www.tdan.com/view-articles/5025)

Published in TDAN.com April 2006

[*This is the first of three articles discussing the new/old ideas of semantics and ontology and how they affect the way we analyze data. This article introduces the main concepts, and the second article will show an example of converting a data model to the web ontology language, OWL.*]

Everyone knows that we are drowning in information, both from the databases in our companies as well as from the world-wide web, the media, and life in general. The information technology industry has been wrestling with this problem for years, and one is entitled to wonder if things will ever get better.

Well, there are a couple of new/old ideas on the horizon that might help: semantics and ontology.

Data modeling was invented three decades ago to assist in the design of databases-in particular relational databases. As it matured, the technique has become recognized as a tool for analyzing the semantics of an organization-what is the structure of the organization's information as it is used in carrying out its mission?

In recent years, from a completely different direction, the artificial intelligence world, semantics has arisen as a subject of interest in its own right. This came the artificial intelligence world's desire to create computerized natural language processors.

These two fields are finally coming together, and this article is an attempt to articulate that link.

In particular, companies are beginning to recognize that semantics is important if their systems (and their people, for that matter) are going to communicate with each other, and, based on this recognition, they are also recognizing the importance of collecting "ontologies", or glossaries that describe the language they use to carry out their activities.

In other words, a couple of 2500 year-old words are becoming the hot new buzzwords in our industry.

Semantics is the Greek philosophic study of the nature of meaning, especially as it is expressed in language. It is the "study of the signification of signs and symbols, as opposed to their formal relations (syntactics)."[1] ***Ontology*** is another branch of Greek philosophy, "concerned with identifying, in the most general terms, the kinds of things that actually exist."[2]

In other words, ***ontology*** tells us *what exists*. ***Semantics*** tells us *how to describe it.*

WHAT IS A DATA MODEL?

A ***data model*** is a drawing that represents data "things" and relationships between them. The meaning of the model varies, depending on its purpose:

- It can represent a *data base design*, with the boxes representing tables and the lines representing foreign keys. Also represented are the columns of the tables.
- It can be a ***conceptual*** model representing the structure of a business, with the boxes representing things of significance to the business and the lines representing semantic relationships between them. Also represented may be the definitions of data describing those things of significance.

These are very different things. A business (or "conceptual") data model captures the semantics of an organization for the purpose of both communicating both with the business community and providing and architecture for database and system design. A database design describes an artifact that can be employed to store and manipulate data.

Other than constraints on cardinality, *business rules* are not generally represented on data models of either kind. Even in the case of business data models, the models are supposed to represent fundamental structures, while business rules represent variable constraints.

In other words, database design, business data modeling, and business rule modeling are three very different things. They do, however, represent a particular mindset, which for purposes of this article, we will characterize simply as the "data modeling mindset".

This article then uses that to describe a completely different mindset.

About Data Models and Ontology Languages

A conceptual data model is, of course, a kind of ontology. It is about defining *categories* of data. Its graphic nature provides an excellent basis for discussing and negotiating the meaning of those categories. Accompanied by business rules analysis, the two provide a basis for collecting data according to those categories, and its corresponding database design provides a mechanism for doing so. The point is that data models are to be understood by humans, with computers only serving as gateways to permit capture of "valid" data.

In its latest incarnations, however, an ontology language begins with *instances* of actual data. It's purpose is to classify them so that computers can make inferences from them.

The data modeling mindset is based upon the ***closed world assumption****:*

> *Only that which is asserted is known.*

Ontology languages are based on the ***open world assumption***.

> *All assertions are assumed to be true until proven otherwise.*

This means that when you build a system using a data modeling approach:

- You can only enter data that you know to be valid.
- You are "encouraged" to enter complete information.
- There are no other data.
- The data model entity classes and their derived tables are templates.

With an ontology database:

- You can enter what you know to be true.
- You can enter incomplete information.
- You (and the computer) can *infer* other things.
- Ontology classes are simply sets of things.

This is a profoundly different view of the world, as we shall see, below.

About the Semantic Web

Before going into ontology languages in detail, it is worth taking a moment to understand "The Semantic Web". As imagined by Tim Berners-Lee, the inventor of the World-wide Web, the "first step is putting data on the Web in a form that machines can naturally understand, or converting it to that form. This creates what I call a Semantic Web-a web of data that can be processed directly or indirectly by machines".[3]

Ok, so how is that different from simply creating a networked database? If all validation of data is in a program, the program acts as a filter, the way we discussed before. If, on the other hand, data are stored with the semantics visible to a wide range of processors, then the data are more powerful, and the opportunities for discovering new things in them is greater. Michael Daconta and his colleagues describe four stages in the smart data continuum:

- ***Text and databases (pre-XML)***-Most data are proprietary to an application. The "smarts" are in the application and not in the data.
- ***XML documents in a single domain***-Here data achieve application independence within a domain. For example XML could describe standard semantics within the health care industry, the insurance industry, and so forth.
- ***Taxonomies and documents with mixed vocabularies***-In this stage data can be collected from multiple domains and accurately classified. This classification can then be used for discovery of data. Simple relationships between categories in the taxonomy can be used to relate and thus combine data. Data are now smart enough to be easily discovered and sensibly combined with other data.
- ***Ontologies and rules***-in this stage, new data can e inferred from existing data by following logical rules. Data are not smart enough to be described with concrete relationships and sophisticated formalisms. Logical calculations can be made on this "semantic algebra". In this stage data no longer exist as a blob

but as a part of a sophisticated microcosm.[4]

The semantic web, then, is an extension of the World-wide web to allow for not just the retrieval of documents based on key words, but for their retrieval based on the semantics of their contents.

The semantic web is based on the idea of a "layered architecture". Much like the ISO concept of layers in data communications, the semantic web architecture is composed of the following layers:

- ***URIs and Namespaces***-the names of things
- ***XML and XMLS Data types***-a means of communicating data
- ***RDF and RDF/XML***-a basic language
- ***RDF Schema and Individuals***-an ontological primitive
- ***Ontology languages, such as OWL***-the logical layer
- ***Applications***-the implementation layer.

The field is new, and it is not clear to this author just what the "Applications" layer might look like. But we can address the others. Specifically, RDF and OWL represent structured languages for representing ontologies that we can map back to what we are used to doing with data models.

Uniform Resource stuff

In order to talk about something, it is necessary to name it. The semantic web provides a scheme for naming things in two layers.

First of all, the general concept of a *Uniform Resource Identifier* (URI) is simply a formatted identifier that identifies anything. The name is in two parts:

- A ***scheme name***, and
- A ***scheme-specific name.***

There is no outside control over URIs, so they can be whatever you want them to be, such as:

> *hay:david*

Note, of course, that within the context of a particular ontology, all URIs must be unique.

A *Uniform Resource Locator* (URL) is a URI that is specifically used to locate resources on the World-wide Web. The scheme name and the first elements of the scheme-specific name are regulated to insure uniqueness across the World-wide Web. To call a particular URL your own, you have to get permission from the Internet Corporation for Assigned Names and Numbers (ICANN). For example, the following is the URL of your author's company:

> *http://www.essentialstrategies.com*

The literature also describes a *Uniform Resource Name*, but the descriptions are contradictory and confusing. It apparently comes down to a URI whose scheme is "urn:".

Namespaces

An *XML namespace* is the URI that describes an ontology from which terms are taken. As you will see, in this context XML is the language that is used to describe an ontology. Since the description of an XML namespace can be lengthy, a prefix is usually assigned to each, in order to simplify referring to a term.

For example, the set of terms that define the OWL language is itself an ontology, defined in XML. Its namespace, then is described as follows:

```
xmlns:owl="http://www.w3.org/2002/07/owl"
```

In the OWL namespace, then, the term "class" would be described thus:

```
xmlns:owl="http//www.w3.org/2002/07/owl#class"
```

Once the namespace is declared, however, this can be abbreviated:

```
owl:class
```

XML and XML Schema

As mentioned above, the RDF and OWL languages are expressed in XML.

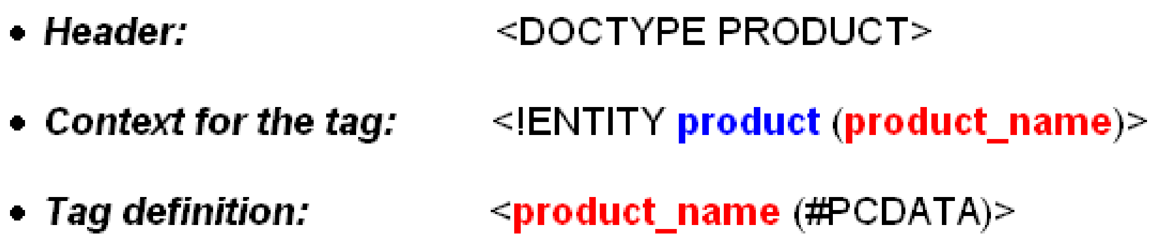
Here's a whirlwind synopsis of XML:

An XML document contains "tags" describing strings of text. These are similar to the tags in HTML, but where HTML tags describe formatting components of a document, these tags describe the semantic content of it. For example,

```
<product>
  <product name>BlackBerry</product name>
</product>
```

As you can see, the tag describes the text that follows. The text is then demarked by a corresponding end tag in the form .

Tags are typically defined in accompanying files called *data type definitions* (DTD). A DTD is itself a document with the following structure:



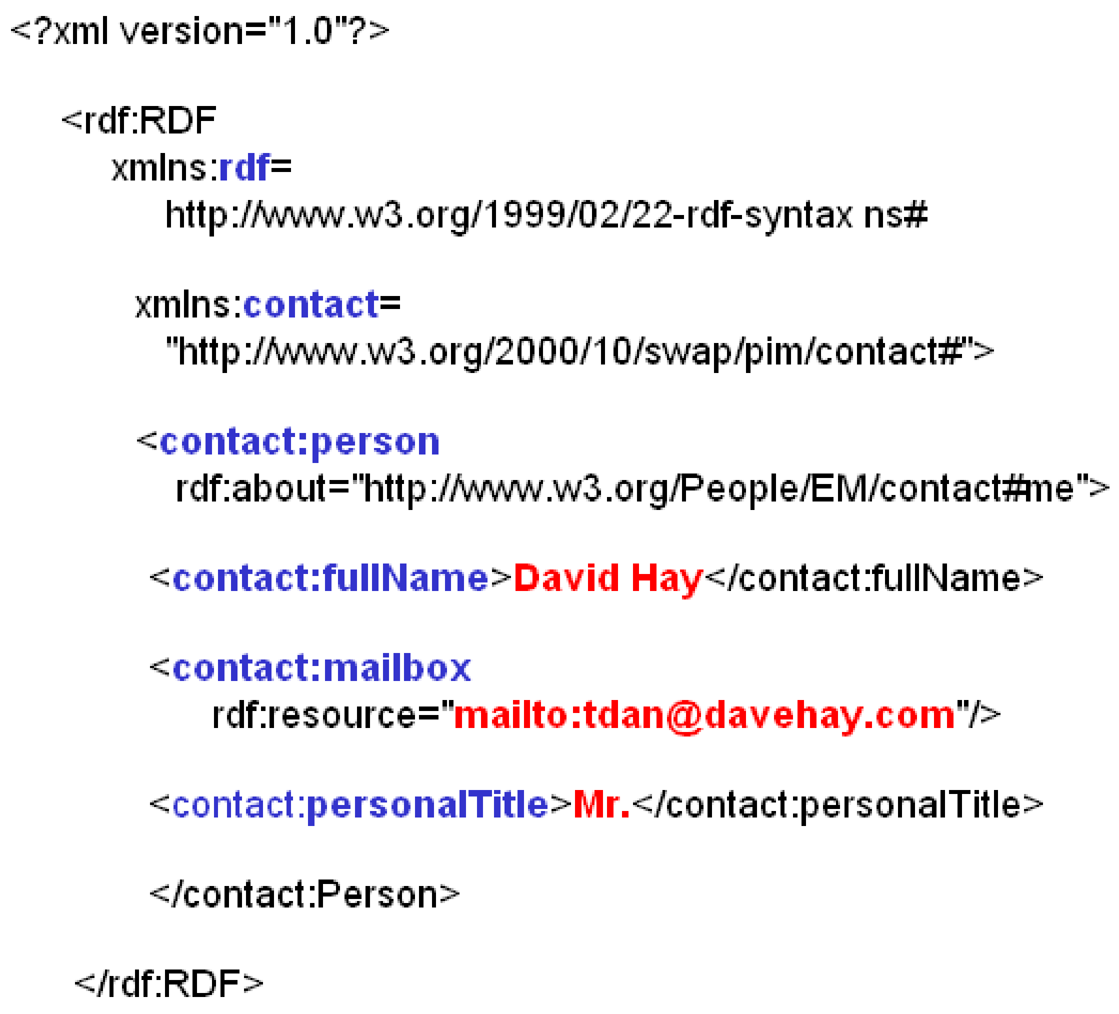
- ***Header:*** <DOCTYPE PRODUCT>
- ***Context for the tag:*** <!ENTITY **product** (**product_name**)>
- ***Tag definition:*** <**product_name** (#PCDATA)>

Note that "(#PCDATA)" simply means that's where actual data go. In the context line, a character may be added after the tag name (for example product_name+). The character determines how many occurrences of the tag are required for each occurrence of the context tag:

- ***(no character)*** - (Default) mandatory single valued (must be ... one and only one.)
- + - Mandatory one or more occurrences (must be ... one or more).
- ? - optional, single valued (may be one and only one).
- * - optional, one or more occurrences (may be one or more).

XML schema is an alternative to DTDs. XML Schema is an XML document that configures other documents.

RDF and OWL are defined as tags in XML Schemas. An ontology is defined as a namespace, and terms are described as elements of that namespace. For example, the Ontology "contact" might be used as follows (note that RDF itself must be defined first):



```
<?xml version="1.0"?>

   <rdf:RDF
      xmlns:rdf=
         http://www.w3.org/1999/02/22-rdf-syntax ns#

        xmlns:contact=
          "http://www.w3.org/2000/10/swap/pim/contact#">

        <contact:person
           rdf:about="http://www.w3.org/People/EM/contact#me">

         <contact:fullName>David Hay</contact:fullName>

         <contact:mailbox
              rdf:resource="mailto:tdan@davehay.com"/>

         <contact:personalTitle>Mr.</contact:personalTitle>

         </contact:Person>

    </rdf:RDF>
```

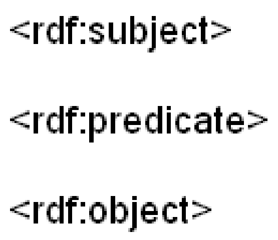
First the contact namespace is defined with the name "contact", and the terms "person", "fullName", "mailbox" and "personalTitle" are used to capture values. The paragraph above asserts that person (me) is described by a full name "David Hay", my mailbox is "tdan@davehay.com", and my (personal) title is "Mr."

Note that contact:person is equivalent to:

http://www.w3.org/2000/10/swap/pim/contact#person.

RDF and RDF/XML

Resource Description Framework is the basic language layer for data representation. It is rendered in XML, and consists of a preliminary set of tags for describing semantics. RDF can be used as metadata to describe documents and images. Its most important tags are:



```
<rdf:subject>

<rdf:predicate>

<rdf:object>
```

RDF and Data Modeling:

RDF tags that correspond to data modeling constructs are the following:

- ***Entity class:***: <rdf:subject>
- ***Relationship:*** <rdf:predicate> **+** <rdf:object>
- ***Attribute:*** <rdf:predicate>**an attribute of**</rdf:predicate>

For example,

<rdf:**subject**>**Kleenex brand tissues**</rdf:subject>

<rdf:**predicate**>**are sold by**</rdf:predicate>

<rdf:**object**>

Kimberly Clark Corporation

</rdf:object>

Note that there is no distinction between classes and instances.

RDF Schema

Resource Description Framework Schema (RDFS) is an extension of RDF. In addition to the RDF tags available are additional tags to define:

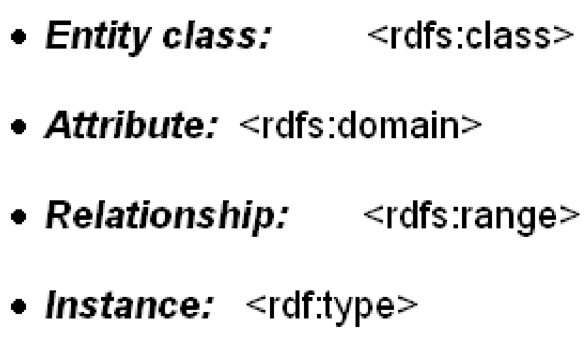
- Resource
- Class
- Sub-class
- Range
- Domain

and others.

Interestingly enough, some RDF tags are actually defined using RDFS tags.

RDFS and Data Modeling

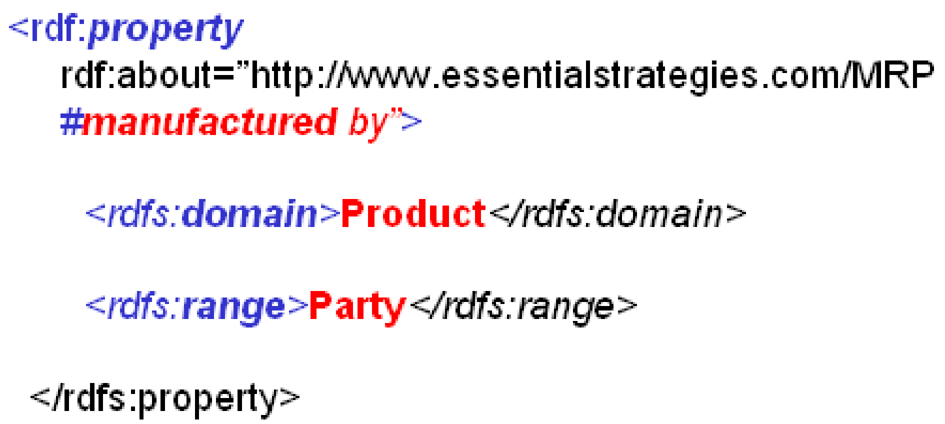
RDFS tags that correspond to data modeling constructs are the following:



- ***Entity class:*** <rdfs:class>
- ***Attribute:*** <rdfs:domain>
- ***Relationship:*** <rdfs:range>
- ***Instance:*** <rdf:type>

In RDFS, attributes and relationships are properties that are defined *before* assigning them to classes.

For example, imagine an Essential Strategies, Inc. ontology called "MRP", containing the concept "manufactured by". This could be defined as a property as follows:



```
<rdf:property
   rdf:about="http://www.essentialstrategies.com/MRP
   #manufactured by">

     <rdfs:domain>Product</rdfs:domain>

     <rdfs:range>Party</rdfs:range>

 </rdfs:property>
```

Here, the relationship "manufactured by" being defined in terms of the classes it relates. Contrary to the way data modelers use the word "domain", here ***domain*** is the class that is on the first end of the relationship. ***Range*** refers to the class that is on the second end of the relationship. A relationship is considered a property of the first class.

Note that all relationships and attributes are considered optional many-to-many. There are no cardinality constraints in RDF.

Ontology Languages

As you can see, even RDFS is quite limited in its ability to express semantic constructs. Most notably, it doesn't allow expression of constraints. Moreover, it has few descriptors to make extensive inferences. Thus it is not really expressive enough to support the Semantic Web.

As part of its IDEF series of notations, the Federal Government has sponsored the creation of IDEF5, an ontology expression graphical language. A report describing it was published in 1994[5], but it has gotten little publicity since then. The report is an excellent overview of ontological topics, and the approach is very thorough.

Since IDEF5, like data modeling, is a graphical approach to ontological modeling, however, it does not serve the purposes of the Semantic Web. The ***Web Ontology Language*** (OWL)* was developed to provide a syntax that can be understood directly by computers. OWL builds on RDF and RDFS, and like them, it is constructed from XML tags.

There are actually three versions of OWL: OWL Lite, OWL DL, and OWL Full. The nuances of the differences among these are beyond the scope of this paper, so we will focus on OWL DL.

Remembering the differences we described above between data modeling and ontology languages, it is important to reiterate that the structures to be built using OWL are not restrictive. The open world assertion applies. Beginning with a series of instances and a series of assertions, the assumption is that anything can be true unless asserted otherwise.

(Also, accept the fact that OWL is primarily intended to be understood by computers, not people. Hence the discussions which follow will seem pretty arcane to the graphically-motivated among us.)

For example, in an airports database, you might have something called "AA243". Absent any other information, that could refer to either a flight or an airport. After you have asserted that it is in fact a flight number, you have *not ruled out* that it might also refer to an airport. To prevent that, you must explicitly declare that the class of airports and the class of flights is ***disjoint.*** That is, the same thing cannot be an instance of both classes.

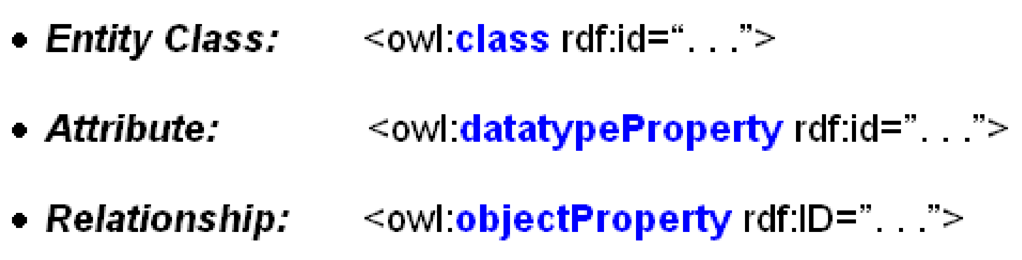
This open assumption is significant, because, given a large amount of data from disparate systems, it is possible that a computerized analysis along these lines might show up things that people would never figure out. To be sure, there will be a lot of nonsense assertions initially, until people learn to clarify the rules, but even that exercise will be useful in helping people better understand their data. This allows data from systems where the language is not quite consistent to at least be viewed collectively, and, with luck, those inconsistencies themselves will become clear.

The idea is to begin with instances and classify them by their properties. Your author's son showed an early propensity for philosophy when, at the age of three, he decided to build a collection of red things. First he sorted his toy trucks, his action figures, and his other toys to gather together the red ones. Then he went around the house to find his mother's lipstick, one of his father's shirts, and various other (red) paraphernalia.

This is what OWL does. Among other things, it classifies things by their properties.

OWL and Data Modeling

The OWL tags that correspond to data modeling constructs include the following:



- ***Entity Class:*** <owl:**class** rdf:id=“. . .”>
- ***Attribute:*** <owl:**datatypeProperty** rdf:id=”. . .”>
- ***Relationship:*** <owl:**objectProperty** rdf:ID=”. . .”>

Note that, as with RDF, both attributes and relationships are ***properties*** that must be defined first before being attached to classes.

For example, here are two classes:

```
<owl:class rdf:ID="Class1">

<owl:class rdf:ID="Class3">
```

The attribute "Attr5" is in fact an attribute of both of these classes:

```
<owl:DatatypeProperty rdf:ID="ATTR5">
   <rdf:Type rdf:resource="owl:#FunctionalProperty"/>
   <rdfs:domain rdf:resource="#Class1"/>
   <rdfs:domain rdf:resource="#Class3"/>
</owl:DatatypeProperty>
```

That the DatatypeProperty is a "functionalProperty" means that the property can have no more than one value in each domain. It is not required to have any values. This is appropriate for attributes in a relational environment, which are not allowed to have more than one value. For relationships, this is useful, although since it does not require a value, it is not adequate for the full range of cardinality issues. There are other problems with it in relationships, which we will discuss below.

RDF had a tag "type" that was supposed to allow you to specify instances of classes, but since it didn't have classes, that didn't make much sense. Now that OWL has classes, we can talk about instances of classes. For example:

```
<owl:Class rdf:ID="Ship"/>

<owl:Thing rdf:ID="Titanic">

   <rdf:type rdf:resource="#Ship">

</owl:Thing>
```

There are two approaches to specifying relationships in OWL.

In the first case, an ObjectProperty is simply defined, where the domain and range are part of the definition. For example:

```
<owl:Class rdf:ID="City"/>

<owl:Class rdf:ID="State"/>

<owl:ObjectProperty rdf:ID="located_in_state">
    <rdfs:domain rdf:resource="#City"/>
    <rdfs:range rdf:resource="#State"/>

</owl:ObjectProperty>
```

Thus since the domain and range are part of the definition of the object property, the name must be unique to this entity class pair. You cannot assign this relationship to any other class pair.

An alternative is to define the property without specifying a domain and range. In this case, you then define a class as being a sub-class of a restriction that applies the property. In this case many class pairs can use the same object property. For example:

```
<owl:ObjectProperty rdf:ID="located_In"/>

<owl:Class rdf:ID="City">
   <rdfs:subClassOf>

         <owl:restriction>

             <owl:onProperty rdf:resouce="located_In"/>

             <owl:someValuesFrom rdf:resource="State"/>

         </owl:restriction>
   </rdfs:subClassOf>

</owl:Class>
```

So, it is possible to convert an ontology represented by a data model into one represented by an ontology language. The model assumes constraints we don't normally realized (like disjointedness), and it will be important to introduce any business rules we've identified as well.

Summary

Data modeling, database design, and business rule modeling are all part of a particular way of looking at the world. The semantic web and the ontology languages that support it are part of a new way of looking at the world. The differences are in terms of premises, the way classes are identified, and the implications of constraints.

- ***Premises***
  - Data modeling, etc.
  - "Closed" world
  - Only what is asserted is true
  - Ontology languages
  - "Open" world
  - Anything may be true if it does not conflict with assertions
- ***Approach to classes***
  - Data modeling, etc.
  - Begin with class definitions of fundamental categories
  - Define attributes
  - Identify sample instances
  - Ontology languages
  - Begin with instances
  - Identify attributes
  - Define classes based on attributes
- ***Constraints and business rules***
  - Data modeling, etc.
  - Determine what data are acceptable
  - Reject data that do not conform
  - Ontology languages
  - Assert what is known to be true
  - Infer what else may be true.

As an example, consider the typical data modeling assertion:

Each CITY must be located in *one and only one STATE.*

To a data modeler, this implies the following:

1. If "Portland" is entered as a CITY without a STATE identified, it is not acceptable.
2. If "Portland" is entered as a CITY and located in STATE "Maine", then a record with the CITY "Portland" located in state "Oregon" is not accepted.

To an ontologist, however, this implies the following:

1. "Portland" may be entered without specifying the STATE.
2. If "Portland" is entered "located in" "Maine", and "Portland" is identified as a CITY then "Maine" must be a STATE.
3. If, in addition to statement 2., "Portland" is entered as "located in" "Oregon", then "Oregon" must be a STATE, and either:
    - "Oregon" and "Maine" must be two names referring to the same STATE, or
    - The CITY referred to by the name "Portland" in "Oregon" must be a *different* CITY than the one referred to by the name "Portland" in "Maine".

Interesting, yes? For an example of converting a data model to OWL, tune in next quarter for the next article on this subject.

---

[1] G. Kemmerling, *Philosophical Dictionary*, 2002.

[2] Ibid., http://www.philosophypages.com/dy/o.htm#onty.

[3] Tim Berners-Lee, *Weaving the Web*. Harper, San Francisco. 1999.

[4] Michael C. Daconta, Leo J. Obrst, Keven T. Smith, *The Semantic Web*. Wiley, Indianapolis. 2003.

[5] Knowledge Based Systems, Inc. *IDEF5 Method Report*. Prepared for Armstrong Laboratory AL/HRGA Wright-Patterson Air Force Base, Ohio. Can be found at http://www.idef.com/pdf/Idef5.pdf.

* You may wonder why the "Web Ontology Language" has the acronym "OWL". It seems that In *Winnie the Pooh*, Owl imagines that his name is spelled "WOL", until his friends correct him. Here, the World Wide Web Consortium (W3C) decided to start with the correct spelling.

Go to Current Issue | Go to Issue Archive

Recent articles by David C. Hay

- From Data Modeling to Ontologies: Discovering What Exists, Part 2

- From Data Modeling to Ontologies: Discovering What Exists, Part 1
- Book Review: Principles of Data Management:

David C. Hay - In the information industry since it was called “data processing,” Dave Hay has been producing data models to support strategic and requirements planning for more than twenty-five years. As President of Essential Strategies, Inc. for nearly twenty of those years, Dave has worked in a variety of industries including, among others, banking, clinical pharmaceutical research, broadcasting, and all aspects of oil production and processing. Projects entailed various aspects of defining corporate information architecture, identifying requirements, and planning strategies for the implementation of new systems.

Dave’s recently published book, *Enterprise Model Patterns: Describing the World*, is an “upper ontology” consisting of a comprehensive model of any enterprise from several levels of abstraction. It is the successor to his groundbreaking 1995 book, *Data Model Patterns: Conventions of Thought* – the original book describing standard data model configurations for standard business situations.

In between, he has written *Requirements Analysis: From Business Views to Architecture* (2003) and *Data Model Patterns: A Metadata Map* (2006). Since he took the unusual step of using UML in the *Enterprise Model Patterns…* book, a follow-on book, *UML and Data Modeling: A Reconciliation* was published later in 2011. This book both shows data modelers how to adapt the UML notation to their purposes, and UML modelers how to adapt UML to produce business-oriented architectural models.

Dave has spoken at numerous international and local DAMA, semantics, and other conferences as well as at various user group meetings. He can be reached at dch@essentialstrategies.com, (713) 464-8316, or via his company's website.