Object Database on Top of the Semantic Web

Jakub Güttner

Graduate Student, Brno Univ. of Technology, Faculty of Information Technology, Czech Republic
guttner@fit.vutbr.cz

Abstract

This article compares the structure of Semantic Web RDF (Resource Description Framework) to a worldwide distributed database. Reasons are given for treating such database as an object-oriented one. The article lists the similarities and differences between RDF and object databases, gives reasons for extracting object data from RDF and shows the structures that can be discovered in RDF graphs, discusses and unifies different approaches for querying and navigating object-oriented and RDF graphs and builds a simple formal model of an object-oriented DB on top of model-theoretic RDF semantics.

1. Introduction

Although the World Wide Web is practical and readable for humans, computers cannot process its semantics. For them, it is hard to tell that a document is a CV, which of the numbers contained in it is the date of birth and which link leads to the company where the person works.

The Semantic Web tries to address this problem and store information in a more organized way. This article shows that such effort is basically building a worldwide database, compares such database to object-oriented databases (OODBs), shows how to extract object data from it and gives an example of building the semantics for a simple OODB on top of the Semantic Web Resource Description Framework.

The following section, Semantic Web and Objects, shows the similarities between these two concepts, explains how an object database is suitable for storing the corresponding information and why it’s useful to extract objects from the Semantic Web.

Third section, Graph-based access, shows how the Semantic Web and object structures differ in the way of obtaining information from graphs, and suggests how to reconcile the two approaches.

Fourth section, An Object Database Over the Semantic Web, gives a summary of a simple model-theoretical object database definition over RDF-based Semantic Web data.

2. Semantic Web and objects

2.1 The vision of the Semantic Web

The Semantic Web is a W3C activity that aims at extending the current World Wide Web with information that has well-defined meaning both to humans and computers. This would create an environment where “software agents roaming from page to page can readily carry out sophisticated tasks for users” [1]. The way to reach this goal lies in inference rules for the agents and presenting information in structured collections in contrast to today’s WWW pages that only define formatting of text, not its meaning.

At the core of the Semantic Web lies the RDF — Resource Description Framework [2] that stores information in graphs where each edge represents a binary predicate (figure 1). For uniqueness across the whole Web, both edges and nodes are labeled with URI references (urirefs). RDF and RDFS (RDF Schema) also provide other features like a collection vocabulary, typing and subclassing, anonymous nodes and elementary datatypes. Semantics of new urirefs can be formally
2.2 Semantic Web as a database

"The Resource Description Framework (RDF) is a framework for representing information in the Web." [2]

RDF and the Semantic Web define how to store information. Although the Semantic Web is mostly mentioned from the viewpoint of knowledge management and artificial intelligence, managing a body of information is a typical database problem. From this perspective, the Semantic Web can and should be viewed as a worldwide database. Of course it is widely distributed and not centrally managed, often incomplete or inconsistent and very loose in its format — but it still is a database, albeit not a relational one. There are, however, other types of databases as well, ones whose structure is very close to the graph nature of the Web.

Figure 2. Comparing OODBs and the Semantic Web

An object-oriented database and the Semantic Web share several important concepts. Adding agents to the Semantic Web is then similar to adding deductive principles to an OODB. Some of the main similarities in structure and concepts are:

**Unique identifiers** — A strong requirement for every OODB are a unique identifiers (OID). In today’s setting that means unique across the whole Internet, which is exactly what URI references do. Using RDF urirefs for OIDs would establish an understanding of the Semantic Web as a worldwide distributed database.

**Graph-theoretic foundations** — RDF uses model theory to interpret graphs. OODBs still do not have a common theoretical foundation [3], [4], but many models use graph theory, combining it with set theory [5], [6] or extending into category theory [7], [4], [8]. Graph theory in OODBs is suitable for modeling relationships between objects or inheritance hierarchies. Using a shared foundation for both RDF and object databases would connect these areas very closely.

**Description logics** — The RDF is built on existentially quantified first order logic. In deductive object-oriented databases, description logics (F-logic, Transaction logic, HiLog [9]) are used for describing a database schema. This is yet another formalism that connects the two together.

Other similarities between the Semantic Web and OODBs are suggested in figure 2 together with examples of specific areas where these two could enrich each other in the future.

2.3 Extracting objects from the Semantic Web

Why extract objects from the Semantic Web?

**Application of algorithms** — Many programming languages are object-oriented. It would be convenient to present them with Semantic Web data in the form of objects. Moreover, strongly typed objects present correct data to algorithms without the need for explicit checking of their structure.

**Integrating database concepts** — Semantic Web data processed as an OODB could use some of the well-developed database concepts like indexing, access control or query languages (see figure 2).

**Efficient storage** — Today the Semantic Web still works in small and manageable scale and there are no serious performance problems. In the future, the performance of physical storage of RDF data may become increasingly important. Looking at RDF data through the object database paradigm allows the use of common object storage techniques.

Below we list the basic features of an OODB according to the ODMG standard [10] and G2 Concept Definition Language [3]. In the final section of this article, we build a formal OODB model on top of RDF semantics that captures all of these areas. Here each of them is explained and its RDF counterpart is mentioned.

**Objects** — Everything in an OODB is stored in objects. An object has a unique OID (URI in RDF) and it contains either a tuple or a collection of attributes or references. All RDF nodes with a uriref can be considered objects.

**Datatypes** — Most OODB models have a set of elementary types that are used to construct composite types. These types cannot be further decomposed and their semantics is fixed. This is equivalent to RDF T-interpretations; one of the most commonly used sets of datatypes is XML Schema.

**Attributes and relationships** — A collection or tuple can either contain or reference a value. This is important for modeling, physical data storage and update semantics. When the value of an attribute is a uriref, it is...
referenced, and when the value is either a datatype or a blank node (a tuple or a collection without a uriref) with one reference to it, its value is embedded within the parent object.

**Tuples** — A tuple has attributes labeled by property urirefs. Every blank RDF node that has at least one non-collection attribute can be considered a tuple. Adding a uriref to the node makes it a tuple object.

**Collections** — RDF has collection vocabulary without any formal semantic restrictions. All its objects need to have the same type. Any blank RDF node with collection attributes (e.g., rdf:_1) is a collection and adding a uriref to the node makes it a full object. RDF can have objects that act as both tuples and collections, similarly to G2 CDL [3], and the model takes it into account.

**Types** — In most OODBs, an object has exactly one type that specifies its internal structure. However, a RDF node can have no types or multiple types. Having no type is equivalent to having the most general type ("Any") and having multiple types is similar to the concept of multiple inheritance and object roles. Nodes that conform to the structure prescribed by all of their types (through domain and range properties of object attributes) are labeled as "strongly typed" in the model.

**Inheritance** — In both RDF and OODBs, inheritance is a fundamental tool for constructing new concepts. The RDF definition of inheritance as a subset relationship on elements of the domain of discourse is suitable for its database counterpart, because it allows multiple inheritance and preserves the notion of "strongly typed" objects.

With these guidelines an object-oriented structure can be extracted from any RDF graph. Depending on the graph, the form of the result can either be quite loose or strongly typed.

The following section discusses some difficulties in navigating and querying RDF graphs as OODBs, while the final section gives an example of a formal OODB model built on top of RDF semantics.

### 3. Graph-based access

#### 3.1 Navigating the Semantic Web graph

For the purposes of reasoning, a RDF graph is often perceived as a set of facts (triples — binary predicates). When retrieving information for deductions using first order logic, the most common query is finding a set of triples with a constant predicate — in Prolog syntax, this would be something like parent(X,oid1) or supervises(X,Y). From the viewpoint of retrieving data, the access point into a RDF graph is the label of an edge, therefore the graph does not need to be connected and it is not important whether all its nodes are reachable. This approach implies that from the viewpoint of physical data organization, a knowledge base is physically grouped by predicates.

While this is common from the deductive point of view, it presents an obstacle for viewing the RDF graph as an object database.

### 3.2 Navigating an object database

The navigation in an object database is different. In an OODB, data are physically grouped by objects. An object is accessed as a whole; it is unusual to retrieve all occurrences of a given attribute. In Prolog syntax, accessing the whole object can be expressed as X(oid1,Y). The main access point into the database is the label of a node, typically an extent — an object that stores a collection of instances of a given type.

The practical result is that an object database is navigated by traversing edges of its graph. In contrast to the Semantic Web approach, the direction of an edge is important and the whole graph must be reachable from the access points.

#### 3.3 A common access model

To access the RDF graph as an object database, the whole graph needs to be reachable from certain access points. Figure 3 shows part of a RDF graph ("Peter likes cabbage"). To make such graph fully reachable, three changes (shown in gray) to the original structure need to be made:

![Figure 3. Reachability in part of a RDF graph](image)

Adding extents of types. In a typical OODB, every object needs to be part of at least one extent. All extents are collections directly reachable from the system catalog (a single access point into the database), which in effect makes all the objects in the database accessible. In RDF setting, extents could actually be identified with types (oodb:PersonExtent and my:Person).
Reversing the direction of rdfs:domain. When finding out information about a certain type, it is useful to ask a question like X(my:Person,Y) to find out all the attributes of my:Person and their types. For this reason, it would be useful to define a predicate that has the same meaning as rdfs:domain, but its direction is reversed (see the curved gray arrow in figure 3).

Connecting edges to corresponding nodes. In RDF, it is natural that an object can act both as a predicate and as subject/object. To find out information about an attribute in an object database, one can usually examine the type of this object. However, the RDF model does allow extra attributes, therefore it should be possible to find the specifics of an attribute elsewhere. In figure 3, straight gray arrow indicates a new connection that needs to be made.

Traversing from access points together with removing property-driven random access into the graph certainly limit the freedom of finding data in the RDF graph, but a big advantage is that the data can be physically organized by objects and the usual OODB optimizations can be applied to things like attribute storage or type information.

4. An object database over RDF

The RDF standard is open to further extensions. New sets of URI references (vocabularies) can get formal meaning either by making statements about them using already defined predicates (like rdf:Bag or rdfs:subPropertyOf) or by redefining their model-theoretic interpretation.

This section presents direct model-theoretic semantics of a basic vocabulary that supports the elementary notions of an object-oriented database (as defined in sources like [10], [11], [5], [3]). A "SODA" prefix (Semantic Object-oriented DAtabase) is used for this vocabulary. With these extensions, object structures can be machine-entailed from arbitrary RDF graphs. This model can also be used for later implementation of OODB on top of RDF with automatic consistency checking using only the means supplied by RDF and for populating the database with RDF data.

4.1 Basic RDF semantics

Description of RDF semantics [2] states that for a set of URI references (a vocabulary) V that includes the RDF and RDFS vocabularies, and for a datatype theory T, a T-interpretation of V is a tuple I = < IR, IP, IEXT, IS, IL, LV > (which stands for Resources, Properties, Extensions, Semantic map, Literal map and Literal Values).

IR, universe, is a set of semantic images of urirefs from V IP is subset of IR. It contains the images of the properties from V — urirefs that label edges of RDF graphs

IEXT: IP → IR×IR defines property extensions. It gives all object-subject pairs that make a property from V true.

IS: V → IR assigns semantic images in the universe to urirefs.

IL, is a mapping from the set of typed literals to the set LV, a superset of all untyped literals and a subset of IR.

The RDF vocabulary interpretation also defines class extensions — for every class c, ICEXT(c) = {x ∈ IR | <x,c> ∈ IEXT (IS(rdf:type))}. The rdf: prefix is connected to http://www.w3.org/1999/02/22-rdf-syntax-ns# and rdfs: then stands for http://www.w3.org/2000/01/rdf-schema#. Apart from the RDF and RDFS requirements, there are several more things that need to be true for every T-interpretation.

4.2 Obtaining the OODB graph

RDF is monotonic, which means that adding edges to a RDF graph cannot change the validity of previous entailments — anyone can publish RDF statements without harming other people's entailments. However, in some cases it is useful to "close" the RDF graph to additions. Otherwise, one thing that would not be possible is deciding whether an instance of an object structurally conforms to its type since the definition of a type cannot be guaranteed to be complete.

4.3 Part of model-theoretic OODB semantics

There is not enough space here to present the full definition of the model including a comprehensive overview of RDF semantics — only several sample definitions and features of the model are given. The full model is given in [12].

Figure 4. Division of the universe

SODA vocabulary: {soda:Concept, soda:Class, soda:Tuple, soda:Meta, soda:Thing, soda:Collection, soda:Attribute, soda:MemberAttribute} ∈ IC;
{soda:collectionOf, soda:tipo, soda:subTypeOf, soda:member, soda:_1, soda:_2,...} ∈ IP. Most of the SODA vocabulary is subclassed from RDF with domain and range restrictions.
A SODA-interpretation is a T-interpretation that makes further conditions (and T-interpreted triples) true. The universe is divided into disjoint sets like in strongly typed programming languages — see figure 4.

Blank nodes cannot represent class instances since they do not have a universal OID (uriref).

\[ \text{xx is blank node} \land \chi \in A \land (\chi, y) \in E \land \text{ICEXT}(\text{IS(soda:Tuple)}) \lor \text{ICEXT}(\text{IS(soda:Collection)}) \]

Strong typing: A tuple must have all the attributes that its type prescribes with an exception of undefined reference.

\[ x \in \text{IEXT}(\text{IS(soda:Class)}) \lor \text{IEXT}(\text{IS(soda:Tuple)}) \land \\
\exist \chi, \gamma \geq 1, x_1 \ldots x_n, a_1 \ldots a_n:
\]

Data recursion: Collections, tuples and literals need to avoid being embedded in each other. Since only one attribute in the graph refers to them, there needs to be a path from some class concept to these concepts (fig. 5).

\[ x \in \text{IEXT}(\text{IS(soda:Collection)}) \lor \text{IEXT}(\text{IS(soda:Tuple)}) \land \\
\exist a_1 \in \text{IEXT}(\text{IS(rdfs:domain)}) \land \\
\exist \chi, \gamma \geq 1, x_1 \ldots x_n, a_1 \ldots a_n:
\]

The resulting vocabulary contains the most important concepts from the area of OODB as specified by ODMG, O2 or Matisse projects (see sources). Moreover, the model has very close correspondence to the CDL (Concept Definition Language) of the G2 database [3] and semantic nuances of the vocabulary definition were formed according to this system.

The article showed the equivalents of fundamental OODB features in RDF graphs. Different ways of navigating the data graphs were presented and some guidelines for reconciling the were given. The article also presented part of a formal OODB model built on top of the RDF semantics that takes the RDF specifics into account.

Future work can further refine the model and bring together different aspects of the Semantic Web and OODBs that are currently used only in one of the areas (see figure 2).

**5. Conclusion**

The Semantic Web with its graph structure and unique identifiers is very similar to an object database in both structure and philosophy. It is useful to extract object data from RDF graphs for algorithm application, using database-oriented data handling and more efficient storage.

The article showed the equivalents of fundamental OODB features in RDF graphs. Different ways of navigating the data graphs were presented and some guidelines for reconciling the were given. The article also presented part of a formal OODB model built on top of the RDF semantics that takes the RDF specifics into account.

This work was supported by the CEZ:J22/98: 262200012 project “Research in Information and Control Systems” and the grants GACR 102/01/1485 “Environment for Development, Modelling, and Application of Heterogeneous Systems” and FRVŠ FR828/03/G1 “Dynamic Object Model in Interpreted Systems”.

**6. References**


