

# Semantic Web and Databases: Relationships and some Open Problems

Stefan Decker

*Gates Bldg 4A/425*

*Stanford University, Stanford, CA, 94306, USA*

*stefan@db.stanford.edu*

**Abstract.** In this position paper I describe the relationship of the database research field “semi-structured data” to the Semantic Web, and describe two different yet unsolved problems. The first problem is the handling of the semantics of modeling languages like UML, TopicMops or RDF Schema within query and storage infrastructures for semi-structured data. A system is required which enables to query semi-structured datasets with different semantics, since creating specialized systems for each dataset recreates the Tower of Babel again that semi-structured data was hoping to overcome. The second problem described is the versioning of semi-structured data, especially for ontology modeling languages. Ontologies are social artifacts constructed within collaborative agreement processes. So far support tools to reduce the cost of ontology development and maintenance are missing, but are paramount for the emergent Semantic Web.

## 1 Introduction

The representation of knowledge (in the sense of machine-understandable data) with semantics on the Web poses new requirements for the supporting standards and infrastructures: the Web has characteristics, like the inherent distribution, diversity and heterogeneity that need to be addressed.

In this position paper I identify requirements for the Semantic Web, link it to existing database technologies and open research questions.

## 2 Requirements for a Data and Knowledge Representation on the Web

In this section, I identify properties of a data model for the Web that may serve as a foundation for knowledge representation tasks.

1. *Data Identifier point to exactly one data item.* Giving the same ID to different data items leads to confusion and conflicts (identification errors of the first kind). Simple naming schemes (like using „JohnSmith002“) for an object in a local database don’t work on the Web, since it is likely that independently somebody else has adapted the same naming scheme. Please note that the opposite problem – that one data object is assigned two or more object Ids (identification errors of the second kind), is less of a

threat – actually, it is inevitable. Errors of the second kind arise from a vague world, where items are described instead of uniquely named.

2. *Data representation must be simple and extensible.* Because of the diversity of communities using the Web and different requirements from each of those communities data on the Web is heterogeneous and often unstructured. There will be no universal usable structured domain specific knowledge and data representation language. Instead, a knowledge representation language has to provide a basis for more specialized languages, such that each community can build what it needs. On the other hand, the benefit of the Web and the Internet is the ease of communication. Although different communities have different requirements for the representation of information, they want to communicate with each other. Thus, the data model has to support information integration and articulation. The data model serves as a foundation for other languages for all communities that exist on the Web. Therefore, the language needs to be simple and acceptable for a variety of user communities, and extensible such that the user communities are able to adapt the language to their needs without compromising the entire approach and making it impossible for others to understand their data.
3. *Data on the Web is distributed.* Data on the Web might be distributed on several locations. Therefore, it should be possible to link to other sources. It should be possible to link to other data resources, and to create a web of machine-processable data. This resembles very much the idea of hyperlinks in HTML pages.
4. *Data on the Web is biased.* There is no such thing as universal truth. Anybody can say anything on the Web. The freedom to express anything is already criminally exploited, say, with forged stock market news. Therefore, an important requirement for representing data is the ability to say something about foreign data (especially about data residing on other servers). For example, whether it is believable, supported, wrong, biased, etc. The requirement for a representation language is that the language needs facilities for expressing metadata about metadata.
5. *Support for syntactic interoperability.* By syntactic interoperability, we mean how easy it is to read the data and get a representation that can be exploited by applications. For example, software components like parsers or query APIs should be as reusable as possible among different applications. Syntactic interoperability is high when the parsers and APIs needed to manipulate the data are readily available and reusable.
6. *Support for semantic interoperability.* Usually data on the Web comes with some semantics attached. With semantic interoperability, we mean the difficulty of understanding foreign data. Please note the difference from syntactic interoperability: syntactic interoperability talks about parsing the data, while semantic interoperability means to define mappings between unknown terms and known terms in the data. We regard this requirement as one of the most important issues, since the cost of establishing semantic interoperation is usually higher than the cost for establishing syntactic interoperation due to the need for content analysis.
7. *Universal expressive power.* The heterogeneity of information on the Web cannot be captured by a fixed schema, since the schema would impose limitations on the expansibility. These limitations are not acceptable on the Web, where the knowledge representation tasks posed by different communities cannot be foreseen. Instead, the information model has to be self-describing, capable of representing arbitrary information, but structured enough to support an effective query processing. Clearly, this requirements limits the level of support that is possible: the more domain specific a language is, the more it is possible to support the language with tools and technologies. The challenge for the Semantic Web is to find a data format that does not restrict

anyone (thus is by no means domain specific), and allows building an effective and efficient infrastructure that users as well as software developers can reuse.

The standards supporting the Semantic Web have to deal with the listed requirements, but fortunately, as argued by the database community, *semi-structured data* is particularly suitable for solving some the problems described in the previous section (cf. [Abiteboul, 1997], [Arbiteboul et al, 2000] for an elaborated introduction).

### 3 Semi-structured data and Web standards

The standards-backbone of the Semantic Web is often presented as a layer cake similar to the one given in figure 1. The figure represents a layering of (partially future) W3C recommendations, similar to the layering of protocols of the ISO/OSI network model. The sole purpose of introducing a layering is to reduce the complexity of the overall task: creating and maintaining a single layer in the layer cake is simpler than solving the complete standards task of creating the Semantic Web.

It turned out that the RDF (Resource Description Framework) datamodel (see [Lassila & Swick, 1999]), the third layer of the layer cake displayed in Figure 1, is almost identical to the notion of semi-structured data developed by the database community. Roughly speaking, semi-structured data is data that is neither raw data, nor very strictly typed as in conventional database systems. Graph-based models for semi-structured data have been defined, a popular example is OEM (Object Exchange Model) (cf. [Goldman et al., 1996], [Suciu, 1998]) and was developed for integration tasks in the Stanford TSIMMIS project [Garcia-Molina et al., 1995]. The differences between RDF and OEM can be summarized as follows:

- RDF and OEM both are based on labeled directed graphs - both rely on the same graph data structure.
- OEM does not specify how the OIDs are generated, whereas RDF requires them to be URIs. RDF edges are labeled with URIs, RDF nodes are either labeled with URIs, literals (Strings or XML markup) or blank nodes. OEM supports types like integer etc.
- RDF does not require the declaration of root objects as OEM does.

Since both data formats are very similar, the technologies that were developed for semi-structured data (like OEM) are directly applicable to RDF (e.g., storage infrastructure like LORE [Abiteboul et al., 1997a] or techniques for establishing interoperation [Abiteboul et al., 1997b], [Papakonstantinou et al., 1996]). RDF already fulfills some of the requirements posed: URIs realize globally unique identifier, which are used to uniquely identify data items, which are possibly distributed on the Web. The graph structure is simple and extensible, and APIs provide a level of syntactic interoperability.

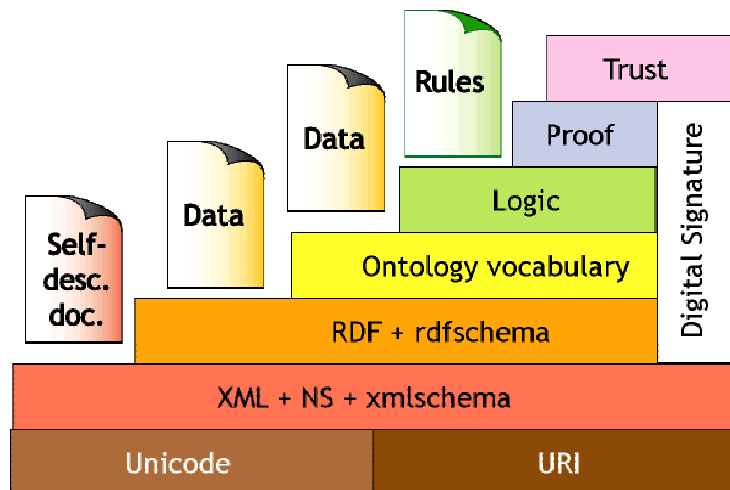


Figure 0 Layer cake of W3C recommendations (taken from <http://www.w3.org/Talks/2002/01/10-video/slide17-0.html>)

#### 4 Open problems in semi-structured data research

Although semi-structured data has been investigated for a number of years, some issues have not been investigated in depth yet.

##### *Multiple Resource Semantics*

On the Semantic Web many different communities are publishing their formal data, and it is unlikely that established data models for representing this data will disappear. Examples of already established data models include UML, Topic Maps, RDF Schema, Entity Relationship Models, DAML+OIL, and more, highly specialized data models. Integrating data based on these different data models has proven to be an error-prone and expensive task: Different storage and query engines have to be combined into one program, and data has to be translated constantly from one representation to another. A first step to improve this situation is the use of RDF as a common representation formalism for all data involved.<sup>1</sup> Representing different kinds of schema information within semi-structured data has been neglected within the database community. The schema approaches listed in [Abiteboul et al 2000] for semi-structured data don't take existing modeling approaches like UML into account and do not investigate systematically how to represent those modeling languages within semi-structured data. The problem is now being extensively investigated within the OIL, DAML+OIL, and W3C Web Ontology efforts, but a systematic generic investigation of possible problems and potential solutions is still missing.

But providing modeling guidance for how to represent a modeling language within semi-structured data does not solve the problem of using those modeling languages within semi-structured data entirely, since it resolves just the *syntactic interoperability problem* mentioned in section 2. To deploy the data, storage and query systems are required such that the different datasets can be queried with different semantics in mind. For example, to query a dataset that is formulated using UML with UML semantics, and another dataset that represents an RDF Schema ontology, different query mechanisms have to be used. Deploying different datasets with different semantics using different, specialized query and storage systems creates a huge semantic interoperability problem, and recreates the Tower

<sup>1</sup> See <http://www-db.stanford.edu/~melnik/rdf/uml/> for a representation of UML in RDF and [Lacher & Decker, 2002] for a representation of TopicMaps in RDF.

of Babel, that RDF was hoping to overcome. Although many query languages and inference engines for RDF exist (e.g., SiLRI [Decker et al, 1998], RQL [Karvounarakis et al 2001], SQUISH<sup>2</sup>) none of them is capable of representing multiple semantics as required by the different heterogeneous data models. E.g., when querying UML data the *Generalization* relation should be treated as a transitive relationship, as well as *rdfs:subClassOf* in RDF Schema. None of the cited query languages have the abilities to query different data models with different kind of semantics. Although some of them (RQL and SiLRI) have a built-in semantics for RDF Schema, this does not generalize to other data models. A first step to remedy this situation has been done in the TRIPLE system [Sintek & Decker, 2002], but further investigations are necessary to make this approach scalable.

Approaches like Lore [Abiteboul et al, 1997A] provide initial insights how to store and query semi-structured data efficiently within secondary storage, and current approaches for XML storing and querying XML [Bayer, 2001] might be extensible.

### ***Version Management of Ontologies***

While solving technical questions (especially from a Computer Science perspective) are important from the Semantic Web, their motivation comes from social problems. Already the Web is more a social phenomenon than a technical one – while having a technical infrastructure was very important, the willingness to share information made the Web the success it is.

For the Semantic Web social processes are even more important: before data can be shared by machines, shared *Ontologies* [c.f. [Mena et al, 1998] need to be defined. Ontologies are the result of agreement processes, which take a lot of effort and time. Agreeing on and later on maintaining an Ontology within a worldwide distributed group of people is a challenging task – even more as currently almost no tools are available to support the agreement or support processes.

Especially the versioning problem is interesting from a database perspective.

Initial ontologies will change over time due to changing environmental conditions, new knowledge about the world and so on. Maintaining different versions of an ontology will become necessary to support collaborative development and to provide different extensions of an existing ontology for different communities. Ontologies that change cause the following problems:

1. Data, applications, and other ontologies that conform to the ontology might become inaccessible, unusable, or inconsistent after the ontology evolved.
2. Managing different versions and branches of an ontology is a laborious task that is a bottleneck for collaborative development of ontologies.

Most of the literature in ontology evolution (c.f., [Heflin et al, 1999] [Klein & Fensel, 2001]) and database-schema evolution [Roddick, 1995] [Sciore, 1991] [Franconi et al, 2000] [Lu et al, 1996] focuses on the first problem. However, for jointly building ontologies, the second problem is more important than the first one: the effort of jointly managing and merging different versions of an ontology greatly increases the cost of establishing a joint ontology.

The problem of managing multiple versions is not unique to ontology management: this problem also arose in software engineering in the context of collaborative, parallel development of large software systems. A simple locking mechanism did not improve the situation, since developers tend to forget to unlock files after they are done, essentially eliminating parallel development. CVS (Concurrent Versions System), which is based on

---

<sup>2</sup> See <http://ilrt.org/discovery/2000/10/swsq1/>.

RCS (Revision Control System) [Tichy, 1985], and similar systems became an effective solution for software evolution and management. The collaborative internet-based open-source software-development efforts are unthinkable without CVS-based support.

To enable incremental improvement, collaborative work, and effective management of different versions of ontologies, an Ontology Versioning System (OVS) is required. This system will answer structural queries such as *Which subclasses of Person were modified in the last month?* To answer structural queries, OVS has to understand the semantics of the ontology representation language (e.g., the semantics of subclass). Unfortunately, CVS or other software-evolution systems cannot be used directly<sup>3</sup>. These systems rely on textual representation of information. While focusing on the textual representation works for text documents and the source of software, it does not provide sufficient support for the joint development of ontologies. Comparing textual serialization of ontologies is not sufficient since two ontologies may have completely different textual representations and still have the same semantics.

Since semi-structured data is the basis for many different data-model, which require versioning support, a base layer for versioning should be built upon semi-structured data. Open questions regarding an OVS are:

1. Which primitives are required to support the collaborative development of ontologies? Examples of such primitives include tagging a certain version with names, the ability to undo changes of certain users, the ability to enable social processes (e.g., voting for the introduction of a new concept), and so on. Other operations include deriving different branches from an existing ontology, merging two different branches, updating a branch and so on. [Lu et al, 1996] sketch the use of a temporal object-oriented database for version control of object-oriented schemata. This work might be usable and extensible with different underlying representations, support collaboration primitives, and allow integration of different ontology versions. The primitives of CVS and RCS deliver a good starting point.
2. How are different versions of the same ontology integrated again? Since the same ontology will be changed by different groups of users independently, there will be a frequent need to reintegrate different versions. Model management and match operation provide possible solutions here.

An ontology versioning tool as suggested here will be crucial for the growing deployment of evolution of the Semantic Web, since it helps to make ontology development and maintenance more cost effective.

## 5 Conclusions

I presented two problems which extend the research on semi-structured data, and which are relevant for the Semantic Web. These problems are handling modeling languages with multiple semantics within semi-structured data, and versioning of ontologies to enables collaborative development. I described only a limited set of research problems out of many more that are of interest within the database community. Examples of other problems include the ability to *declaratively describe and combine webservices* with respect to security and transactions.

Another, already active field important for the Semantic Web is *Model Management* [Bernstein et al, 2000], which deals with the systematic management of schema information in an algebraic way.

---

<sup>3</sup> Actually, the GenOntology Consortium is using CVS for Ontology Maintenance due to the lack of better tools, which clearly proves the need (Daniel Rubin, Personal Communication, Feb. 2002).

## ***Acknowledgement***

Many of the here presented ideas and problems originated from joint work and contributions from Natasha Friedman-Noy, Sergey Melnik, Prasenjit Mitra, Michael Sintek, and Gio Wiederhold.

## **References**

- [ABITEBOUL, 1997] S. Abiteboul: Querying semi-structured data. In: Proceedings of the International Conference on Database Theory (ICDT), Delphy, Greece, 1997.
- [ABITEBOUL ET AL., 1997A] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener: The Lorel query language for semistructured data. In: The Lorel query language for semistructured data. In: *Intl. Journal on Digital Libraries*, 1(1);68-88, April 1997.
- [ABITEBOUL ET AL., 1997B] S. Abiteboul, S. Cluet, T. Milo: Correspondence and Translation for Heterogeneous Data. In: *Proceedings of the International Conference on Database Theory (ICDT)*, 351-363, Delphy, Greece, 1997.
- [ABITEBOUL ET AL., 2000] Serge Abiteboul, Peter Buneman, Dan Suciu: Data on the Web: From Relations to Semistructured Data and XML. *Morgan Kaufman*, 2000.
- [BERNSTEIN ET AL, 2000] Philip A. Bernstein, Alon Y. Halevy, and Rachel A. Pottinger: Model Management: Managing Complex Information Structures. SIGMOD Record 29(4), December 2000. pp 55-63.
- [BAYER, 2001] R. Bayer: XML Databases: Modeling and Multidimensional Indexing. Database and Expert Systems Applications (DEXA). 2001
- [DECKER ET AL, 1998] Stefan Decker, Dan Brickley, Janne Saarela, and Juergen Angele: A query and inference service for RDF, QL'98 - The Query Languages Workshop, WorldWideWeb Consortium (W3C), Boston, USA, 1998, <http://www.w3.org/TandS/QL/QL98/>
- [FRANCONI ET AL, 2000] Enrico Franconi, Fabio Grandi and Federica Mandreoli (2000). A Semantic approach for Schema Evolution and Versioning in Object-Oriented Databases. 6th International Conference on Rules and Objects in Databases (DOOD'00), London, UK, July 2000.
- [GARCIA-MOLINA ET AL., 1995] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and Jennifer Widom: Integrating and Accessing Heterogeneous Information Sources in TSIMMIS. In: *Proceedings of the AAAI Symposium on Information Gathering*, pp. 61-64, Stanford, California, March 1995.
- [GOLDMAN ET AL., 1996] R. Goldman, S. Chawathe, A. Crespo, J. McHugh: A Standard Textual Interchange Format for the Object Exchange Model (OEM). *Technical Report*, Stanford University, 1996, <http://dbpubs.stanford.edu/pub/1996-5>
- [HEFLIN ET AL, 1999] J. Heflin, J., Hendler, J., and Luke, S., Coping with Changing Ontologies in a Distributed Environment, in AAAI Conference Ontology Management Workshop. 1999, AAAI Press. p. 74-79.
- [KARVOUNARAKIS ET AL 2001] G. Karvounarakis, V. Christophides, D. Plexousakis, and S. Alexaki. Querying CommunityWeb Portals. Available at <http://www.ics.forth.gr/proj/isst/RDF/RQL/>, 2001

- [KLEIN & FENSEL, 2001] Michel Klein and Dieter Fensel. Ontology versioning for the Semantic Web. In Proceedings of the International Semantic Web Working Symposium (SWWS), Stanford University, California, USA, July 30 – August 1, 2001.
- [LACHER & DECKER, 2002] Martin S. Lacher, Stefan Decker: RDF, Topic Maps, and the Semantic Web. Markup Languages: Theory and Practice, MIT Press, April 2002
- [LASSILA & SWICK, 1999] O. Lassila, R. Swick (eds.): Resource Description Framework (RDF) Model and Syntax Specification, *W3C Recommendation 22 February 1999*, <http://www.w3.org/TR/REC-rdf-syntax/>
- [LU ET AL, 1996] J. Lu J, P. Barclay, J. Kennedy: On Temporal Versioning in Object Oriented Databases, MoBIS'96 Modelling Business Information Systems, Cottbus, Germany, Oct 1996.
- [MENA ET AL, 1998] E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth: Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure, Proc. Int'l. Conf. on Formal Ontology in Information Systems, Trento, Italy, 1998.
- [PAPAKONSTANTINO ET AL., 1996] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In Proc. *Int. Conf. on Very Large Data Bases (VLDB)*, pages 413-424, Bombay, India, September 1996.
- [RODDICK, 1995] J. F. Roddick. A Survey of Schema Versioning Issues for Database Systems. *Information and Software Technology*, 37(7):383-393, 1995.
- [SCIORE, 1991] Edward Sciore: Using annotations to support multiple kinds of versioning in an object-oriented database system. In: *ACM Transactions on Database Systems (TODS) Volume 16* , Issue 3 (September 1991).
- [SINTEK & DECKER, 2002] Michael Sintek and Stefan Decker: TRIPLE -An RDF Query, Inference, and Transformation Language for the Semantic Web. In: *Proceedings International Semantics Web Conference 2002*, Sardinia, Springer. June 2002.
- [SUCIU, 1998] Dan Suci: An Overview of Semistructured Data. In: *SIGACT News*, vol. 29 no. 4, pp. 28-38 , December, 1998.
- [TICHY, 1985] Walter F. Tichy. RCS - A system for version control. *Software Practice and Experience*, 15(7):637 - 654, July 1985.