Aspect-Oriented Data

Ora Lassila, Mika Mannermaa, Marwan Sabbouh
Nokia Services
Burlington, MA, USA

Ian Oliver
Nokia Research Center
Helsinki, Finland

***Abstract*—Many data-intensive applications have the need to represent and record *cross-cutting aspects* of data that are difficult if not impossible to express within a particular domain model. As a possible approach to implementing these aspects, we discuss the idea of "decoupling" domain data models from some type of "raw" representation or view of data, and associating the aspects (such as *provenance* information) with this new view.**

I. Introduction

Several "real-world" applications would benefit from fine-grained metainformation about their data. The problems encountered often involve management of data that originates in *multiple sources* (and may – independently of sources – have *multiple owners*) and/or data that is subject to one or more *policies* (security, privacy, etc.). This paper describes work-in-progress where our particular use case is a large, highly diverse database of information used to back a set of commercial Web-based services; data in the database includes users' *personal information* (PIM data, "social" data, photographs, etc.), information that constitutes an e-commerce *product catalog*, data about various temporal *events* (e.g., purchases, downloads, sensor readings), and *geographical data* (e.g., "points-of-interest"). This data can be considered to have multiple owners (e.g., users own their own data) and comes originally from multiple sources (e.g., product suppliers, other social network sites); furthermore all data is subject to a number of policies defined by various stakeholders.

In practice, incorporating all or even any of the aforementioned metainformation into one's data schema or ontology is difficult, and easily results in a very complicated data model that is hard to understand and maintain. It can, in fact, be observed that the metainformation listed above are cross-cutting *aspects* of associated data, and by and large are independent of the domain models in question. The problem with such aspects is similar to the problem of cross-cutting aspects in software system design, a situation[1] that has led to the development of *Aspect-Oriented Programming* [2].

In a typical scenario, problems with aspects arise from the fact that the "domain-schema" (i.e., an application's own data model) and the "storage-schema" (i.e., how data is organized in the underlying storage subsystem) are very intricately "married" together; in fact, in a typical application that uses a relational database, these two are one and the same. There is, effectively, no view of "raw" data, i.e., data *independent of* the domain model. In a situation like this, one could imagine aspects that,

[1]The motivating situation is sometimes referred to as the "tyranny of dominant decomposition" [1]. Because data models exhibit similar characteristics, the approach presented in our paper is dubbed *"Aspect-Oriented Data"*.

perhaps, were associated with each relational record or object, but building any finer-grained aspect representation would complicate the domain model beyond what realistically can be managed. It would therefore seem desirable that the domain model and the storage schema be "decoupled" somehow.

Much of the metainformation we aspire to record about data is in various ways associated with *data provenance*. Over the years lots of work has been invested in provenance in various ways – [3] provides a general overview of these efforts in the database field. In this paper, however, we are mostly interested in the "low-level" details of recording information that could be used, among other things, to implement more sophisticated provenance tracking and other operations involving and/or requiring provenance. Our own interest, when it comes to specific applications, lies much within practical issues of security and privacy, or more generally *policy enforcement*, as well as being able to manage data that *originated in many sources* and which has *many different owners* (e.g., to synchronize data between various systems and devices).

II. RDF AND PROVENANCE

Since our paper discusses data management largely from the viewpoint of the Semantic Web [4], it should also be noted that with RDF [5] there have been various efforts to deal with provenance. *Reification*, as part of the abstract syntax, is the mechanism provided by the standard itself: A *reified statement* is a node in an RDF graph that *represents* an actual statement (in the same graph); this node has the properties *subject*, *predicate* and *object* whose values are the constituents of the statement being represented. Fig. 1 shows the graph

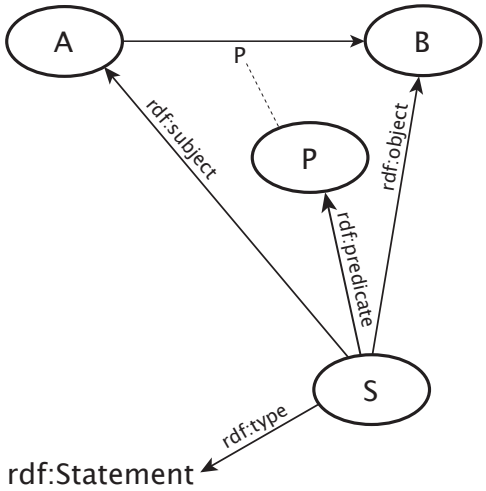

Fig. 1. Reified Statement $\langle A, P, B\rangle$

fragment corresponding to a reified statement $\langle A, P, B\rangle$. Provenance information could thus be attached to the reified statement S as normal RDF properties.

Other approaches – [6], [7], [8], [9], [10], [11], to name a few – attempt to solve the provenance issues by effectively grouping triples and subsequently allowing a group to be referred to as a first-class object and thus made statements about.[2] Syntactically these approaches may be more convenient, but fundamentally there is little difference between them and reification. The real problems arise from the fact that seldom are the "domain-level" data and the provenance data completely separate (in such cases the semantics would be clear) but one will see cases where – for example – a policy is expressed in terms of both provenance data *and* the applications' own data. Standard RDF formal semantics [12] give little guidance in this area.

[2]These groups are called many things, such as *named graphs*, *contexts*, *quoting* or *molecules*.

III. RDF and Aspect-Oriented Data

As a data model (and formalism), RDF is "compatible" with our idea of Aspect-Oriented Data: For one, the triple-based "storage-schema"[3] readily allows to be decoupled from domain-level data models. Reification, on the formalism level, already could be used to implement aspects. In the graph fragment of Fig. 1, aspects could be properties of the node S. Reification, however, is problematic, not only from the semantics standpoint, but also because to reify a triple results in the addition of at least four new triples in the graph – this is clearly evident in the figure.

Our approach takes a practical approach with regard to formal semantics,[4] and also acknowledges that in a triple store, assuming some type of relational storage model, the "triple records" already are *a representation* of the actual statements and thus could be used as the basis of some type of "virtual" reification; this is the approach taken in [13] and [14, section 6.5] to implement paths through reified statements, and similar to the approach taken in [15] to delay concrete reification. Implementation of Aspect-Oriented Data should thus be possible via additional fields/attributes of triple storage, at least assuming we can identify a stable set of aspects. A bigger "show-stopper" may be that available triple stores do not typically support this type of triple metadata (our observation is that this should be easy to implement if you are building your own triple store, but difficult if you use an available product). In comparison to the approaches where triples are grouped, one could view this as the denormalized form of such data.

[3]You could think of this as a *meta-schema*.

[4]Some people would say "cavalier".

IV. Aspects and the "Meta-Schema"

Characteristic to the Semantic Web (and more broadly to *semi-structured data*) is the flexibility of data models and data schemata; it is easy to combine data that use different schemata, and even schema evolution can be effortless (at least if you make "monotonic" additions). What is "fixed" about the data representation is the *meta-schema*, i.e., the underlying view of data as a graph or as a collection of object/attribute/value "triples".

Despite potential aspirations for similar flexibility with regard to the meta-schema (now enhanced with various *aspects*), the approach to implementing the aspects as additional columns/fields of the "triples" may dictate the need to fix the set of aspects to something small that provides decent coverage for many use cases. In our own system we are converging towards the following set:

- **Owner:** Who is the owner of this data in the legal sense, particularly with regard to various policies and policy enforcement.
- **Source:** Where did this data originate (in our early implementations, this is the URL from which a particular triple or set of triples was loaded into the system). This corresponds to the notion of *where-provenance* as defined in [16].
- **Creation time:** When was this particular triple inserted into the system (we assume triples are immutable, so this effectively serves as the modification time as well).

To illustrate how these three aspects could work in conjunction with the reified statement S from Fig. 1, let us say that we want to record that this statement has source C, owner D and creation time T. Fully reified, we would have

the following triples:

$$\langle A, P, B\rangle$$
$$\langle S, \texttt{rdf:subject}, A\rangle$$
$$\langle S, \texttt{rdf:predicate}, P\rangle$$
$$\langle S, \texttt{rdf:object}, B\rangle$$
$$\langle S, \texttt{rdf:type}, \texttt{rdf:Statement}\rangle$$
$$\langle S, \texttt{aod:source}, C\rangle$$
$$\langle S, \texttt{aod:owner}, D\rangle$$
$$\langle S, \texttt{aod:time}, T\rangle$$

If we implemented the aspects as part of the "triple record", we could have this:

$$\langle A, P, B, C, D, T\rangle$$

For the purposes of provenance, it is likely that some other aspects have to be added; likely candidates include:

- **Quality metrics:** Some notion of the reliability or uncertainty associated with a particular triple.
- **Jurisdiction:** Which laws or regulations apply to this piece of data with regard to policy enforcement (it is likely, though, that jurisdiction can more broadly be associated with data owners and/or sources).

Furthermore, some applications may need information about *modification history* of their database, in which case one might be able to use the aspect mechanism to indicate that some triples are "historical" and do not represent the "current" snapshot of data. There are, however, many issues related to proper representation of data as it changes over some span of time. Most representation systems, as already observed in [17], lack any such capability; we consider this to be outside the scope of this paper.

Generally, our emphasis on provenance and the choice of aspects reflects "housekeeping" information, separate from any domain model, and in this regard our approach is quite opposite to the findings in [18].

We also note that there are various issues in reflecting any of the aspects to the concrete level of representation (even though the RDF reification mechanism theoretically would allow one to do so). Discussion of such reflection is considered a potential future goal.

V. Applications of Aspects

Our primary use cases for aspects – *policy-awareness* [19] as well as *data synchronization* – deal with situations where data comes from multiple sources and has multiple owners. We feel that one should strive for a minimal stable set of aspects: having different aspects in each system might prove problematic when integrating data from these systems. The aspect data we now record constitutes a clean separation of concerns as suggested in [1], but it is exactly some policies that will tie aspects together and thus could turn out to be problematic.

VI. Conclusions

We have presented ongoing work, based on RDF, on *decoupling* domain data models from "raw" representation of data, and thus enabling the introduction of *cross-cutting aspects* into data. These aspects, such as *owner*, *source*, and *creation time*, can be used to implement mechanisms needed to track provenance and to enable higher-level functionality predicated on provenance information.

Ideally, and considering future work, we would like to generalize the idea of aspects to something where instead of a *fixed* meta-schema, aspects could be chosen dynamically to support any future use case. Considering this, the

"naïve" implementation approach where aspects are new columns in the "triple table" may not be suitable; naturally, other implementation approaches have to be considered anyway if other type of graph storage is employed (e.g., vertically partitioned databases).

REFERENCES

[1] H. Ossher and P. Tarr, "Multi-Dimensional Separation of Concerns in Hyperspace," in *Proceedings of Aspect-Oriented Programming Workshop at ECOOP'99*, C. Lopes, L. Bergmans, A. Black, and L. Kendall, Eds., 1999, pp. 80–83.

[2] G. Kiczales, "Aspect-Oriented Programming," *ACM Computing Surveys*, vol. 28, no. 4, 1996.

[3] P. Buneman and W.-C. Tan, "Provenance in databases," in *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, 2007, pp. 1171–1173.

[4] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 34–43, May 2001.

[5] O. Lassila and R. R. Swick, "Resource Description Framework (RDF) Model and Syntax Specification," World Wide Web Consortium," W3C Recommendation, Feb. 1999.

[6] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler, "Named Graphs, Provenance and Trust," in *WWW '05: Proceedings of the 14th international conference on World Wide Web*. ACM, 2005, pp. 613–622.

[7] P. Pediaditis, G. Flouris, I. Fundulaki, and V. Christophides, "On Explicit Provenance Management in RDF/S Graphs," in *TAPP'09: First workshop on on Theory and practice of provenance*. USENIX Association, 2009, pp. 1–10.

[8] E. Watkins and D. Nicole, "Named Graphs as a Mechanism for Reasoning about Provenance," in *Frontiers of WWW Research and Development - APWeb 2006*, ser. Lecture Notes in Computer Science, vol. 3841. Springer, 2006, pp. 943–948.

[9] A. Reggiori, D. van Gulik, and Z. Bjelogrlic, "Indexing and Retrieving Semantic Web Resources: the RDFStore Model," in *SWAD-Europe Workshop on Semantic Web Storage and Retrieval*, Amsterdam, The Netherlands, 2003.

[10] L. Ding, T. Finin, Y. Peng, A. Joshi, P. P. da Silva, and D. L. McGuinness, "Tracking RDF Graph Provenance using RDF Molecules," in *ISWC 2005 Poster & Demonstration Proceedings*, R. Mizoguchi, Ed., Nov. 2005.

[11] H. Story, "Temporal Relations," *Sun Babelfish Blog*, March 2006. [Online]. Available: http://blogs.sun.com/bblfish/entry/temporal_relations

[12] P. Hayes, "RDF Semantics," World Wide Web Consortium," W3C Recommendation, Feb. 2004. [Online]. Available: http://www.w3.org/TR/rdf-mt/

[13] O. Lassila, "Generating Rewrite Rules by Browsing RDF Data," in *Proceedings of the Second International Conference on Rules and Rule Markup Languages for the Semantic Web (RuleML 2006)*. IEEE Computer Society, 2006.

[14] ——, "Programming Semantic Web Applications: A Synthesis of Knowledge Representation and Semi-Structured Data," Ph.D. dissertation, Helsinki University of Technology, November 2007.

[15] S. R. Newcomb, "Preemptive Reification," in *The Semantic Web - ISWC 2002, 1st International Semantic Web Conference*, ser. Lecture Notes in Computer Science, I. Horrocks and J. Hendler, Eds. Springer-Verlag, 2002, vol. 2342, pp. 414–418.

[16] P. Buneman, S. Khanna, and W. Tan, "Why and Where: Characterization of Data Provenance," in *International Conference on Database Theory (ICDT 2001)*, 2001, pp. 316–330.

[17] T. L. Dean and D. McDermott, "Temporal Data Base Management," *Artificial Intelligence*, vol. 32, no. 1, pp. 1–55, 1987.

[18] M. D'Hondt and T. D'Hondt, "Is Domain Knowledge an Aspect?" in *Proceedings of the Aspect-Oriented Programming Workshop at ECOOP'99*, C. Lopes, L. Bergmans, A. Black, and L. Kendall, Eds., 1999, pp. 37–45.

[19] M. Mannermaa, "Policy-awareness in Data-intensive Mobile Services," Master's thesis, Helsinki University of Technology, 2010.