Panel: Is Generic Metadata Management Feasible?

Philip A. Bernstein

Microsoft Corporation
One Microsoft Way, Redmond, WA 98052-6399
philbe@microsoft.com

1. Panel Overview

The database field has worked on metadata-related problems for 30 years. Examples include data translation and migration, schema evolution, database design, schema / ontology integration, XML wrapper generation, data scrubbing and transformation for data warehouses, message mapping for e-business, and schema-driven web site design. Tools that address these problems are strikingly similar in their design. Arguably, we are making very little progress, since we keep reapplying the same old 1970's techniques of data translation [9] and views to one new problem after another, without getting much leverage from each succeeding step. Despite all the research on the above tools, we have so far been unable to offer general-purpose database technology that factors out the similar aspects of these tools into generic database infrastructure.

This panel addresses the following questions:

- Is it feasible to develop a generic infrastructure for managing models? If so, what would it need to do, beyond what's offered in the best object-oriented databases and repositories?
- Can we devise a useful generic notion of model that treats all popular information structures as specializations (SQL schemas, ER diagrams, XML DTD's, object-oriented (OO) schemas, web site maps, make scripts, etc.)?
- Can we produce a generic model manipulation algebra that generalizes transformation operations developed for data integration and translation, such as union, match, difference, and merge? What about generic operations on mappings between models, such as invert and compose?
- What is the role of an expression language that captures the semantics of models and mappings, not only for design but also for run-time execution?
- Does a generic approach offer any advantages for model manipulation areas of current interest, such as data integration and XML?

If the skeptics are right that a generic approach to model management is unachievable pie-in-the-sky, are writers of metadata-driven applications doomed forever to writing special-purpose object-at-a-time code for navigating their information structures? If so, what is the leverage that the database field can offer for these problems?

2. Panelists

- Dr. Laura Haas, IBM Research is working on a tool that can (semi-)automatically produce mappings between two data representations. She has been working on various aspects of data integration since starting the Garlic project in 1994.
- Prof. Matthias Jarke, GMD-FIT and Aachen University of Technology, led the ConceptBase project, a model management system that combines semantic modeling and deductive database technology.
- Prof. Erhard Rahm, University of Leipzig, works on the evaluation of metadata management for data warehouses and web portals. He is currently investigating the applicability of generic model management operations to these areas.
- Prof. Gio Wiederhold, Stanford University, has worked on integration of data and knowledge bases for over 20 years [10]. He has proposed a generic

Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt, 2000

ontology algebra that can be used, among other things, for constructing "articulation" models that link and therefore allow integrated access to existing models, without having to integrate them completely.

3. A Case for Model Management

The metadata-related applications listed in Section 1 all involve the manipulation of models and mappings between models. By "model," we mean a complex discrete structure that represents a design artifact, such as an XML DTD, web-site schema, interface definition, relational schema, database transformation script, workflow definition, semantic network, software configuration or complex document. One way to make DBMSs easier to use for metadata-related applications is to make *model* and *mapping* first-class objects with high-level operations that simplify their use. We call this capability *model management* [2].

A possible representation for models and mappings is directed graphs. This amounts to representing them as interconnected sets of objects, which is how most model management applications work today. What's different in the approach proposed here is that an entire graph (i.e., a model or mapping) can be manipulated as a single object, something that no OO database currently does.

To make it easier to build applications, we need to raise the level of abstraction of operations on models and mappings — something much higher level than navigating object structures. Some candidate operators are:

- Match – automatically create a mapping between two models.
- Merge – merge two models into a third, or merge one model into another based on a mapping.
- Compose – return the composition of two mappings (if map_1 relates model M_1 to M_2, and map_2 relates model M_2 to M_3, then return map_3 that relates M_1 to M_3).
- Invert – reverse the direction of a mapping.
- Set operations – union, intersection, difference
- Project and Select – comparable to relational algebra.
- Bulk operations on models, such as Apply (apply a function to all objects in a model) and Delete (delete all objects in a model).

Several research projects in the data integration field have used graph-oriented representations with high-level algebraic operations like those listed above [1][3][4][5] [8]. However, they haven't been generalized to make them applicable to a broad range of model management problems. Other researchers have developed similar operations using sets, rather than graphs, as the representation of models [6].

Graphs and sets can describe the syntax of models and mappings. However, to capture semantics, an expression language is needed, such as some form of logic (predicate calculus, description logic), algebra (relational algebra, arithmetic), or formal language (regular expressions, BNF). To be truly general-purpose, a model management facility would need to factor out the inferencing engine module that can manipulate these expressions, so that one could plug different inferencing engines into the facility. For example, when executing the Compose operation, the model management facility would send the expressions associated with the two mappings being composed to the inferencing engine, which would return the composition of those expressions.

There are many research areas whose technology is potentially worth including in a model management facility, such as deductive databases, answering queries using views, transitive closure and recursive queries, differencing, schema and graph matching, and data translation. This past work gives us some confidence that a generic model management facility is feasible. There is much work to build on.

4. Reasons for Skepticism

The proposed approach to generic model management consists of the following generalizations of problem-specific approaches:

- a single data structure representation for models,
- generic operations on that data structure representation, and
- a generic interface for plugging in inference engines for various expression languages.

These are big steps, given that the state of the art for model management applications is to:

- customize the data structure representation for a particular type of model for a given class of application
- augment a predicate calculus query language (e.g., SQL, OQL) with object-at-a-time code for navigating object structures
- hard code the mapping logic for the problem at hand, rather than use a generic inference engine.

There are many ways to represent the semantics of models and mappings. Moreover, there are many semantic ambiguities in models, which often are quite application-specific. So is there any reason to believe that a generic data model can be defined that meets the needs of these diverse semantic representations and includes operations that correctly interpret subtle ambiguities?

5. Counterpoint

A counter-argument is to identify small, tractable steps. For example, one could do detailed walkthroughs of practical model manipulation problems, showing how generic operations can be used to solve them. This could be done by looking at hard examples solved by some of today's tools for database design, schema mapping, or schema integration. Then one could implement a few generic algebraic operations and apply them to several model manipulation problems, to prove that it's possible to define generic operations that solve practical problems. For example, all of the panelists are currently applying high level operations to internet-oriented data integration problems. In some cases, the operations are partly customized to the application. Whether these operations generalize to other areas is somewhat problematic, but the similarity of operations developed in different research projects gives some reason for hope. In other cases, operations allow for a human in the loop to interpret subtle semantics. But this is hardly a show-stopper, since most metadata management problems involve a human designer anyway, to ensure that models match the real world requirements.

Integration of heterogeneous web-based data sources and e-commerce sites is the latest and most pressing metadata management problem. But it is hardly unique. Many, perhaps most, of the hardest and most pervasive problems facing data management involve the manipulation of models. Yet applications that manipulate models are complicated and hard to build. The best hope for speeding up and reducing the cost of developing such applications is to significantly raise the level of abstraction of model manipulation functions that these applications rely on. This is the goal of generic model management. If successful, it could improve programmer productivity for model management applications by an order of magnitude. It is feasible? There are few database research questions for which there is as much at stake.

6. References

[1] Bergamaschi, S., S. Castano and M. Vincini, "Semantic Integration of Semistructured and Structured Data Sources," *SIGMOD Record* 28, 1, March 1999.

[2] Bernstein, P.A.., A. Y. Levy, and R.A. Pottinger. A Vision for Management of Complex Models. Technical Report MSR-TR-2000-53, http://www.research.microsoft.com/pubs/, June 2000.

[3] Jannink, Jan, Prasenjit Mitra, Erich Neuhold, Srinivasan Pichai, Rudi Studer, and Gio Wiederhold. An Algebra for Semantic Interoperation of Semistructured Data. In *1999 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99),* Nov. 1999.

[4] Milo, Tova and Sagit Zohar. Using Schema Matching to Simplify Heterogeneous Data Translation. In *Proc. 24th VLDB*, pp. 122-133, 1998.

[5] Mitra, Prasenjit, Gio Wiederhold, and Martin L. Kersten. A Graph-Oriented Model for Articulation of Ontology Interdependencies. In *EDBT 2000,* Springer Verlag LNCS 1777, pp. 86-100.

[6] Mylopoulos, John and Renate Motschnig-Pitrik. Partitioning Information Bases with Contexts. In *Proc. 3rd CoopIS*, Vienna, pp. 44-54, May 1995.

[7] Nissen, Hans W. and Matthias Jarke, "Repository Support for Multi-Perspective Requirements Engineering," *Information Systems 24, 2* (1999), Special Issue on Meta-Modelling and Methodology Engineering, pp. 131-158.

[8] Palopoli, Luigi, Domenico Sacca, and Domenico Ursino. Semi-Automatic Semantic Discovery of Properties from Database Schemas. In *IDEAS* 1998, pp. 244-253.

[9] Shu, N.C., B.C. Housel, R.W. Taylor, S.P. Ghosh, and V.Y. Lum. EXPRESS: A Data EXtraction, Processing and REStructuring System. *ACM TODS* 2,2: 134-174, June 1977.

[10] Wiederhold, Gio, and Ramez Elmasri. Data Model Integration Using the Structural Model. In *ACM SIGMOD Conf.*, pp. 191-202. 1979.