Creating a Mediated Schema Based on Initial Correspondences

Rachel A. Pottinger
University of Washington
Seattle, WA, 98195
rap@cs.washington.edu

Philip A. Bernstein
Microsoft Research
Redmond, WA 98052-6399
philbe@microsoft.com

1 Introduction

Video Place, a company that sells videos through its website, has entered a partnership with Movie Reviews, a website that lists movie facts and reviews. Video Place and Movie Reviews wish to set up a data integration system to allow both databases to be accessed at once. Much schema level work is required to do this: a mediated schema must be developed, mappings must be created between the source schemas and the mediated schema, and a mechanism must be provided for translating queries over the mediated schema into queries over source schemas. In addition, if Video Place and Movie Reviews want to change their previous queries to retrieve data from both sources, queries must be translated from the source schemas to the mediated schema.

First, we must create the mediated schema. We can reduce the problem of creating the mediated schema to three tasks: (1) creating a mapping between the source schemas by matching the two schemas to determine where they overlap, (2) merging the two schemas to create a data-model-independent mediated schema, and (3) applying a translation operator to transform the schema into a specific data model (e.g., relational).

Consider the first task of creating a mapping between the two schemas [DMDH02, MHH00, MBR01, RB01]. Suppose each database's actor information is structured as an XML-tree-like representation where each edge is a sub-element relationship, as shown in Figure 1. In this example, the matching task is trivial since elements that correspond to each another have the same name. However, even with such a perfect mapping, differences can remain. For example, the Video Place database (VPDB) represents an actor's name by a single element while the Movie Reviews database (MRDB) represents it by two elements, FirstName and LastName.

Given this difference, should the mediated schema represent Name as one element, two elements (FirstName and LastName), three elements (Name with FirstName and LastName as children), or some other way? The answer affects the behavior of Merge, the amount of guidance about the merge result that should be encoded in the mapping, and the semantics of the mapping needed between the source schemas and mediated schema. The first two issues are largely data-model-independent. Often the third is too, as in our example where the semantics can be expressed by concatenation, an operator present in most view definition languages. Yet most work on merging schemas concentrates on performing the task in a data-model-specific way, e.g., for XML [BM99], or relational and object-oriented databases [LNE89, BC86]. Our goal here is to show how to abstract out the data-model-independent part of the task so it can be reused for any data model. The data-model-dependent part can be performed as a separate subsequent step.

Section 2 describes how to merge two source schemas to create a mediated schema, given an initial mapping between the source schemas. Section 3 describes the kinds of conflicts encountered in creating the mediated

Copyright 2002 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.
Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

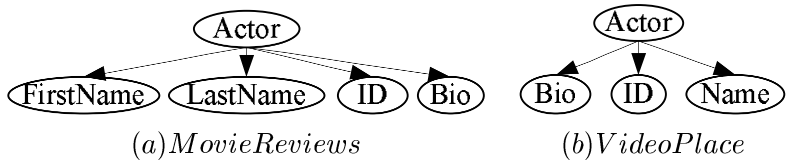

Figure 1: The actor information schema from (a) MRDB and (b) VPDB stored in a generic graph representation

schema. Section 4 describes issues that occur when the semantics of a specific data model is desired. Section 5 describes model management, a data-model-independent approach to a broader collection of schema-related problems similar to the one discussed here. Section 6 concludes.

2 Creating the Mediated Schema

Our goal is to design a generic (i.e., data-model-independent) merge operator that, given two schemas and a mapping between them, returns a merged (i.e., mediated) schema that describes all of the information in the input schemas. To be generic, Merge must use a schema representation that is rich enough to encode features of most commonly used data models. We refer to a schema in this data-model-independent representation as a *model*. Due to space limitations, we'll use an oversimplified representation consisting of elements, properties, and relationships. An *element* is a first-class object that can have single-valued scalar *properties*, including name and (unique) ID, and *relationships* that connect it to other elements. There are two kinds of relationships: Contains that connects an element to its components, and Type-Of that connects an element to its type (which is an element). A model is represented by a set of elements that are connected by Contains and Type-Of relationships. A *meta-model* is a set of elements that represents the types in a given data model, e.g., for SQL schemas or ODMG schemas [CBB+00]. Usually all elements of a model have types taken from one meta-model (e.g., a relational schema). For simplicity, types are omitted from the figures. The *meta-meta-model* is the representation in which models and meta-models are expressed; in our case it consists of the elements, properties, and relationships described above.

The mapping that is input to Merge is a model some of whose elements are *mapping elements*, each of which tells how some elements in the two source schemas are related to each other. A mapping element has:

- *Mapping relationships* (a third kind of relationship) to corresponding elements of the source schemas.
- Figure 2 specifies a potential mapping between MRDB and VPDB where all mapping elements have HowRelated = "equal". If corresponding source elements are tagged as equal, then Merge will collapse them into a single element in the merged schema. By contrast, a different mapping between MRDB and VPDB might indicate that Name corresponds to (but is not equal to) FirstName and LastName. This would be represented by replacing elements 1, 2, and 3 in Figure 2 by a single mapping element with HowRelated = "similar" and with mapping relationships to Name, FirstName and LastName. Merge preserves each similarity mapping element along with the elements it relates to, so that later operations that understand the semantics of the similarity can resolve them.
- An optional property, called Expression, that provides semantics for the correspondence. For example, Expression might say that Name equals FirstName concatenated with LastName. An expression may be meta-model specific. For example, it might be expressed as a conjunctive query.

The mapping need not (and usually does not) refer to all elements of the source models. As well, mappings between elements in the models do not have to be one-to-one.

Reifying a mapping as a model allows us to represent more complex relationships between the input schemas than could be represented by direct correspondences between the input schemas. For example, the mapping in

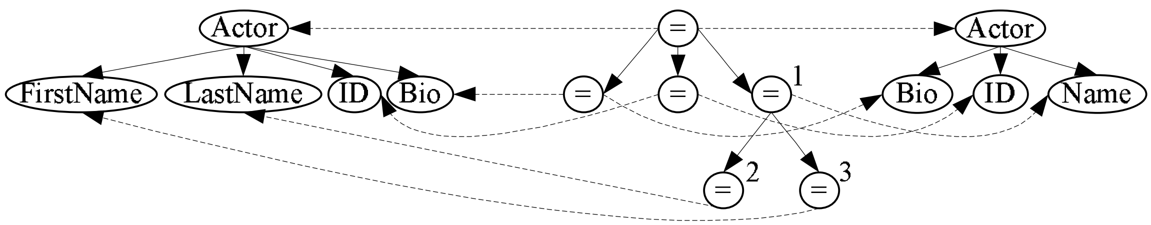

Figure 2: This mapping relates the actor information in MRDB to the actor information in the VPDB. The dashed lines represent mapping relationships

Figure 2 shows that FirstName and LastName in MRDB are effectively sub-elements of Name in VPDB. This is stylistically similar to mappings in [CL93, MBDH02], where a mapping is a logical expression over elements in the two source schemas. We encapsulate such logical expressions in a mapping element's Expression property.

The merge operator produces a mediated schema with the following properties:

- Elements in the two source schemas that are declared to be equal by the mapping are collapsed into one element.
- Elements in the two source schemas that are declared to be similar by the mapping are retained as distinct elements in the mediated schema, along with a description of the similarity relationship between them. We do not specify the detailed semantics of the similarity relationship since that is data-model-dependent.
- Elements in a source schema that are not related to the mapping are retained as distinct elements in the mediated schema.
- For each relationship between two elements in a source schema, if the two elements remain distinct in the mediated schema, then the relationship is copied to the mediated schema
- Each element in the mediated schema includes the disjoint union of all the properties of the input elements from which it was derived.
- No additional elements or relationships are in the result beyond those implied by the previous five bullets unless needed to satisfy constraints (see Section 3).

For example, using the mapping in Figure 2, Merge would return the mediated schema shown in Figure 3.

Merge also returns a mapping between the mediated schema and each source schema. Each mapping relates each element in the mediated schema to the source schema elements that correspond to it. These mappings are not shown in the figure.

3 Conflict Resolution

A schema returned by the previously described merge operator may be semantically flawed. In particular, the schema may be internally inconsistent (i.e., no non-empty database satisfies its constraints) or ambiguous (e.g., incompatibilities between the source schemas have not been reconciled). Most published schema merging algorithms address such flaws by so-called conflict resolution rules.

We classify these flaws into three kinds of schema conflicts: representation (model) conflicts, meta-model conflicts, and fundamental (meta-meta-model) conflicts. Some of them are meta-model-specific and are therefore not appropriate to solve in a generic Merge. Others are generic. We describe here the different classes of conflicts and where they should be resolved.

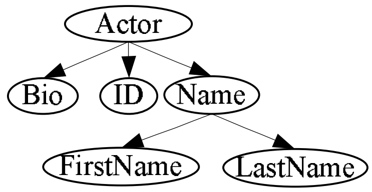

Figure 3: The result of merging the source schemas using the mapping in Figure 2.

Representation Conflicts A representation conflict arises when two source schemas describe the same concept in different ways. Our running example has one such representation conflict: an actor's name is represented by two elements in MRDB (FirstName and LastName) but by one element in VPDB (Name).

We solve a representation conflict by requiring a mapping between the conflicting representations. The mapping describes how the elements in one representation are related to those of the other. This explicitly determines the representation in the mediated schema, including how many elements must be present in the mediated schema and their relationship to one another. If the mapping specifies that the conflicting representations are equal, then Merge can simply collapse the representations into a single element that includes all relationships incident to the elements in the source schemas. If the mapping says that the conflicting representations are similar, then the mediated schema retains the correspondence. In either case, Merge resolves the representation conflict simply by following the course that the mapping has laid out.

For example, the mapping in Figure 2 specifies one possible resolution to the name conflict. The result of that resolution is shown in Figure 3. In this case Name has FirstName and LastName as sub-elements.

Meta-model Conflicts Meta-model conflicts occur when the merge result violates schema constraints specific to a given meta-model (e.g., SQL). For example, suppose that MRDB is a relational database, VPDB is an XML database, and we want the mediated schema to be relational. If we model Actor as a table and use the mapping in Figure 2 to obtain the result in Figure 3, there will be a meta-model conflict because SQL (a particular meta-model) has no concept of sub-column.

By definition, meta-model conflicts must be solved with respect to a specific meta-model, i.e., data model. That is, they are inherently non-generic, so we leave them to be solved after the generic Merge. This adds another requirement to Merge: It must retain enough information in the merge result to enable a post-processing step to repair meta-model conflicts.

Fundamental Conflicts A fundamental conflict arises when the merge result is not a well-formed schema according to the rules of the meta-meta-model. One example is that the meta-meta-model may require an element to have at most one type. So an element with two types manifests a fundamental conflict. For example, VPDB might say Actor ID is of type integer while MRDB says it is of type string. According to the properties of Merge, Actor ID should be both of type date and string, which is a fundamental conflict.

Since Merge must return a well-formed instance of the meta-meta-model, it must resolve fundamental conflicts. Resolution rules for some fundamental conflicts have been proposed, such as [BDK92] for the above one-type restriction. We have identified some other types of fundamental conflicts and resolution rules for them. We incorporate all of these rules into our generic Merge.

4 Semantics

While creating the generic mediated schema can be largely done without considering semantics, semantic issues are unavoidable to make the mediated schema useful. One example is enforcing constraints by resolving

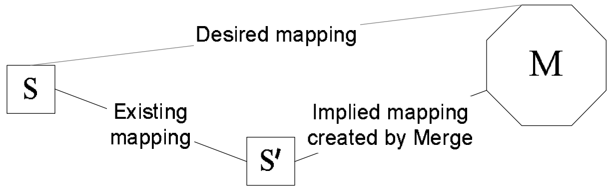

Figure 4: The desired mapping can be created by composing the previously existing mapping between S and S' and the implied mapping between M and S'

representation and meta-model conflicts, as discussed in Section 3.

We must be able to translate user queries posed over the mediated schema into queries over the source schemas. To do so requires semantic mappings that describe how the mediated schema is populated with tuples based on tuples in the source relations. Merge only provides generic mappings from the mediated schema back to the source schemas. While these mappings lack the necessary semantics, they constrain the problem of determining the semantic mappings.

Clio [MHH00] allows users to create semantic mappings based on *value correspondences* which provide information that items are related. Using the additional structure that our mappings offer over their value correspondences, we suspect that some of the mapping semantics can be generated automatically. We plan to explore this, particularly how to exploit the Expression properties of the mapping elements of the mapping that is input to Merge.

5 Model Management and Data Integration

The merge operator described here addresses only one of the schema manipulation problems that arise in data integration. In [BHP00], we proposed several other generic schema manipulation operators as part of a general-purpose model management system. (There we exclusively use the term model instead of schema to emphasize it genericity.) The other main operators are:

- Match - create a mapping between two given models
- Apply - apply a function to all of the elements in a given model
- Compose - given a mapping between models A and B and a mapping between models B and C, compose them to return a mapping between A and C
- Difference - given two models and a mapping between them, return a model consisting of all the items in the first model that are not mapped to the second model.

These operators can help us solve other data integration problems. For example, Apply can be used to encapsulate resolution rules for meta-model conflicts and, therefore, to do the necessary post-processing on the result of Merge. As another example, suppose a mediated schema M was constructed by a series of merges, and we want to add another source S. First, we use Match to create a mapping between S and M, followed by human review to polish the mapping. Then we use Merge to return a schema that subsumes S and M.

Suppose that instead of using Match to create a new mapping between M and S we can use a previously existing mapping between S and a source S' that has already been incorporated into M, as shown in Figure 4. We can compose the S–S' mapping with the S'–M mapping that was returned by Merge, to create a mapping between M and S.

The mapping returned by Merge between a source schema S and a mediated schema M can also be used to identify which elements to remove from M should S leave. The difference operator can tell which elements of M correspond to those in S. Such elements that are not in the range of a mapping between M and some other data source should be removed.

6 Summary

We presented a generic merge operator for creating a mediated schema, discussed various conflicts that can arise, and showed how we solve them in generic model management framework.

Acknowledgements

We thank Alon Halevy and Renée Miller for many helpful comments throughout this ongoing research.

References

[BC86] J. Biskup and B. Convent. A formal view integration method. In *SIGMOD*, pages 398–407, 1986.

[BDK92] Peter Buneman, Susan B. Davidson, and Anthony Kosky. Theoretical aspects of schema merging. In *EDBT*, pages 152–167, 1992.

[BHP00] Philip A. Bernstein, Alon Y. Halevy, and Rachel A. Pottinger. A vision of management of complex models. *SIGMOD Record*, 29(4):55–63, 2000.

[BM99] C. Beeri and T. Milo. Schemas for integration and translation of structured and semi-structured data. In *ICDT*, pages 296–313, 1999.

[CBB+00] R. G. G. Cattell, Douglas K. Barry, Mark Berler, Jeff Eastman, David Jordan, Craig Russell, Olaf Schadow, Torsten Stanienda, and Fernando Velez, editors. *The Object Database Standard: ODMG 3.0.* Morgan Kaufmann Publishers, 2000.

[CL93] Tiziana Catarci and Maurizio Lenzerini. Interschema knowledge in cooperative information systems. In *CoopIS*, pages 55–62, 1993.

[DMDH02] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Learning to map between ontologies on the semantic web. In *WWW*, 2002.

[LNE89] James A. Larson, Shamkant B. Navathe, and Ramez Elmasri. A theory of attribute equivalence in databases with application to schema integration. *Transactions on Software Engineering*, 15(4):449–463, April 1989.

[MBDH02] Jayant Madhavan, Philip A. Bernstein, Pedro Domingos, and Alon Halevy. Representing and reasoning about mappings between domain models. In *AAAI*, pages 80–86, 2002.

[MBR01] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with Cupid. In *VLDB*, pages 49–58, 2001.

[MHH00] Renée J. Miller, Laura M. Haas, and Mauricio A. Hernández. Schema mapping as query discovery. In *VLDB*, pages 77–88, 2000.

[RB01] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal.*, 10(4):334–350, 2001.

[SJJ96] William W. Song, Paul Johannesson, and Janis A. Bubenko Jr. Semantic similarity relations in schema integration. *Data Knowledge and Engineering*, 19(1):65–97, 1996.