RESOLVING SEMANTIC HETEROGENEITY THROUGH THE EXPLICIT REPRESENTATION OF DATA MODEL SEMANTICS

Susan D. Urban and Jian Wu
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287-5406
urban@asuvax.eas.asu.edu

1 Introduction

An important aspect in addressing the semantic heterogeneity problem of multidatabase systems is adequate representation of meta data. Meta data in a heterogeneous database includes schema information about each local database, about the global, integrated view of the system, and about mapping/conversion information between global and local schemas. The explicit representation of such information is required to support the translation of global queries to local queries, to support the formation of updatable global views, and, in general, to support the ability to reason about semantic heterogeneity. As described in [Sche90, Shet91], the problem of view update propagation in a heterogeneous environment remains to be solved. We attribute the lack of results in this area to the inability to reason about semantic differences between different models and local schemas.

This paper describes an approach to the representation of meta data that makes use of current research on object-oriented databases (OODB's) [Zdon90], semantic data models [Hull87], and self-describing data models [Burn86] to explicitly represent semantic heterogeneity. We are particularly interested in environments that require some form of global, integrated schema over traditional database systems. As described in [Shet90], the global schema approach, although impractical for large complex applications, is particularly suited to environments that support naive users or where some degree of control is required over access to information. A semantic model as a front-end to an OODB provides an expressive tool for representing global schemas. Furthermore, current research on object algebras [Shaw90] provides a framework for the expression of set-oriented queries over an OO view of heterogeneous data. Object algebras therefore provide an avenue for investigation of updatable global views.

Forming updatable views, however, requires the ability to reason about how modifications to an OO global schema are subsequently implemented in local databases. To achieve an explicit representation of semantic heterogeneity, the approach described in this paper uses a self-describing semantic model to represent the structural aspects of the OO global view. The semantic model is also model-describing since it is used to describe the structural semantics of other models that participate in the multidatabase environment and the global-to-local mappings. Additional conversion factors are expressed as import/export procedures associated with the OO schema. The global schema is essentially derived data where the mapping and conversion procedures provide derivation rules and algorithms. Because of the explicit representation of the structural semantics of each local database, rules can also be defined to reason about meta data for query translation and view update propagation.

The remainder of this paper is structured as follows. Section 2 presents the self-describing view of a semantic data model and the model description of the relational model. Section 3 then describes the use of mappings and import/export procedures to express semantic heterogeneity between global and local schemas. A summary and discussion of future research is provided in Section 4.

2 Representation of Meta Data

The top portion of Figure 1 illustrates the features of the semantic data model used in this research. The diagram in Figure 1 uses the semantic model in a self-describing mode, which allows the meta schema information to be represented and queried in the same manner as the global schema. As indicated in Figure 1, there are two primary model constructs: CLASSs and PROPERTYs. An ISA connection (thick arrow) indicates that CLASSs can be either SIMPLE-CLASSs with printable values or ABSTRACT-CLASSs representing objects that must be explicitly created and deleted. Details of SIMPLE-CLASSs are not presented here.

ABSTRACT-CLASSs can have either single-valued or multi-valued properties. Graphically, single-valued properties are indicated with single-headed arrows, while multi-valued properties are indicated with double-headed arrows. In the meta schema, single-valued and multi-valued properties are distinguished using the *sv/mv* boolean-valued attribute of the PROPERTY class (i.e., true if the property is single-valued and false otherwise).

A PROPERTY is viewed as a function that maps objects in the *domain* to objects in the *range*. PROPERTYs can be inverses of other PROPERTYs, as indicated by the *inverse-of* attribute of the PROPERTY class. PROPERTYs can also be characterized as *required* (no null values) or *unique* (1:1). Graphically, required property names

are indicated with an "(R)" following the property name. An ABSTRACT-CLASS can also have composite KEYs which are formed from one or more PROPERTY objects.

Finally, DISJOINT-CATEGORYs are a means of specifying disjoint subclasses for a given superclass. The superclass is the *category-owner* while the subclasses are the *category-members*. Disjoint classes are denoted graphically with an arc through the appropriate subclasses of the superclass.

In addition to using the semantic model for representing its own structural semantics, the model is also used to describe the semantics of the data models employed by the local schemas that map to the global schema. The bottom portion of Figure 1 illustrates the use of the semantic model to describe the structure of a relational schema. The description is similar to the self-describing view of the relational model as presented by Mark and Roussopoulos in [Mark86]. The description in Figure 1 is different, however, in that the relational model is described using semantic modeling concepts. The semantic model used in Figure 1 is therefore self-describing *and* model-describing; in the top portion of Figure, 1 the semantic model describes itself while in the bottom portion of the figure the semantic model describes another data model.

Figure 1 indicates that each RELATION is associated with a specific *database* and can have many ATTRIBUTEs. Each ATTRIBUTE has a *name* and a *domain*. ATTRIBUTEs can also be either *required* and/or *unique*. *Dependency* is a boolean attribute indicating whether the ATTRIBUTE object is a functional or multi-valued dependency with respect to the key of the relation. Finally, the KEY class indicates the candidate and foreign keys of each relation. The *key-part* attribute indicates the attributes that compose the key. The *fk-to-pk* attribute links the attribute as a foreign key to its associated primary key. A similar representation of the network model is described in [Urba91].

3 Representation of Semantic Heterogeneity

Semantic heterogeneity is found in many different forms. One form is found in the structural differences between data models such as the network and relational data models. The model-describing approach of this research provides a way to directly address structural differences within a heterogeneous environment, with the structural description of different data models represented directly in the meta data of the global-to-local mappings. For example, if we are integrating relational and network data models, Figure 1 would be extended to include the semantic model description of the network model as in [Urba91], with appropriate mappings from semantic to network model constructs. The structural aspects of relational and network models are therefore made explicit so that the heterogeneous environment can reason about the differences between relational and network schema organizations.

Another form of semantic heterogeneity is found in representational differences. For example, an object represented as a relation on one database may be represented as an attribute in another database. In our approach, the mappings between the semantic model and the underlying data models (i.e., the dashed arrows in Figure 1) help to resolve representational differences. A third form of semantic heterogeneity is found in different units for the representation of data (e.g., pounds, dollars, francs, etc.). Still another form is found in the differences in semantic interpretation of data (e.g., a hotel room rate in one database may not include tax, while a room rate in another database does include tax).

The approach described in this section makes use of a combination of encapsulation together with the explicit representation of data model semantics and mappings to address the problem of semantic heterogeneity. Encapsulation is particularly useful for encoding the computational differences between data units and the semantic interpretation of data values. Such conversions are difficult to express in a declarative form. Structural/representational differences and global-to-local mappings, however, are more useful if represented in an explicit form. The explicit representation of such information supports a rule-based approach to query translation and global query plan formulation/optimization [Weis91]. An explicit representation also supports the ability to reason about the realization of updates on local data.

Referring again to Figure 1, the dashed lines represent mappings between the structural representation of the global schema and the structural representation of a relational schema. For example, an instance of ABSTRACT-CLASS in the global schema could be represented as a RELATION or as an ATTRIBUTE of a relation, while an instance of the PROPERTY class can correspond to one or more ATTRIBUTES of a RELATION. Notice that the mapping between the models is multi-valued in both directions. For example, an ABSTRACT-CLASS can correspond to more than one RELATION in the same database. An ABSTRACT-CLASS can also correspond to a RELATION in one database and an ATTRIBUTE in another database. The global-to-local mappings therefore help to resolve differences in structural representations between local data models. From the opposite point of view, a RELATION may represent data that corresponds to several different ABSTRACT-CLASSes. Furthermore, the ATTRIBUTEs of a RELATION can correspond to PROPERTYs or ABSTRACT-CLASSes in the global view.

Mappings such as those in Figure 1 provide a framework for transforming queries over the OO schema to queries over the local databases. The mappings alone, however, do not provide enough information to resolve the semantic differences between global and local views. To address semantic heterogeneity involving computational differences, additional specifications are required to enhance the data model semantics and global-to-local mappings.

Figure 2 presents specifications for the semantics involved in deriving global data from local dependencies. The first statement in Figure 2 indicates the start of specifications for mappings from the semantic model to the relational database DB1. A similar specification would exist for each local database in the environment. The three occurrences of ABSTRACT-CLASS in Figure 2 indicate mappings from classes in the OO schema to relations and/or attributes in DB1. The STUDENT class, for

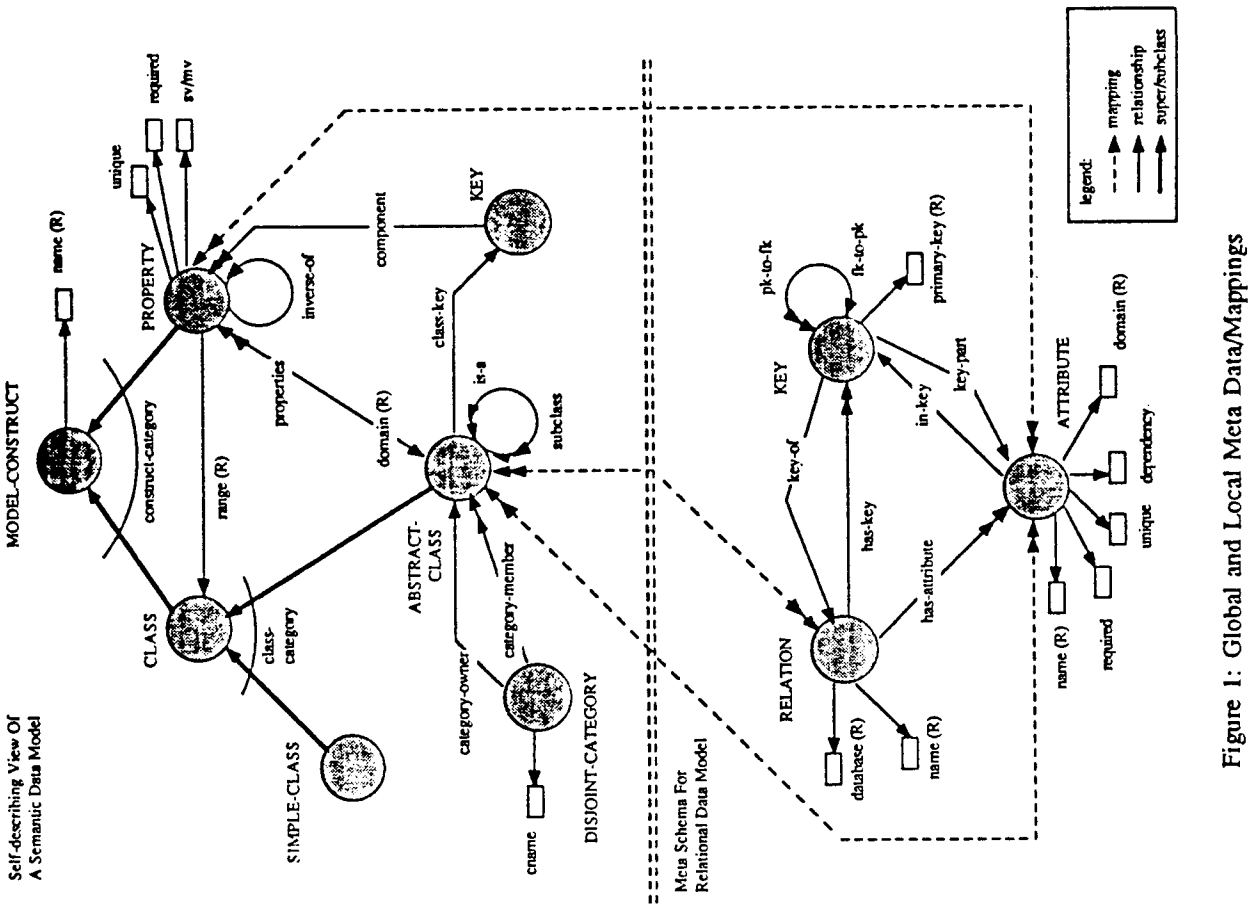

Figure 1: Global and Local Meta Data/Mappings

example, represents a union of the MALE-STUDENT and FEMALE-STUDENT relations. The FACULTY class on the other hand corresponds to a join of the INSTRUCTOR and COURSES-TAUGHT relations. The STAFF class illustrates that a class in the OO view may not be directly represented as a relation but as attributes in another relation. The information in Figure 2 therefore describes additional semantics about the mappings in Figure 1.

The specification of property mappings are shown for several properties of the STUDENT class. The property mappings make use of encapsulation through the specification of import and export procedures. Import procedures are derivation procedures for the OO view of the data, performing conversions necessary to resolve semantic differences between the actual database and the OO view. The parameters of the import procedures define the mappings that are stored in Figure 1 and the information that must be retrieved to create the OO view. The code attached to each procedure defines how the retrieved data must be manipulated to produce the OO view of the data.

In Figure 2, the import procedure for S.Age indicates that Age is calculated from MALE-STUDENT's birthdate in DB1. A similar import procedure (which is not shown in Figure 2) also exists for FEMALE-STUDENT. Thus, import procedures may be overloaded depending on the attributes, relations, and databases from which the property is derived. S.Name, on the other hand, is represented in the OO view as a concatenation of three string values from DB1. Again, the parameters of the import procedure define the mappings in Figure 1 (i.e., the property S.Name in the OO view maps to three attributes in the relational database) while the code of the procedure defines conversions to produce the OO view of the data. Anticipating the need for updatable views, the export procedure for S.Name provides a way to translate changes to the OO view of S.Name into the First, Initial, and Last strings of DB1. As with import procedures, export procedures can be overloaded if a property maps to multiple attributes/relations. Properties without export procedures represent data that cannot be modified in the OO view.

The import procedure for S.Courses, a multi-valued property, indicates that each object in S.Courses corresponds to a Course-no in the COURSES-TAKEN relation of DB1 (i.e, a flat table representation of the courses taken by a student). A mapping for a multi-valued property in the OO schema therefore indicates the relational attributes that represent elements of the multi-valued property. The mapping for S.Gpa, on the other hand, indicates that Gpa is calculated from the set of grades (indicated by braces) of all courses taken by the student. In other words, the value of a single-valued property in the OO view is calculated by retrieving a set of values from the local database. Import and export procedures therefore enhance global-to-local mappings with additional semantics about how the mapping occurs.

```
GLOBAL-TO-LOCAL(DB1)
ABSTRACT-CLASS: STUDENT
  Maps to Relations: MALE-STUDENT UNION FEMALE-STUDENT
ABSTRACT-CLASS: FACULTY
  Maps to Relations: INSTRUCTOR * COURSES-TAUGHT
ABSTRACT-CLASS: STAFF
  Maps to Attributes: Secretary-name, Secretary-phone of DEPT-STAFF

PROPERTY MAPPINGS FOR STUDENT S
  S.Age ← import(DB1, MALE-STUDENT(S).birthdate)
      { code to calculate age }
  S.Name ← import(DB1, MALE-STUDENT(S).First, MALE-STUDENT(S).Initial, MALE-STUDENT(S).Last)
      { code to concatenate First, Initial, and Last as one string }
  S.Name → export(DB1, STUDENT(S).Name, MALE-STUDENT(S).First, MALE-STUDENT(S).Initial,
      MALE-STUDENT(S).Last)
      { code to translate S.Name from one string into three strings }
  Each C in S.Courses ← import(DB1, COURSES-TAKEN(S).Course-no)
      { code to add a course object to the set of courses taken by a student in the OO schema }
  S.Gpa ← import(DB1, {COURSES-TAKEN(S).Grade})
      { code to calculate gpa for a set of grades }
  S.Sex ← import(DB1, MALE-STUDENT(S).Ssn, FEMALE-STUDENT(S).Ssn)
      { code to determine the value of the Sex property in the OO schema based on existence of
          the object in the MALE-STUDENT or FEMALE-STUDENT relation }
```

Figure 2: Mapping/Conversion Specifications

4 Summary and Future Work

The framework described in the previous section is currently being used to develop an environment for investigation of updatable global views in a heterogeneous environment. The self/model-describing approach allows the meta data of the global and local schemas to be represented and queried in the same manner as the integrated, OO view of heterogeneous data. The uniform representation of global and local schemas also supports explicit representation of global-to-local mappings. Mapping information can therefore be enhanced with rules about query translation procedures to different data models. We are currently developing a rule-based approach to query transformation from an object algebra to SQL, with plans to extend the translation to network databases. Future work will also address updates to OO views and the development of rules that use the mapping and import/export procedures to correctly translate global updates to each local database. Further issues of semantic heterogeneity that must be addressed are those of resolving the identity of objects that exist in more than one database.

5 References

[Burn86] Burns, T., et al., "Reference Model for DBMS Standardization," *ACM SIGMOD RECORD*, vol. 15, no. 1, March 1986, pp. 19-58.

[Hull87] Hull, R. and King, R., "Semantic Database Modeling: Survey, Applications, and Research Issues," *ACM Computing Surveys*, vol. 19, no. 3, Sept. 1987, pp. 201-260.

[Mark86] Mark, L. and Roussopoulos, N., "Metadata Management," *IEEE Computer*, vol. 19, no. 12, Dec. 1986, pp. 26-36.

[Sche90] Scheuermann, P. and Clement, Y., "Report in the Workshop in Heterogeneous Database Systems," *Data Engineering Bulletin*, vol. 13, no. 3, Dec. 1990, pp. 3-11.

[Shaw90] Shaw, G. M. and Zdonik, S., "A Query Algebra for Object-Oriented Databases," *Proceedings of the Sixth International Conference on Data Engineering*, Los Angeles, February 1990, pp. 154-162.

[Shet90] Sheth, A. P. and Larson, J. A., "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *Computing Surveys*, vol. 22, no. 3, Sept. 1990, pp. 183-236.

[Shet91] Sheth, A. P., "Issues in Database Design and Integration: A Researcher in Search of Pragmatic Solutions," Panel Session, *VLDB 91*, Barcelona, 1991.

[Urba91] Urban, S. D., "A Semantic Framework For Heterogeneous Database Environments," *First International Workshop on Interoperability in Multidatabase Systems*, Kyoto, Japan, April 1991, pp. 156-165.

[Weis91] Weishar, D. and Kerschberg, L., "An Intelligent Heterogeneous Autonomous Database Architecture for Semantic Heterogeneity Support," *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems*, Kyoto, Japan, April 1991, pp. 152-155.

[Zdon90] Zdonik, S. and Maier, D., *Readings in Object-Oriented Database Systems*, Morgan Kaufmann, 1990.