SIGMOD RECORD

A Quarterly Publication of SIGMOD
the Special Interest Group on
Management of Data

Volume 20 Number 4 December 1991

CONTENTS

Continued on inside front cover

acm PRESS

Semantic Issues in Multidatabase Systems
Preface by the special issue editor

Amit P. Sheth, Bellcore

Semantic issues in the database context

Many database researchers believe that the two most important current areas of database research are DBMS support for next generation database applications and management of heterogeneous, distributed databases (e.g., [L90]). Other terms related to heterogeneous, distributed databases include multidatabases, interoperable databases, federated databases, and interdependent databases.

Several projects and prototype development efforts on heterogeneous, distributed databases started in the late '70s and early '80s, mostly focusing on providing retrieval access to databases managed by heterogeneous DBMSs. The last five years has seen an explosion in research in this area. Four workshops, at least five special issues of research periodicals, and several conference panels have provided forums for discussion and early dissemination of ideas.

One way to study the issues in this area is to look at the systems and solutions in the three dimensional space defined by distribution, heterogeneity, and autonomy [S87]. With earlier research in distributed database as background, the current emphasis is to address various issues of heterogeneity. Research addressing issues along the autonomy dimension seems to be in its infancy. This could be the subject of the next wave of research.

We can classify the issues along the heterogeneity dimension as system (DBMS) heterogeneity issues and semantic issues [SL90]. Current research in transaction management issues in multidatabase systems (extended transaction models, concurrency and recovery concepts and algorithms) belong to the first class. After a few shaky starts, it is possible to claim that significant understanding of the issues of transaction management in multidatabase systems has been achieved. Much less progress can be claimed regarding the semantic issues. *With high interconnectivity and access to many information sources, the primary issue in the future will not be how to efficiently process the data that is known to be relevant, but which data is relevant and where it is located.*

As we move away from system issues to semantic issues, we move from well defined computational paradigms for symbol manipulation to the issues of meaning and use of data as used by different applications and by different human data administrators and end users. We need to deal with multiple (possibly changing) interpretations of data by different users in different contexts, data inconsistencies, and incomplete information. We need to deal with real world entities (or phenomena), their modeling, and partial information about real world entities stored as extensional information in databases. These requirements often introduce subjective issues.We need to tap into the wealth of knowledge in the fields out side database systems, such as AI, knowledge representation, information systems (IFIP society), and natural language processing. I would argue that these issues must be studied from the database viewpoint if appropriate use of data stored in databases is to be supported for advanced applications.

A terminological introduction to semantic issues

Semantic issues arise in all types of multidatabase architectures-- tightly coupled federations that support transparent access through integrated schemas to loosely coupled federations that support access using a multidatabase language without integrated schemas. They arise during different contexts of information management-- when designing or integrating schemas, when processing multidatabase queries and updates and when interpreting the results of a multidatabase query.

The purpose of this special issue is to discuss the basic issues that arise without specific attention to different architectural or information management contexts. So what are the basic semantic issues?

I think they belong to the following two classes:

- Determining if and how two (or more) objects are related. There are two complementary view points.
One view point emphasizes relationships which exist. ***Semantic equivalence*** specifies that two or more data items (records, tuples, objects, ...) or meta-data objects (attribute types, tables, object classes, ...) refer to the same real world entity and class of entities, respectively. In different context of information management and in different representation paradigms, the process of determining semantic equivalence takes different form. For example, in E-R schema integration, we talk about attribute equivalence and equality correspondence between entity types. In the object-oriented world, we talk about object-identity (*oid*) equivalence (i.e., when *oid*s in two or more databases refer to the same real world entity?). The issue of equivalence can be extended to talk about more general ***semantic relationships*** (e.g., inclusion) to denote related, but not exactly the same data and meta-data. We will use the term semantic relationship where the exact nature of relationship (i.e., mapping using an algebraic formula, function, translation table, or a program) is known. Even a more general term is ***semantic compatibility***, ***semantic relevance*** or ***semantic resemblance*** where two objects are known to have some relationship but exact relationship (i.e., mapping) cannot be specified. Multiple representation, multiple contexts, and incomplete information make identification of semantic equivalence/relationship/relevance difficult in multidatabase systems.
Another view point emphasizes the lack or relationship and semantic differences. ***Semantic heterogeneity*** exists when two objects that represent the same real world entity have different information or are represented differently. Semantic heterogeneity may also refer to a disagreement about the meaning, interpretation, or intended use of the same (semantically equivalent) or related (semantically compatible) data/objects [SL90]. ***Semantic discrepancy*** exists when two objects that represent the same real world entity or have the same definition have inconsistent information (i.e., different extensions). ***Semantic incompatibility*** exists when two objects are unrelated by definition.
- Supporting ***semantic reconciliation,*** a process or technique to resolve semantic heterogeneity and identify semantic discrepancy, and ***semantic relativism*** that supports multiple views or interpretations of the same stored data. Brodie [B84] defined semantic relativism as "the ability to view and manipulate data in a way most appropriate for the viewer."

In the above discussion, I have taken the liberty of defining the terms as I understand them and have not made an attempt to determine if these definitions and explanations agree with their previous use. In fact, the reader will find a lack of agreement regarding the definition of these terms among the authors of papers in this special issue. Some may also find it unnecessary to distinguish between all these terms and their nuances, and limit their attention to a subset of important ones, such as semantic relationship and semantic heterogeneity. Perhaps we should have tried to evolve a consensus for their use during the process of preparing this special issue.

Next let me share a potpourri of my observations and beliefs, a detailed discussion of which is not possible here.

- We must distinguish between modeling (the model world or the representation) and reality (the real world or the conceptual world). When reasoning about semantic relationships between meta-data objects (e.g., as in schema integration), we are trying to correlate objects in the model world based on their relevant relationships in the real world. In other words, a semantic relationship is determined with reference to the real world and not the model world. Naming, data type, or such infor-

mation about any object is only a part of information that could help us in understanding the semantics of an object. The incompleteness and possible inconsistencies of the modeling of the real world entities (phenomena) and their relationships must be recognized and understood. Because the system has incomplete, sometimes inconsistent knowledge, semantic relationships cannot always be determined by the system without human input [SG89].

- Each schema or view can be said to represent one domain of discourse (or a relevant subset of the real world). When reasoning about semantic relationships, the interpretation (or definition) of each meta-data object should not only be valid and consistent with their original domain of discourse, but also with the composition of domains of discourses (corresponding to all schemas being integrated).
- The fact that two objects are some how related, even when exact mappings between related objects are not known, can be useful. In this case, the user can be presented with extensions of compatible objects (that may not have an algebraic property such as union compatibility) and let the user determine the use. For example, if we know two object classes in different databases describe quality of food, but use altogether different measures and criteria to define it (i.e., are semantically heterogeneous), when a user wants to know about the quality of food, extensions of both the classes may be presented to the user along with the corresponding definition of the classes.
- Many techniques are possible to help a person in determining semantic equivalence/relationship/compatibility. This involves comparison of objects (and their definitions/meta data). A partial taxonomy of techniques is given below. The taxonomy shows that techniques from many different branches of computer science are being investigated.
  - Graphical facilities and query languages, CASE techniques/tools
  - Formal/logic-based techniques: classification (e.g., based on terminological logic), logic based approach, constraint analysis, model-based specifications, structural integration
  - AI/heuristic techniques: expert systems, case based reasoning, learning (on meta-data and/or data), problem solving
  - Formal language, natural language
  - Use of a large existing knowledge base, use of thesaurus/dictionary/meta-data, naming standards and guidelines, library of managed objects or reusable generic modeling concepts

A brief overview of the special issue

This special issues consists of thirteen papers and three extended abstracts. Most papers were submitted in response to a solicitation to researchers whom I knew were active in the area, with the rest submitted as a result of an announcement at SIGMOD '91. We went through a two step process. During the first step the potential contributors submitted an abstract in response to a statement of scope of this special issue. Those proposals that matched the scope (some after negotiation and considerable re-focusing) were invited to submit papers for reviews. Considering the overwhelming response, the papers were limited to four to six pages each. In the second step, each contribution was reviewed by two to four peers (from among potential contributors) and the editor. The primary purpose was not to select the papers, but to determine the novelty of the contribution (papers presenting the ideas that are in papers submitted for future conference and journal publications were acceptable), to identify material not directly within the scope, and to improve the quality of presentation.

Some of the papers are position papers discussing basic semantic issues in multidatabase systems, some papers discuss the techniques for resolving semantic heterogeneity, and some discuss both issues. While

some of the semantic issues discussed in the papers may also arise in single databases, they occur more frequently or are exacerbated in multidatabase systems. Following is a very brief overview of what the papers discuss.

Kent gives a nice introduction to what factors cause semantic heterogeneity and semantic discrepancy in multidatabase systems. The section titled 'The Breakdown in Mutidatabase Systems', discussing how the differing assumptions from centralized databases to multidatabases lead to semantic problems in the latter, is especially interesting.

Ventrone and Heiler discuss the interesting problem of domain evolution which refers (slightly rephrased) to 'how changes in the (real world) semantics of domain and stored values may cause the domain to become an aggregate of semantically incompatible sub-domains'.

Kalinichenko discusses semantic abstraction mapping that establishes a correspondence between an information resource (database) and an application. Application semantics of a resource (data) is an important aspect of (real world) semantics of data and can help in establishing semantic relationships.

Eliassen and Karlsen study the issue of *oid*, autonomy, and a type of semantic relationship which they call semantic replication. They briefly study the different notions of *oid* supported in different data models and observe that a strong notion of *oid* in a federated database system can only be supported by sacrificing the autonomy of component database systems.

Worboys and Deen introduce two types of semantic heterogeneity- generic and contextual-for multidatabases containing spatially referenced information such as geographic data.

Gangopadhyay and Barsalou agree with earlier propositions that in general semantic relationship can only be ascertained by providing additional assertions that are external to schema information, but draw their justification from the writing of philosophers and logicians. They identify two sources of semantic heterogeneity, different representations and incomplete information. They also propose a framework for reasoning with constructs and structures from a variety of data models to help in resolving semantic heterogeneity resulting from the representational differences. This paper as well as the next paper add to our understanding of distinctions between syntax/structure and semantics.

Geller et al. present their distinction of structure and semantics based on an object-oriented model that supports separate *class* and *type* definitions.

Saltor et. al. discuss essential and recommended features of a data model that would make it suitable for a federated database system. This paper provides us with a recapitulation of considerable, albeit generally inconclusive, earlier discussions of what are the properties of a good data model, but in the context of multidatabase systems. The properties of data models also have relevance to our ability represent semantic heterogeneity and to support semantic reconciliation and semantic relativism.

Spaccapeitra and Parent study semantic heterogeneity and semantic relationship using four types of conflicts. Particularly interesting is the specification of assertions between entities with structural conflicts (i.e., semantically related entities with different structures/representations).

Urban and Wu use a semantic model which is self-describing (i.e., its meta-model is represented in the same model). Canonical descriptions of other models (e.g., relational model) are also described in this semantic model. Then the dependencies from local schemas are given to a global schema described in the semantic model. This provides a framework for dealing with representational differences among schemas in different data models.

Fankhauser et al. present a novel strategy of using fuzzy and incomplete knowledge about terminological relationships between names together with structural knowledge to predict relationships between corresponding real world objects/classes.

Chatterjee and Segev use a probabilistic model to deal with semantic heterogeneity in the context of query evaluation in a multidatabase system. An interesting feature is estimation of accuracy of a join against heterogeneous databases.

Weishar and Kerschbarg also look at the issues of dealing with semantic heterogeneity in the context of query evaluation against heterogeneous databases. They propose developing domain models in a canonical knowledge representation scheme corresponding to each local or export schema, and resolving semantic heterogeneity using problem-solving techniques and a global thesaurus in a blackboard architecture. Readers interested in AI and knowledge-based approaches to deal with semantic issues may also wish to look at [CHS91].

Litwin et al. extend the concept of normalization by defining a "1st order normal form" for a relational database, and they further extend this concept to a multidatabase. They then observe that multidatabase normalization can be impossible due to autonomy of component databases and justify the need for higher order languages.

Siegel and Madnick discuss the importance of context representation, context models, common meta-data vocabularies, and context management for comparing and integrating objects.

Yu et al. present an overview of their work on establishing relationships among names (an aspect of determining semantic relationship) using a set of knowledge bases consisting of common concepts and a technique based on text processing.

[B84] M. Brodie. On the Development of Data Models. In Brodie, Mylopoulos & Schmidt, Eds. On Conceptual Modeling, Springer Verlag, 1984.

[CHS91] C. Collet, M. Huhns, and W-M. Shen. Resource Integration using an Existing Large Knowledge Base. MCC Technical Report # ACT-OODS-127-91, Austin TX, May 1991.

[L90] Database Systems: Achievements and Opportunities (The "Lagunita" Report of the NSF Funded Workshop on the Future of Database Systems Research). A. Silberschatz, M. Stonebraker, and J. Ullman, Eds. SIGMOD Record, December 1990.

[S87] A. Sheth. Heterogeneous Distributed Databases: Issues in Integration. Tutorial Notes. Intl. Conf. on Data Engg., February 1987.

[SG89] A. Sheth and S. Gala. Attribute Relationships: An Impediment to Automating Schema Integration. In workshop on Heterogeneous Database Systems, Chicago, December 1989.

[SL90] A. Sheth and J. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. ACM Computing Surveys, September 1990.

Structure and Semantics in OODB Class Specifications

J. Geller and Y. Perl
CIS Dept. and CMS
NJ Inst. of Technology
Newark, NJ 07052

E.J. Neuhold
IPSI
GMD
Darmstadt, Germany D-6100

Abstract

A class specification contains both structural aspects and semantic aspects. We introduce a mathematically based distinction between structural and semantic aspects. We show how this distinction is used to identify all structural aspects of a class specification to be included in the *object type* of a class. The model obtained is called the Dual Model due to the separation of structure and semantics in the class specification. Advantages of the separation of structure and semantics have been discussed in previous papers and include separate hierarchies for structural and semantic aspects, refined inheritance mechanisms, support of physical database design and *structural integration* which is impossible in other models.

1 Introduction

In this paper we will discuss a distinction between "structure" and "semantics" that has grown out of our work on an object-oriented data model called the Dual Model. Our purpose in writing this paper for the "Special Issue of Sigmod Record" is, however, **not** to present the Dual Model, but to draw attention to the definitional issues of what semantics is. It is essential to have at least a theory about the nature of semantics in order to deal effectively with semantic heterogeneity.

The basic units of object-oriented database systems are objects and classes. A class can be regarded as a container for objects which are similar in their structure and their semantics in the application. The description of a class contains both structural aspects and semantic aspects. In current systems [CM84, F87, SR86] the structural and semantic aspects are always mingled together. Furthermore, in many systems as O_2 [LR88] and Vbase [AM87] the subclass hierarchy is used (1) to factorize common structure and behavior of classes and (2) to express additional semantic connections between classes.

Because of that, two classes modeling semantically related objects could only be dealt with, if the objects in question are structurally related as well. The use of a single hierarchy for two conceptually distinct connections among specifications resulted in inadequate conceptual models. Therefore, it will be advantageous to separate those two aspects of the specification, to achieve improved modeling capabilities.

We are following the basic "Semantic Data Model tradition" [HM81] in that we are modeling the relations that occur commonly in realistic databases, but we have advanced and refined the terminology. We introduce a mathematically based distinction between structure and semantics. Since there is no common agreement about what is considered "semantics" [W75], we see an advantage in a mathematically based distinction which is not user dependent.

In this paper we show how this distinction led us to create the *object type* to include the structural aspects of a class. Each class has an object type but several classes may share the same object type. We are referring to the model obtained by the separation of structure and semantics as the *Dual Model* [NPGT89a, NPGT89b, NGPT90]. This model supports two kinds of hierarchies, structural and semantic, and therefore enables more accurate modeling of applications.

One advantage of this distinction is demonstrated by a new integration technique, *structural integration* [GPCS91, GPN91a, GPN91b]. This technique permits the integration of classes which are similar in their structure but different in their semantics. The Dual Model is the only model that can integrate such classes.

The Dual Model and structural integration can be said to support and exploit a novel form of *semantic relativism*. Semantic relativism was defined as "the ability to view and manipulate data in the way most appropriate for the viewer" [B84]. Two forms of semantic relativisms discussed in literature are: (a) ability to interpret a data model structure differently (e.g., a relation can be viewed as an entity or a relationship) [B84], and (b) defining multiple views (external schemas) over a database schema (conceptual schema or federated schema) to support different users' needs for viewing and using data differently [B84,S86]. The Dual Model supports a novel form of semantic relativism where structural aspects are represented as object types and semantic aspects are represented as classes. By mapping multiple classes on a single object type, multiple semantics (uses and meanings of data) are supported by a single structure. Structural integration allows us to exploit this form of semantic relativism in the context of integrating multiple views or schemas.

2 Classes and the Distinction of Structure and Semantics

In order to structure the set of objects in the domain of our interest we collect objects into classes. An object is said to belong to the class or be an instance of the class. The actual set of objects which belong to a given class is called the *extension* of the class.

This research has grown out of work on an object-oriented data model called VML (Vodak Modeling Language) [FKRST89]. VML shares many features with our current work and with other object-oriented systems, but it does not support the separation of structure and semantics. A VML class description contains four kinds of properties that define the structure and semantics of its objects.

(1) *Attributes* specify printable values of a given data type.

(2) *Relationships* specify pointers to other classes.

(3) *Methods* specify operations that can be applied to instances of a given class.

(4) *Generic Relations* specify special kinds of pointers to other classes.

The representation of a class may contain constraints. The first constraint is that of an *essential property*. The existence of an object is conditioned on the existence of its essential properties. The second constraint is that of a *dependent relationship*. If the existence of an object depends on the existence of another object, we can model this with a dependent relationship.

In the following two definitions the term "aspect" will be used for attributes, relationships, methods, generic relations, and constraints.

Definition 1: An aspect of a specification is considered *structural* if either (1) it is composed of names, types, and logical or arithmetic operations, or (2) it is decidable whether this aspect is consistent with the mathematical representation of the class(es) it connects to.

Note that the names of a property are considered semantic in other models (e.g., E-R model) but are **not** considered semantic in the Dual Model.

Definition 2: An aspect of a specification is considered *semantic* if either (1) it refers to actual instances of objects in the application or (2) it is **not** decidable just based on the mathematical representation of the class(es) it connects to, whether this aspect describes properly the connection between the corresponding real world objects and their features.

Condition (2) of Definition 2 implies that an intuitive understanding of the application is required to decide whether a semantic aspect of a specification describes properly the reality of the application.

We do not believe that it is possible to provide a mathematical definition to of the complex notion of real world semantics. However, it is possible to capture the notion of *structure* with mathematical terms, as was done in Definition 1. Since an aspect of a class specification is either structural or semantic our definitions enable the distinction between structural and semantic aspects. Such a mathematically based distinction is moving the frontier of understanding what real world semantics is one step further.

Intuitively one can summarize our distinction as follows. The aspects which can be captured formally are the structural ones. On the other hand those aspects which are referring to actual objects of the real world or cannot be fully captured with mathematical terms are the semantic aspects.

On a slightly more abstract level one can say that structural aspects deal with features of the representation system. Semantic aspects, on the other hand, deal with the real world and how it is captured by the representation system. The representation system must permit to capture many different possible states of the world. Therefore, semantic aspects have to be highly flexible. It follows that it is not possible to constrain them in a way that would permit a decision whether a semantic description is consistent, based only on the features of the representation language. Rather, one has to know about the sub-world described by the application to decide about the consistency of a semantic aspect.

The decision whether a property is essential or a relationship is dependent cannot be made based on the mathematical representation of the two classes involved but requires an intuitive understanding of the application. Thus, these two constraints are semantic aspects of a class.

So far we have assumed that any two object classes model different objects of the real world. But, if we want to model the same real world objects in two different ways, we must introduce two different object classes. However, we still want to express the fact that these two object classes describe the same real world object. We say that A is a specialized class of B if any (real world) object which can be represented as an instance of the class A can also be represented as an instance of the class B. In other words, the set of (real world) objects corresponding to the extension of A is a subset of the set of (real world) objects corresponding to the extension of B.

We define two kinds of specialization connections between classes called *categoryof* and *roleof*. However, as opposed to previous models, both *categoryof* and *roleof* are treated as semantic generic relations (Definition 2) and are contrasted with the structural *subtypeof* relation which will be discussed in detail in Section 3. *Categoryof* and *roleof* can be compared to Abiteboul *et al.'s* [AH87] generalization and specialization relationships. Our distinction differs from theirs in that it is based on the notion of a *real world context* while theirs is based on the ability of an object to change the subclass it belongs to.

The *categoryof* connection relates the specialized class to the more general class where both are seen in the **same** application context. The class specialized by *categoryof* is used to model the same real world object with additional knowledge. Thus the representation of the *categoryof* specialized class is a refinement of the general class description.

The *roleof* connection relates the specialized class to the more general class, where the two classes are in **different** contexts of the application. The class

specialized by *roleof* is used to model (a subset of) the real world objects of the general class with additional context-dependent information.

The decision whether a specialization connection is either a *categoryof* or a *roleof* generic relation depends on a decision about the context which cannot be made from the representation of the classes but requires an intuitive understanding of the application. Hence, those connections are semantic.

3 Object Types as Structural Representations

Data types describe the common properties of objects. Since all objects of a class have a common structure, they will have a common representation and will be considered as instances of the same data type. This type is called the *object type* of that class. However, the objects of two different classes may be of the same object type, although the two classes model objects with different semantics in the real world.

An *attribute* is composed of a name and a data type. Thus, it is a structural aspect of a class. Therefore it is included in the object type. A *relationship* contains semantic information as it refers to another class which contains semantic information in the context of the application. However, we can represent the *structural aspects of a relationship* in the following way. In an object type a relationship is defined as referring not to an object class, but to the object type of that object class. This is structural information and as such can be included in the object type. The reference to the actual object classes is contained in the specification of a class that uses this object type and reflects the context of the application at hand.

In summary, this means that the structural aspects of relationships are defined in object types and refer to object types, while the corresponding semantic aspects of relationships are described in the object class, substituting each object type in the relationship by the corresponding object class. We call our model the "Dual Model" due to this separation of the class description into two layers. The distinction between types and classes in the Dual Model is different from the distinction made by Beeri [B90] in a number of points. Specifically, he does not associate types with structure and classes with semantics. His structural layer has to be understood in contrast to a behavioral layer.

The determination of the status of *methods* is more involved. A computational method as, for example, a method calculating the average of a set of numbers is structural, while an access path method is composed of a sequence of transitions between object classes which carry semantic information and thus, such a method contains semantic information. Computational methods are therefore already structural and can be included in the object type. For access path methods a structural description is derived in the same way as for relationships. All the classes in an access path method are replaced by their corresponding object types, resulting in a structural description of the method. This structural description can be included in the object type.

Thus, we are ready for the formal definition of an object type. Like a class an object type is determined by a list of properties which correspond to the already introduced class properties, namely

(1) *Attributes*, with values of some data type;

(2) *Relationships*, with references to other object types (not classes!);

(3) *Methods*, to be used on the instances of the object type (not class!);

(4) *Structural generic relations*, with predefined references to other object types (not classes!).

A *subtype* generic relation connects a refined object type to a more general object type, enabling inheritance of the properties of the general type to the refined type. The *subtype* generic relation specifies that the set of properties of the supertype is a subset of the set of properties of the subtype. This generic relation is structural, since the existence of the subtype relation from class A to class B can be decided mathematically, based on the complete representation of the two classes where the subtype relation is not specified in A (i.e., when the properties of the supertype are also listed for the subtype), without an understanding of the application.

Whenever we need a relationship to refer to a set of objects, we can define an object type to represent this set. The connection of the set object type to the object type representing one object of that set is expressed with the structural *setof* and *memberof* generic relations. The generic relation *setof* from class A to class B is structural because it describes the situation where an instance of class A is actually a set of instances of class B. In such a situation it can be decided mathematically that a set relation holds, based on the representation of the two classes where the *setof* relation is not specified for A, but the description of A shows explicitly that its instance is a set of objects with full description of the properties of such an object. The same applies to the *memberof* generic relation.

An interesting effect of the separation of structure and semantics in the Dual Model is that four possibilities exist with respect to similarity between two database schemata.

(1) Two schemata may be semantically as well as structurally similar. In this case standard generalization-based integration tools can be used successfully.

(2) Two schemata may be semantically as well as structurally different. In this case no integration is possible.

(3) Two schemata may be semantically different but structurally similar. In this case generalization-based integration fails, however, structural integration [GPCS91, GPN91a, GPN91b] is possible. This case is of considerable interest for the top-down process of (view) integration. Structural integration is obtained by identifying two or more classes which may share the same object type. By showing a common object

type we save in the specification of the properties of the two classes in spite of their semantic difference.
(4) Finally, two schemata may be semantically similar but structurally different. This case is of considerable interest in bottom-up (database) integration and will be investigated in the Dual Model environment.

Acknowledgment: The authors thank Amit Sheth for inspiring discussions concerning the ideas of this paper, especially their relations to the concept of semantic relativism.

References

[AH87] Abiteboul, S. and Hull, R.. "IFO: A Formal Semantic Database Model", ACM Transactions on Database Systems, Vol 12, No. 4, pp. 525-565, 1987.

[AM87] Andrews, T. and Morris, C., "Combining Language and Database Advances in an Object-Oriented Development Environment", Proceedings of the OOPSLA Conference, 1987.

[B84] Brodie, M. "On the Development of Data Models," in Brodie, Mylopoulos & Schmidt (eds.): On Conceptual Modeling, Springer Verlag, TIS, 1984.

[B90] Beeri, C. "A formal approach to object-oriented databases," Data & Knowledge Engineering, 5(4), 1990, pp.353-382.

[CM84] Copeland G. and Maier, D., "Making Smalltalk a Database System", ACM SIGMOD, June 1984.

[F87] Fishman, D. et al., "IRIS: An Object-Oriented DBMS", ACM Transactions on Office Information Systems, 4(2), April 1987.

[FKRST89] Fischer D., Klas, W., Rostek, L., Schiel, U. and Turau, V., "VML - The VODAK Data Modelling Language", GMD-IPSI, Technical Report, Dec. 1989.

[GPN91a] Geller, J., Perl, Y., and Neuhold, E., "Structural Schema Integration in Heterogeneous Multi-Database Systems using the Dual Model", First International Workshop on Interoperability in Multidatabase Systems, Kyoto, Japan, April 1991, pp. 200-203.

[GPN91b] Geller, J., Perl, Y., and Neuhold, E., "Structural Schema Integration with Full and Partial Correspondence using the Dual Model", Research Report CIS-91-11, CIS Department, NJIT, submitted as Journal Paper.

[GPCS91] Geller, J., Perl, Y., Cannata, P., and Sheth, A., "Structural Integration using the Object-Oriented Dual Model", New Jersey Inst. Of Tech., Tech Report CIS-91-01.

[HM81] Hammer, M. and McLeod, D., "Database description with SDM: A semantic database model", ACM Transaction on Database Systems, Vol. 6, N. 3, 1981, pp. 351-386.

[LR88] Lecluse, C. and Richard, P., "Modeling Inheritance and Genericity in Object-Oriented Databases", LNCS #326, ICOT 1988, p. 223-237.

[NGPT90] Neuhold, E. J., Geller, J., Perl, Y., Turau, V.. "A Theoretical Underlying Dual Model for Knowledge-Based Systems", Proc. of the first Intl. Conf. on Systems Integration, Morristown, NJ, 1990. pp. 96-103.

[NPGT89a] Neuhold, E. J., Perl, Y., Geller, J., Turau, V., "Separating Structural and Semantic Elements in Object-Oriented Knowledge Bases", Proc. of the Advanced Database System Symposium, Kyoto, Japan, pp. 67-74, 1989.

[NPGT89b] Neuhold, E. J., Perl, Y., Geller, J., Turau, V., "The Dual Model for Object Oriented Knowledge Bases", New Jersey Inst. of Tech., Tech Report CIS-89-23.

[S86] Saltor, F., "On the Power to Derive External Schemata from the Database Schema," Proc. of the 2nd IEEE Int. Conf. on Data Engineering, Los Angeles, CA, 1986.

[SR86] Stonebraker M. and Rowe, L., "The Design of POSTGRES", in Proc. of ACM SIGMOD Conference on Management of Data, Washington, D.C., May 1986.

[W75] Woods, W.A., "What's in a Link: Foundations for Semantic Networks", Representation and Understanding: Studies in Cognitive Science, ed. D.G. Bobrow and A.M. Collins, New York: Academic Press, 1975 pp. 35-82.