Semantic Data Models

Article *in* ACM Computing Surveys · September 1988

DOI: 10.1145/62061.62062 · Source: DBLP

2 authors, including:

Joan Peckham
University of Rhode Island
59 PUBLICATIONS **993** CITATIONS

Semantic Data Models

JOAN PECKHAM and FRED MARYANSKI

Department of Computer Science and Engineering, University of Connecticut, Storrs, Connecticut 06268

Semantic data models have emerged from a requirement for more expressive conceptual data models. Current generation data models lack direct support for relationships, data abstraction, inheritance, constraints, unstructured objects, and the dynamic properties of an application. Although the need for data models with richer semantics is widely recognized, no single approach has won general acceptance. This paper describes the generic properties of semantic data models and presents a representative selection of models that have been proposed since the mid-1970s. In addition to explaining the features of the individual models, guidelines are offered for the comparison of models. The paper concludes with a discussion of future directions in the area of conceptual data modeling.

Categories and Subject Descriptors: D.2.1 [**Software Engineering**]: Requirements/Specifications—*methodologies*; D.2.10 [**Software Engineering**]: Design; D.3.2 [**Programming Languages**]: Language Classifications—*design languages*; H.2.1 [**Database Management**]: Logical Design—*data models*; H.2.3 [**Database Management**]: Languages—*data description languages (DDL)*; *data manipulation languages (DML)*; *query languages*; H.2.8 [**Database Management**]: Database Applications; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*semantic networks*; K.6.3 [**Management of Computing and Information Systems**]: Software Management—*software development*

General Terms: Design, Languages

Additional Key Words and Phrases: Conceptual data models, database systems, data models, logical database design, next generation data models

INTRODUCTION

Although the relational model has provided database practitioners with a modeling methodology independent of the details of the physical implementation, many designers believe that the relational model does not offer a sufficiently rich conceptual model for problems that do not map onto tables in a straightforward fashion. The past decade has seen the emergence of numerous data models with the aims of providing increased expressiveness to the modeler and incorporating a richer set of semantics into the database. This collection of data models can be loosely categorized as "semantic" data models since their one unifying characteristic is that they attempt to provide more semantic content than the relational model. The first research papers on semantic data models appeared approximately 7 years after Codd's initial publications describing the relational model. Thus, in perhaps another 5–7 years, one of the modeling methodologies discussed here may attain commercial viability. This survey selects a representative sampling of the new generation of data

CONTENTS

models and analyzes them in terms of their fundamental modeling constructs, the representation methodologies for the objects of the model, and the methods by which the modeler may express the semantics of the application environment. The goal of the presentation is to provide the reader with an appreciation of the problems that these semantic models attempt to resolve and their contributions to the art of database modeling.

The piece of the "real world" that is represented by the database is commonly called an *enterprise*. The structure or schema of the database model is a representation of the elements and interconnections between elements within the enterprise. The constructs used to model elements are usually called *objects* or *entities*. Enterprises are usually not static; thus database models have structures for modeling operations used to manipulate the objects of the database schema. These structures can be atomic operations or more complex transactions and can be considered objects with relationships defined between them.

One problem inherent in modeling any subset of the real world is the difference between the human's perception of the enterprise and the computer's need to organize the structures in a particular way for efficient storage and performance. This gives rise to three database modeling levels that reflect the user's conceptual model, the machine's physical model, and the mapping from one to the other. These levels are described as follows in the ANSI/SPARC proposal [Burns et al. 1986; Jardine 1977] for database architecture standardization:

(1) *External level.* The user's logical views of the enterprise without consideration for performance or storage issues.
(2) *Conceptual level.* The information model, providing the mapping from the logical to the physical, or internal, level, describing the semantics of the entities and relationships, including descriptions of connections and consistency constraints.
(3) *Internal level.* An abstract model of the physical database concerned with the access paths to and the storage of data.

Using this categorization, the relational model [Codd 1970] and models that are direct extensions of it can be seen as conceptual models. The tables of the relational model, although lacking the ability to express all relationships between the objects of the logical model, do provide convenient means for mapping to the physical model. Models such as the entity–relationship model [Chen 1976] can be viewed as conceptual and external since there is a means to specify objects and relationships corresponding to the user's logical view of the enterprise, as well as a means to map these data structures to the physical structures.

This survey considers conceptual semantic models that make use of entities, relationships, and constraints to describe static, dynamic, and temporal qualities of an enterprise. The desired result is a representation of the enterprise that closely parallels the user's perception, without concern for the physical model. Frequently, standard relationships with their semantics, including the associated operations and constraints, are predefined and strategies for access and storage provided. The user need only choose and combine these relationships to model the real-world

enterprise. As we shall see, there are varying levels on which this modeling support is provided in semantic conceptual models.

For the purposes of this paper, a distinction is made between the abstract models used for representing "real-world" enterprises and the systems that have been developed for use in creating these representations. More is said about this distinction at the end of this section.

Early database research concentrated on the physical structure of databases. Little consideration was given to the user's perception of the data. Of utmost concern were the physical and information structures necessary to provide consistent and efficient database storage and retrieval. The hierarchical [Tsichritzis and Lochovsky 1976] and network models [Taylor and Frank 1976] offer the user the means to navigate the database at the record level, thus providing operations to derive more abstract structures. The relational model [Codd 1970] adds a data structure level, eliminating the necessity of performing primitive record level manipulations of the database. The former approach might be considered as operational, whereas the latter might be considered structural. Modeling capabilities with these approaches are still closely related to the record structure of the database.

In the middle seventies, researchers attempted to simplify the design and use of databases by providing modeling structures that were capable of supporting the user's view of the data. Three papers [Chen 1976; Schmid and Swenson 1975; Smith and Smith 1977] addressed two important ideas in data modeling and signaled the emergence of semantic data models. The first idea was that of *data independence*. Perhaps influenced by developments in programming languages, database researchers felt the user should be free from the details of the physical structure of the database. In this way, the user could model the data in a manner similar to the human perception of the application.

The second idea involved capturing additional semantics in the data modeling process. Existing models were capable of defining some data semantics. For example, functional dependencies from the relational theory established some lower level semantics for data models. Attempts were made to extend these semantics to interrelational dependencies or connections. The meaning of the dependencies and the consistency rules that follow from them were described in the early papers.

One example of the above is a paper by Schmid and Swenson [1975] in which the relationships, characteristic and association, and their associated semantics are described. This was followed by a paper by Smith and Smith [1977] in which abstractions already identified by psychologists and AI researchers were employed for database modeling. These abstractions, *generalization* and *aggregation*, are provided by virtually all current semantic data models.

Generalization is the means by which differences among similar objects are ignored to form a higher order type in which the similarities can be emphasized. An example of this is a PUBLICATION entity in a library database[1] that exhibits characteristics common to all publications held by the library. In Figure 1, relationships between the PUBLICATION entity and other more specialized entities are represented using a type hierarchy with more generalized types at the top and more specialized ones at the bottom. For example, PUBLICATION may be considered a generalization of JOURNAL_PAPER, BOOK, and CONFERENCE_PAPER in the database. Similarly, BOOK may be considered a specialization of PUBLICATION.

Aggregation is the means by which relationships between low-level types can be considered a higher level type. The relational data model employs this concept by aggregating attributes to form a relation. Semantic data modeling permits the aggregation of entity types (or relations) to form higher order entities. An example is the aggregation of TITLE and AUTHOR types

[1] This database will serve as a running example throughout the paper. A description appears in Section 1.

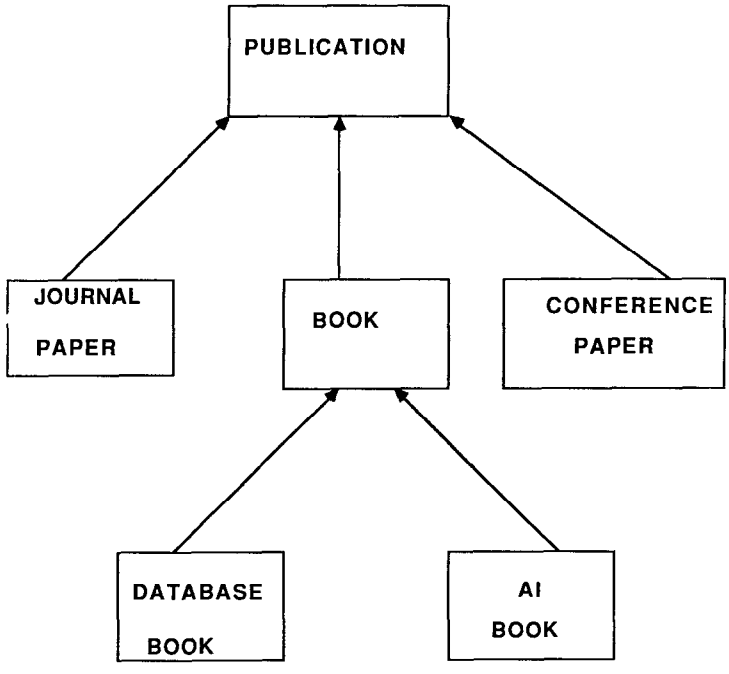

Figure 1. Generalization.

to form a PUBLICATION type in our sample database, as illustrated in Figure 2.

In addition to generalization and aggregation, many semantic models support the *classification* and *association* relationships [Brodie 1984]. Classification is a form of abstraction in which a collection of objects is considered a higher level object class. Essentially, it represents an *is-instance-of* relationship. For example, in our sample database, a BEST_SELLING_BOOK object class consists of all BOOK objects with sales greater than 10,000. The object IT is an instance of the BEST_SELLING_BOOK object class. Classification provides a mechanism for the specification of the type of a specific object, whereas specialization involves the derivation of a type definition from that of an existing type.

Association is a form of abstraction in which a relationship between member objects is considered a higher level set object [Brodie 1984]. The *is-member-of* relationship embodies the association concept. The set DATABASE_BOOKS is an association of BOOK objects as is the set AI_BOOKS. Criteria for set membership is typically based on the satisfaction of some predicate, such as TOPIC = DATABASE for the

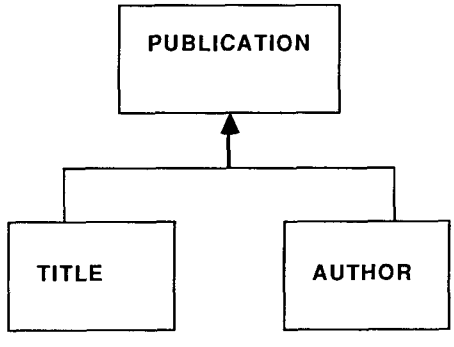

Figure 2. Aggregation.

DATABASE_BOOKS set. The set definition mechanism could, however, be purely external. Consider the GOOD_BOOKS set as an example of a set in which the end user, not the schema designer, determines set membership.

Although association and aggregation define new object types from previously defined types, the represent fundamentally distinct abstractions. Aggregation provides a means for specifying the attributes of a new object type, whereas association is the mechanism for defining a type whose value will be a set of objects of a particular type.

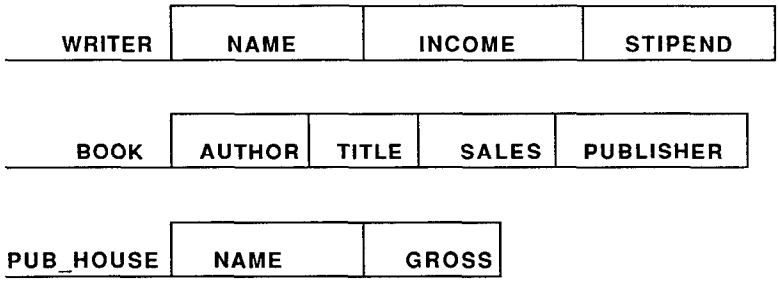

Figure 3. Relational library schema.

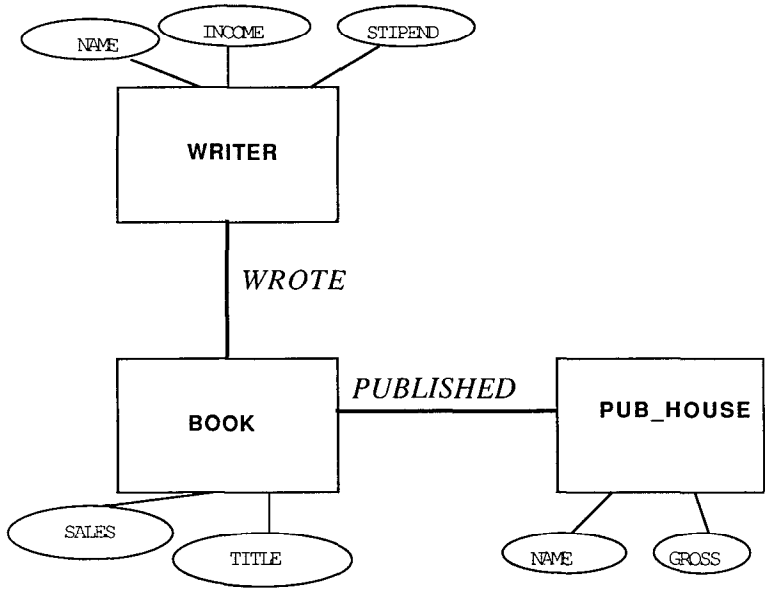

Figure 4. Semantic library schema.

As the class of semantic data models has grown, benefits related to the original research objectives have become clear.

(1) *Economy of Expression.* The semantic data models are usually complete in the sense that the user is capable of extracting the full range of information from the database as easily as in the earlier models. All of these models, however, also provide an economy of expression that can be thought of as stronger than completeness in the following sense: Not only is the user able to extract exactly the same information, but much of this information can be extracted with greater ease.

For example, with the relational model, the user must be aware of the attributes involved in the implicit definition of interrelational connections and perform complex operations on these attributes using projections and joins to extract information through these connections. Hence the semantics of the relationship are embedded within the user program. With a semantic model, operations are explicitly defined upon the relationships. Thus, the semantics exist within the data model itself.

Consider an example dealing with writers and books in the library database. Figure 3 presents the relational schema, and Figure 4 the corresponding semantic representation. The query of interest involves the identification of all poor writers who had successful books published by big publishing houses. The quantification of "poor,"

"successful," and "big" appear in the queries themselves. The relational representation of this query is

```
SELECT NAME FROM WRITER
  WHERE WRITER.INCOME < 20000
  AND WRITER.NAME = BOOK.AUTHOR
  AND BOOK.SALES > 50000
  AND BOOK.PUBLISHER
    = PUBLISHER.NAME
  AND PUB_HOUSE.GROSS > 1000000
```

The same query can be expressed using a semantic model and the notation of Tsichritzis and Lochovsky [1982] as follows:

```
SELECT NAME FROM WRITER
  WHERE WRITER.INCOME < 20000
  AND BOOK.SALES > 50000/WROTE
  AND PUB_HOUSE.GROSS > 1000000/
  PUBLISHED
```

Since the semantics of the relationships are embodied in the join operations that are explicitly required in the relational query, the query has a more compact expression in the semantic model.

(2) *Integrity Maintenance.* The traditional models force the user either to keep track of connections between database objects or to maintain intraobject consistency through navigation of the connections on the physical level. Semantic models provide mechanisms for the definition of integrity constraints and at the same time allow the user to view the data on a level removed from the low-level record structure. Although this functionality is present in all semantic models, the degree varies. (See the discussion on insertion/deletion/modification constraints in Section 1.)

(3) *Modeling Flexibility.* Most traditional[2] data models provided only one means of representing data. Semantic data models, through the use of abstractions, permit the user to model and view the data on many levels. This provides enhanced capabilities for modeling "real-world" situations, since viewing data on many levels is consistent with the way in which people view the world. For example, the abstraction generalization permits one to think of objects in a very detailed or a very superficial way [Hull and King 1987]. The entity–relationship (E–R) model [Chen 1976] provided an early example, permitting the user to perceive of a relationship between two entities as a relationship or as an entity. The latter case is represented in Chen [1976] as a relationship relation.

(4) *Modeling Efficiency.* The designer, while constructing a particular database schema, does not have to implement on a low level. Most semantic models contain built-in elementary operations and constraints. One example is a reference relationship in which one class of objects in the database references another class through an attribute of the referencing object. In our example database, BOOK objects could reference WRITER objects through an attribute, AUTHOR. The reference relationship will have specific operations and constraints associated with it. For example, there may be an insertion operation and an associated constraint that specifies that a referencing object may not be inserted if it references a nonexistent object in the database. This saves the designer from implementing the operation and constraint every time a reference relationship type is defined. This is not a new idea; it parallels work in language theory with abstract data types. In fact, one could say that most semantic models encourage these sound programming techniques.

In the decade that has passed since the first semantic modeling constructs were proposed, a plethora of models has been proposed. This paper surveys and compares a representative sample of these models, focusing upon abstract conceptual models proposed for use in the logical design and specification of semantic databases. The models are analyzed for the presence of constructs representing the fulfillment of the general semantic modeling goals outlined in this section.

Some systems, such as ACM/PCM [Brodie and Silva 1983], provide support for the

[2] We use the term "traditional" to describe the well-established models such as hierarchical, network, and relational.

entire database system life cycle, including specification, creation, and maintenance. These systems are beyond the scope of this survey. The CRIS conference series has focused upon research on the development of such information systems. The reports of these meetings provide in-depth descriptions of a variety of projects, including ACM/PCM, ISAC, NIAM, and D2S2 [Olle et al. 1982, 1983, 1986]. Other such systems are described in Braegger et al. [1985] and Bryce and Hull [1986].

This paper emphasizes semantic data models. For example, SHM+, which serves as ACM/PCM's conceptual model for the description of entities, operations, and constraints, does fall within our spectrum of interest, whereas the remainder of the ACM/PCM work does not. (See Section 2.8 for a discussion of SHM+.)

The remainder of the paper includes Section 1, which defines parameters for comparison of the models, Section 2, in which the models are individually described, Section 3, which provides a tabular comparison of the systems using the parameters given in Section 1 and discusses possible measures for the goodness of such models, and Section 4, which briefly enumerates some future goals for researchers in this and closely related areas.

1. BASIS OF COMPARISON

The recent boom in the development of semantic data models might lead one to believe there is no basis of comparison for this large and seemingly disparate collection. Authors who have attempted to provide some basis for comparison have usually created classes of semantic models. For example, Brodie [1984] groups the models into extensions of classical models, mathematical models, irreducible data models, static semantic hierarchy models, and dynamic semantic hierarchy models. Tsichritzis and Lochovsky [1982] classify models as traditional, entity–relationship, binary, semantic network, and infological data models. Hull and King [1987] take a slightly different approach: A model constructed with fundamental semantic data modeling components is used for comparison with each model. The Hull and King paper has a significant tutorial flavor, addressing the issue of semantic database modeling using a pedagogical data model. That work emphasizes implementation aspects of database systems developed around semantic models, whereas this paper focuses more on conceptual modeling issues.

Kerschberg et al. [1976] classify a collection of conceptual data models along the following parameters: structural, conceptual, and semantic. Set-theoretic versus graph-theoretic parameters are first used; then mathematical foundations are identified to evaluate the structural characteristics of the models. The means by which entities are represented and used to overlay the mathematical structures of the models are used to illuminate the conceptual properties of the models. A linguistic approach is taken to determine semantic levels of abstraction.

In this paper, we identify a collection of concepts that are used to measure the semantic modeling capabilities of each model. This method of presentation focuses upon the support of relationships, the abstractions they represent, the manner in which the semantics are specified, and the approach (if any) to dynamic modeling. The results are used to compare and contrast the models and to identify which approaches might best fulfill the stated objectives of semantic modeling systems.

Every semantic model has *objects* (or entities), *relationships* (functional or relational), *dynamic properties*, and a means for handling *integrity constraints*. Relationships can be characterized by the abstractions they are capable of representing and the means by which they do so. Dynamic properties can range from the simple specification of insertions and deletion constraints to the modeling of operations and transactions. Constraints can be collected from the user and represented and/or automatically implied by the semantics of the model's relationships. Both the level and mechanisms of information representation are used to characterize and compare models.

In this spirit, the following characteristics are identified as being fundamental to semantic data models.

(1) *Representation of Unstructured Objects.* Unstructured data types are defined in Tsichritzis and Lochovsky [1982] as low-level or primitive types that are not constructed through aggregation of lower level types. Strings, integers, and reals are examples of low-level types. Data types of this nature typically are directly supported in the hardware of the underlying computer system. Some models developed for specific applications [Christodoulakis et al. 1986; Su 1983; Woelk et al. 1986] provide primitives more elaborate than those presented by most machines or compilers. Examples are types to support statistical, text, voice, and image data.

(2) *Relationship Representation.* Relationships are analyzed in terms of their presentation to the modeler. Conceptually, the relationship construct may appear in the model as an attribute, entity, independent element, or function. A relationship embodied by attributes is one in which the attribute of one object is connected to, points to, or is derived from another object. For example, if the AUTHOR attribute in a BOOK type is defined to be of type WRITER, where WRITER is an entity type, then the relationship between BOOK and WRITER is represented through the attribute of the BOOK entity.

A relationship is presented as an entity if the relationship of two or more objects conceptually describes a distinct model object. For example, we may choose to represent the REVIEW relationship between REVIEWER and BOOK as an entity in the model, having NAME (of the reviewer) and TITLE (of book), as well as RATING and DATE (of the review), as attributes.

Relationships can also be viewed as independent objects distinct from entities. This does not imply that the physical database represents relationships and entities with different structures, but at least on the conceptual level, entities are viewed as separate from relationships. For example, in Figure 4, the *WROTE* and *PUBLISHED* relationships have distinct presentations from those of the WRITER and PUB_HOUSE entities. In this case the user of the system will view the relationship as a simple connection between two types, and not as a separate entity or attribute as above.

Functional representation is obtained by permitting specification of relationships of objects through functional definitions in the data definition language. For example, the statement

DECLARE AUTHOR (BOOK) = = > > WRITER

permits the designer to define a relationship between BOOK and WRITER types in which the AUTHOR of a given BOOK object is a function of the BOOK object and is a WRITER object. Within the systems surveyed, we shall find examples of each approach.

(3) *Standard Abstractions Present.* As discussed in the Introduction, the abstractions that have most frequently been identified for use in semantic databases are classification, generalization, aggregation, and association.

(4) *Networks or Hierarchies of Relationships.* Virtually all semantic data models offer a diagrammatic construct for the conceptualization of a schema. In most such models, this diagram represents the fundamental modeling abstraction of the model. The most common example is a generalization/specialization graph (or IS-A diagram) representing the derivation of object types. The nature of the graph (network versus hierarchy, cyclic versus acyclic) plays an important role in the characterization of the data model. The graphical expression of a model's fundamental abstractions may take on forms other than the IS-A diagram. For example, the situation given in Figure 2 could be expanded into the aggregation hierarchy of Figure 5, where PUBLICATION is an aggregation of TITLE and AUTHOR, with AUTHOR defined as an aggregation of NAME, INCOME, and STIPEND. Other models, such as the entity–relationship model

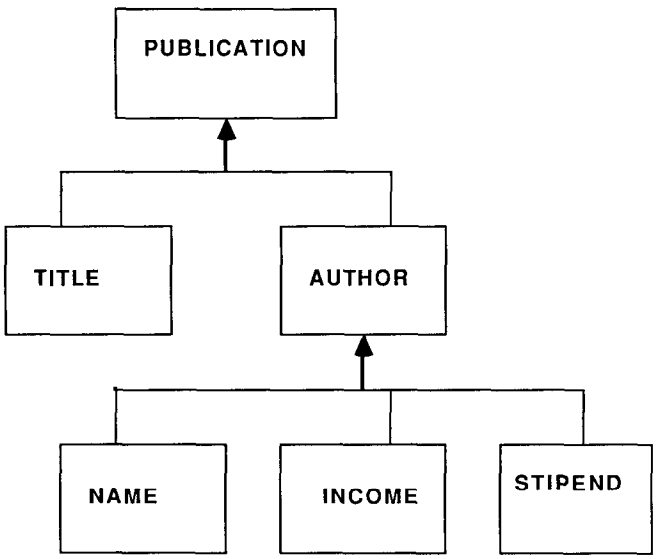

Figure 5. Aggregation hierarchy.

[Chen 1976], support only networks of relationships, as illustrated in Figure 6. With this approach, no provision exists for the expression of a hierarchical structure among relationships (although it may be present in an implicit sense). In Section 3, each model is examined for the extent to which hierarchies (versus networks) are used as organizational structures.

(5) *Derivation/Inheritance.* There are two means by which semantic models handle repeated information within the database schema: semantic connections and derivation. Repetition within individual object types is handled by defining two separate types with semantic connections between them, thus limiting the degree of redundancy (see Section 2.5). Repetition between types is handled with derivation, which is the means by which the attributes of one object are computed or inherited from other objects. Alternatively, class attributes can be used to hold derived information about a class of objects taken as a whole. For example, the class attribute AVERAGE_SALES for the type BEST_SELLER can be computed as the mean of the mean of the number of SALES for all entries in BEST_SELLER.

Inheritance in a generalization hierarchy is the means by which attributes of a more general object are passed on to the more specific object. This can be thought of as a trivial derivation in which the computation is the identity function; that is, if PUBLICATION is defined to be a supertype of BOOK and if the attribute TITLE is assumed to be inherited from PUBLICATION by BOOK, then this can also be thought of as a derivation where

BOOK.TITLE = PUBLICATION.TITLE

Multiple inheritance is the mechanism by which objects in a generalization/specialization hierarchy are permitted to inherit properties from multiple higher level objects. This is convenient for some applications but can be difficult to control. The trouble arises when one specialized object inherits the same property from two higher level objects. Consider the generalization/specialization and aggregation hierarchies of Figure 7. The LITERARY_FIGURE type represents those individuals who are both writers and reviewers. LITERARY_FIGURE is a specialization of both WRITER and REVIEWER and therefore inherits properties from both. REVIEWER and WRITER both have a STIPEND attribute; an *inheritance conflict* can arise since REVIEWER.STIPEND refers to the amount of money the person receives for

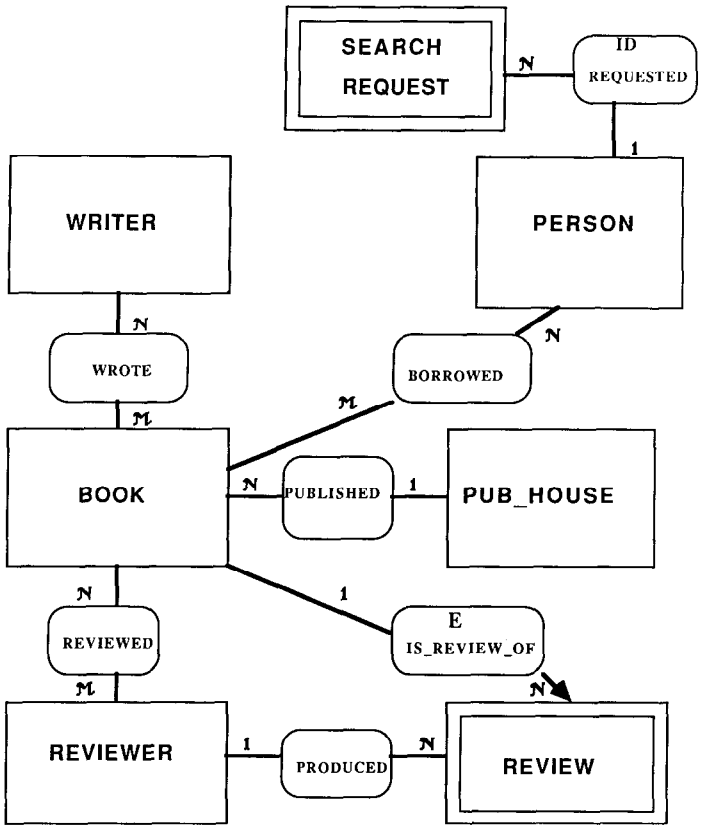

Figure 6. Relationship network.

reviewing a book, and WRITER.STIPEND refers to the sum tendered when a book is written. A semantic model can either prohibit the use of multiple inheritance or offer built-in mechanisms for handling conflicts that may arise.

(6) *Insertion/Deletion/Modification Constraints.* The insertion and deletion constraints used to maintain the integrity of the semantic database form one of its most important features. The specification of these constraints is the physical and operational interpretation of the semantics of the model. If objects are connected through relationships, then the insertion, deletion, or modification of one object will impact the existence status of other objects connected to it.

It is important that the relationships of the model clearly reflect the semantics of the relationship to the database designer and end user; that is, all users should have a clear notion of the consequences of database manipulation. Unforeseen side effects are clearly not desirable. For example, IS-A is a commonly used generalization relationship. If BOOK IS-A PUBLICATION, then when a PUBLICATION object is removed from the database, the corresponding BOOK object will also be removed. These semantics of the IS-A relationship should be clearly conveyed to the designer and user of the model. Alternatively, some models permit the designer to specify the insertion/deletion/modification semantics of relationships.

(7) *Degree of Expression of Relationship Semantics.* Some models leave the expression of the semantics of cardinality, null values, inverse relationships, derivations,

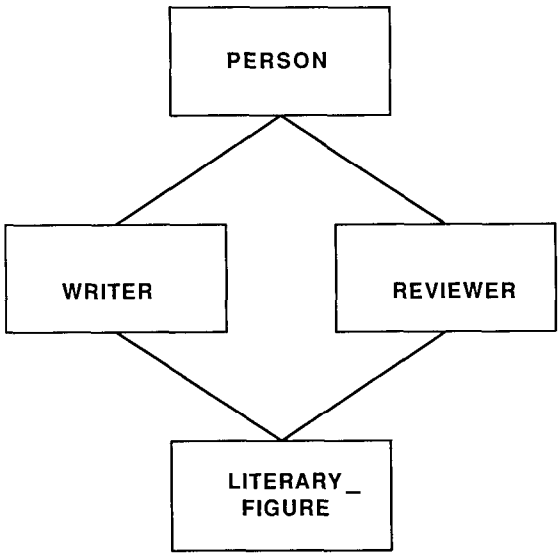

Figure 7. Multiple inheritance.

inheritance (partial/full or over relationships other than those representing generalization), or default values to the designer. Other models completely define the behavior of one or more of these features. The amount of flexibility, and consequently responsibility, given to the designer by the model serves as an important discriminant among models.

(8) *Dynamic Modeling. Dynamic modeling* refers to the description of the semantic properties of database transactions. The counterpoint to dynamic modeling is *static modeling*, which entails the description of the properties of the data objects and relationships. In most projects, research on semantic data models has emphasized the static aspect. However, a minimal degree of dynamic modeling, such as the specification of insertion and deletion constraints, the handling of pre- and postrequisites, error recovery, and transaction definition, appears in most modeling systems.

Some models take a more active role in helping the user define higher level operations and transactions. In certain approaches this is provided only on the level with the objects that they modify. In others, modeling primitives are defined on the basis of object-oriented models in which operations are encapsulated with the objects they modify. See Dittrich [1986] for a general description of object-oriented data models and Dayal and Dittrich [1986] for details on several specific models. Nevertheless, the emphasis in these models lies with the static object as opposed to the dynamic transaction. In addition to the object-oriented viewpoint, some models support a generalization hierarchy approach, which is data driven, or a flow-of-information approach, which is more temporal than data driven.

A few models have taken the description of transactions a step further by applying their primitive modeling abstractions to transactions. These models support the development of complex transactions for more primitive ones by use of specialization or aggregation. Discussions of specific applications of these techniques appear in Section 3. Related information modeling systems that support combining static and dynamic aspects but fall outside of the scope of this paper include NIAM [Verheijen and Van Bekkum 1982] and ISAC [Olle et al. 1982].

1.1 Library Example

A consistent example is used in the presentation of the various models considered in this paper in order to provide the reader

with an informal means of comparing the models. The example serves to highlight certain interesting features of particular models. The role of the example is pedagogical, and therefore there is no intention of presenting a comprehensive description of all aspects of a library database. Elements of the database that would be modeled by a repetition of features already presented are not included since they would not add to the reader's comprehension of the models.

The enterprise selected for the example is a library. In order not to bias the reader toward one modeling methodology and not to make any assumptions as to the reader's prior experience with data models, the information about the library is presented in textual form. Assume that the statements given here were obtained as a result of interviews with the library staff, with some database terminology inserted by the interviewer. For each semantic data model presented in Section 2, we develop a conceptual model of the library based on the following statements:

(1) A PUBLICATION entity has TITLE and AUTHOR attributes.
(2) BOOKs, JOURNAL_PAPERs, and CONFERENCE PAPERs are particular kinds of PUBLICATIONs.
(3) A BOOK entity has SALES, TOPIC, AUTHOR, PRICE, PUBLISHER, INTEREST_INDEX, and ACQUISITION_PRIORITY attributes.
(4) A BEST_SELLING_BOOK has SALES greater than 10,000.
(5) A WRITER entity has INCOME, NAME, and STIPEND attributes.
(6) The AUTHOR attribute of BOOK is of type WRITER.
(7) A PUB_HOUSE entity has NAME and GROSS attributes.
(8) A PERSON entity has a NAME attribute.
(9) A REVIEWER entity has a STIPEND attribute.
(10) A LITERARY_FIGURE is both a WRITER and a REVIEWER.
(11) A BORROWED entity has PUBLICATION and DUE_DATE attributes.
(12) A LIBRARY entity has ACQUISITIONS and ORDERED_PUB_LIST attributes.
(13) The set of DATABASE_BOOKS contains all BOOKs with TOPIC = DATABASE and also has a TOTAL_COST attribute.
(14) The set of AI_BOOKS contains all BOOKs with TOPIC = AI.
(15) The BOOKs in the set of GOOD_BOOKS are identified by the end user.
(16) The set DB_AI_GROUP_BOOKS consists of BOOKs in both DATABASE_BOOK and AI_BOOK.
(17) All BOOKs in the set RESEARCH_GROUP_COLLECTIONS are GOOD_BOOKs.
(18) WROTE is a relationship between BOOK and AUTHOR.
(19) PUBLISHED is a relationship between BOOK and PUB_HOUSE.
(20) REVIEW is a relationship between BOOK and REVIEWER with attributes RATING and DATE.
(21) A REVIEWER is a PERSON.
(22) HOLD is a relationship between BOOK and LIBRARY.

2. REPRESENTATIVE SEMANTIC MODELS

In the following sections, a number of semantic data models selected to represent the various major approaches to conceptual modeling are discussed. Salient features of each model are mentioned, and each is analyzed using the parameters of comparison discussed in Section 1.

2.1 Entity–Relationship Model

The entity–relationship (E–R) model [Chen 1976] is an early semantic data model that unifies features of the traditional (as defined in the Introduction) models to facilitate the incorporation of semantic information. As indicated by the

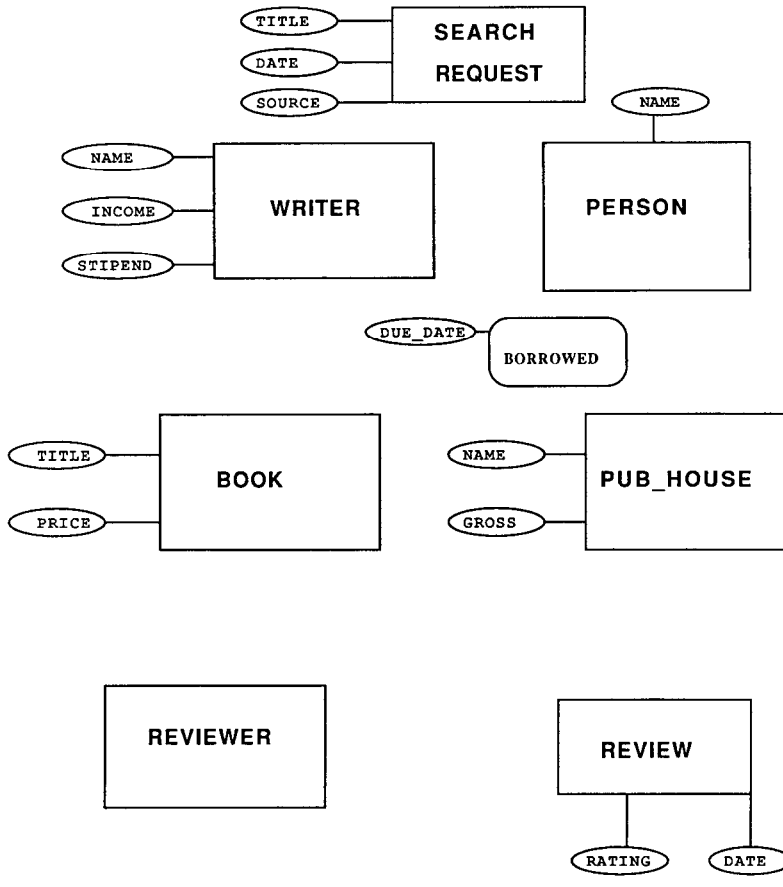

Figure 8. Types and attributes in the E–R model.

name, the two primary modeling constructs are the *entity* and the *relationship*. From a conceptual perspective, the enterprise being modeled is viewed as a collection of entity and relationship types represented graphically (similar to the network model). At the representational level, the information structures for entity and relationship instances strongly resemble relations.

Figure 6 in Section 1 presents an E–R diagram for a portion of the library database. Figure 8 graphically depicts the attributes of the entity and relationship types. From a very abstract viewpoint, as in Figure 8, a relationship exists between a type and its attributes. Although not defined as such in discussions of the entity–relationship model, the relationship between a type and its attributes presents the same abstraction as aggregation to the data modeler.

The E–R model provides strong support for a multiplicity of constraints. The characters 1, $\mathcal{M}$, and $\mathcal{N}$ in Figure 6 indicate the cardinality of the relationship defined by the arc on which they appear. $\mathcal{M}$ and $\mathcal{N}$ denote many. Hence the model explicitly supports one-to-one, one-to-many, and many-to-many relationships. Insertion/deletion constraints are defined using existence dependencies. In Figure 6, the existence of REVIEW entities depends on the existence of a BOOK entity. Thus, if a BOOK entity is deleted, all related REVIEW entities will also disappear from the database. The existence dependency is

represented by enclosing the dependent entity (REVIEW) within a double rectangle, then inserting the label E on the relationship diamond (IS_REVIEW_OF), and including an arrow pointing to the dependent entity (REVIEW). The dependent entity type is known as a *weak entity* in entity–relationship terminology.

The identification of entities can depend upon the nature of the relationships in which the entity participates. Normally, an entity can be uniquely identified by the values of some set of its attributes, that is, NAME of WRITER. The identification of other entities, however, may depend on the relationship with other entities. For example, in Figures 6 and 8, a SEARCH_REQUEST entity, which consists of TITLE, DATE, and SOURCE, does not have a unique identification of its own but must be referenced through the PERSON entity via the REQUESTED relationship. Again, in Figure 6 the presence of an identification dependency has a graphical representation and is denoted by the double rectangle around the SEARCH_REQUEST entity, the arrow on the relationship arc, and the ID in the relationship box.

The only abstraction directly supported in the original E–R model is aggregation, although there are proposed extended models that include generalization [Teory et al. 1986]. A variety of extensions to and applications of the E–R model appears in Chen [1985], March [1988], and Spaccapietra [1987]. The major contribution is that it was one of the first models attempting to provide multiple abstraction levels by combining the best features of the network and relational models. Descriptions of similar models can be found in Pirotte [1977] and Tsichritzis and Lochovsky [1982].

2.2 TAXIS

TAXIS [Borgida et al. 1984; Mylopoulos et al. 1980; Nixon et al. 1987; O'Brien 1983 is a language for the design of interactive database systems that places emphasis on classification and generalization/specialization abstraction hierarchies. The data model combines ideas from programming language and database theory in order to support the following capabilities:

(1) *Data encapsulation.* The operations on database objects are included in their definitions.
(2) *Semantic data modeling.* Specialization is extended from the static to the dynamic portion of the database definition. This includes the modeling of transactions and exception-handling operations. Examples are given below.
(3) *Embedded database types and operations.* The database types, *classes* in the TAXIS terminology, and their operations, *transactions*, can be embedded in existing higher level languages (e.g., Pascal).

The result is a highly structured model, providing integrated modeling constructs for the static and dynamic portions of databases. Generalization/specialization serves as the fundamental organizing abstraction of the TAXIS modeling approach. This conceptual abstraction takes the form of an IS-A hierarchy when presented graphically as in Figure 1. In that figure, BOOK IS-A PUBLICATION and thus inherits all properties of BOOK. In addition, BOOK will take on its own properties. The definition of the TAXIS BOOK class appears below:

```
dataclass BOOK with
  attributes
    AUTHOR: set of WRITER;
    SALES: 0...99999999;
    TITLE: string;
    PUBLISHER: PUB_HOUSE;
    BORROWER: set of PERSON;
    REVIEWERS: set of REVIEWER;
    REVIEWS: set of REVIEW;
end BOOK;
```

TAXIS permits multiple inheritance. As mentioned in Section 1, this can cause problems if not properly handled. TAXIS, however, provides for the resolution of inheritance conflicts through its exception-handling mechanism, which is described shortly.

One special feature of TAXIS is the modeling of the dynamic portion of the database using specialization. The notion of classes

is also applied to the definition of transactions. Consider the OBTAIN transaction for the PUBLICATION class:

```
transaction OBTAIN with
  parameters
    p: PUBLICATION;
    l: LIBRARY;
  prerequisites
    Not_in_library?: (p not_in l.acquisitions);
    Not_yet_ordered?: (p not_in l.ordered_
      pub_list);
  actions
    a1: add p to the l.ordered_pub_list;
  end OBTAIN;
```

In a TAXIS transaction, the prerequisites serve as preconditions that must evaluate to "TRUE" in order for the actions to execute. In this example, the prerequisites verify that the object p does represent a publication already in the library or on order. As a result of this transaction, the publication is placed on the ordered list for processing by the purchasing department. Since the purchasing of a book involves specific operations, such as checking the price and submitting a book order to the publisher, a transaction that processes the purchase of a new book can be created by specializing the OBTAIN transaction as illustrated below. The specialized version of OBTAIN is executed if the object passed as parameter p is of type BOOK.

```
specialize OBTAIN (p: BOOK)
  add
  prerequisite
    Cost_reasonable?: (p.price < = 50);
  action
    a4: order_book (p, p.publisher);
  end OBTAIN;
```

The technique of specialization is extended to encompass the modeling of exceptions as well. A general exception handler is defined for each transaction specifying general actions upon any exception. Specialized exceptions can be associated with the general exception handler defining actions for particular exception types. For example, a general exception handler may be defined for the OBTAIN transaction, with specialized exception handlers to perform specific actions for each exception type. Figure 9 presents a hierarchy of exceptions for the OBTAIN transaction. The modification of the transaction to include exception handling and a sample exception handler is as follows:

```
transaction OBTAIN with
  parameters
    p: PUBLICATION;
    l: LIBRARY;
  prerequisites
    Not_in_library?: (p not_in l.acquisitions);
    Not_yet_ordered?: (p not_in l.ordered_
      pub_list);
  actions
    a1: add p to the l.ordered_pub_list;
      for exception e in OBTAIN_EXCEP-
        TION with pub <-p, lib <-l
      use EX_HANDLER (e);
  end OBTAIN;

Script class EX_HANDLER (e: book_out_of_
    print)
  transitions
    send message;
  actions
    inform user that book is out of print
  end;
```

The BOOK_OUT_OF_PRINT exception would arise if the OBTAIN transaction were called with an unsuccessful candidate as a parameter. Exception handlers are realized in TAXIS as *scripts*, which are generalized processes with explicit communication and synchronization mechanisms. The communication aspects of scripts are of interest here since exceptions frequently result in the display of a message to the user.

In conclusion, TAXIS is a system in which classification and generalization hierarchies are emphasized and extended to the dynamic portion of the database. A distinguishing characteristic of the system is the use of database abstractions to model exception handlers and transactions.

2.3 SDM

Many of the models considered in this survey offer the modeler a small set, typically one or two, of fundamental abstractions. Recall the E–R model with entities and relationships and TAXIS, which provides classes and generalization/specialization hierarchies. SDM [Hammer and McLeod

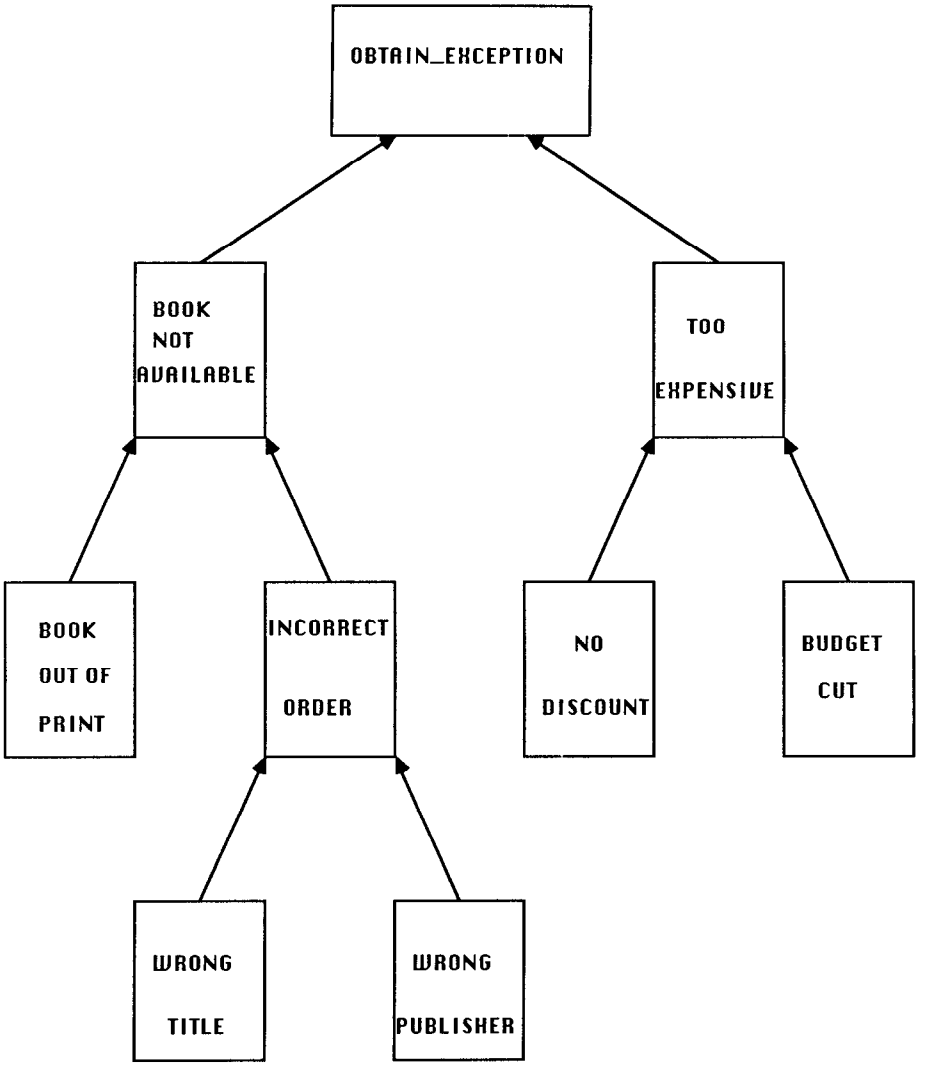

Figure 9. Exception hierarchy.

1981] takes a different approach by incorporating a wide range of modeling constructs into a single abstraction, the *class*. The intent is to permit the database designer to express the "meaning" of the database clearly with mechanisms designed to map directly onto the designer's concepts. Where most other semantic models provide primitives from which the designer can construct more complex conceptual objects, SDM attempts to offer a full set of modeling facilities. In keeping with semantic modeling philosophy, one objective is to permit flexible and multiple views of the data and at the same time carefully control the repetitive modeling effort that arises from such requirements. Aggregation, generalization, classification, association, and derivation are all used to attain these objectives.

Classification and association have greater emphasis in SDM than aggregation and generalization. An SDM database is a collection of entities (instances) organized into classes, or types. The designer defines classes and within this framework specifies member and class attributes, interclass connections, and derivations. This is

different from models with entity and relationship definitions in that the focus is upon the definition of the class itself and not its links to other classes via an E–R diagram or an IS-A hierarchy.

Another example from the library database illustrates the SDM approach. Assume the definitions of BOOK and DATABASE_BOOK given below. The description of BOOK contains only member attributes that reflect the properties of individual books. DATABASE_BOOK, however, contains an interclass connection description and class attributes, in addition to its member attributes. The interclass connection description indicates that DATABASE_BOOK is a subclass of BOOK, where *database* is an element of BOOK.Topic. The class attributes describe properties of the class as a whole.

```
BOOK
  description: all books within the library mem-
      ber attributes:
    Title
      value class: STRINGS
    Author
      value class: PERSON
    Publisher
      value class: PUB_HOUSE
    Price
      description: price of book at library rate
      value class: DOLLARS
    Topic
      value class: TOPICS
    Interest_Index
      value class: INTEGERS
    Acquisition_priority
      value class: INTEGERS
      derivation: order by decreasing Interest_
        Index within BOOK.
  identifiers
    Title

DATABASE_BOOK
  description: all database books within the
        library interclass connection: sub-
        class of BOOK where
    BOOK.TOPIC = 'database'
  member attributes:
    Subtopic
      value class: DB_SUB_TOPICS
  class attributes:
  Total_cost
      value class: DOLLARS
      derivation: sum of Price over mem-
        bers of this class
```

As illustrated by *Total_cost* and *Acquisition_priority* above, the values assumed by attributes can be computed from any other information in the database schema using a sophisticated collection of derivation primitives such as statistical, Boolean, and ordering computations and recursive schema tracing capabilities.

Two types of interclass connections, subtype and group, can be specified in a class definition. The subtype and grouping connections are used to handle generalization and association abstractions, respectively. As in most systems having a generalization hierarchy, inheritance of attributes is included. Since multiple inheritance is permitted, rules are specified to handle inheritance conflicts that may arise when two or more supertypes of a type define an attribute with the identical name but some differences in the values class or derivation. Attributes may be specialized as one moves down the generalization hierarchy. In the above definition, DATABASE_BOOK is a subtype of BOOK. As an alternative, subtypes may be defined in which membership is explicitly stated by the user as in

```
GOOD_BOOKS is a subclass of BOOK
    to be specified by the user.
```

Membership in other classes can also be used to specify a class as follows:

```
DB_AI_GROUP_BOOKS is defined as a
  subclass of BOOK
  where class membership is defined as
  books that are in the classes DATA-
  BASE_BOOKS and AI_BOOKS.
```

The grouping connection is used to specify classes that consist of groups of objects, again either user or constraint specified. As an example, the definition of RESEARCH_GROUP_COLLECTIONS appears below:

```
RESEARCH_GROUP_COLLECTIONS
  interclass connection: grouping of BOOK as
    specified
  member attributes:
    Good_book_list
      description: the good books that are in
          the collection (if any)
      value class: BOOK
      derivation: subvalue of Contents where is
          in GOOD_BOOKS
      multivalued
```

The *interclass connection* statement specifies that the elements of the second research group's collection are user specified. In this case the collection contains the books selected by the members of the research group. *Good_book_list* is a multivalued, derived, member attribute that contains the set of those selected books that are also classified as good books. *Contents* is a multivalued member attribute that is automatically established for each grouping class. The value of this attribute is the collection of members of the class underlying the grouping that forms the contents of that member. In this example each member of the grouping RESEARCH_GROUP_COLLECTIONS, that is, the collection of books in each research group, has as the value of its Contents attribute the set of books selected by the users in that research group. The *derivation statement* for the good_book_list attribute identifies those members of the Contents attribute set that are also in the GOOD_BOOK class.

SDM employs the class abstraction as its primary conceptual modeling vehicle. Whereas classes effectively represent entities when using the SDM approach, relationships are embodied in the interclass connections that are specified as part of the class definitions. Within the class abstraction, SDM offers a rich set of inheritance, constraint, and derivation options. A distinguishing feature of the SDM approach is the focus on the specification of the class without the development of hierarchies or networks presenting interrelationships among classes, as found in several other models addressed in this survey.

Although it is generally true that more primitive modeling constructs do provide greater versatility, one might argue that the difference is in emphasis. SDM can be viewed as providing an extremely rich collection of constructs capable of representing the user's view of the data. SDM has served as a prototype system for other semantic models: Many later systems have indeed chosen a subset of the SDM offerings to provide useful sets of modeling tools for specific classes of enterprises and modeling philosophies.

2.4 Functional Data Model

The functional data model [Shipman 1981] was constructed in conjunction with the data definition language DAPLEX. The objective was to provide a model and definition/manipulation language that is capable of representing applications with naturalness and simplicity. Most models use a variety of constructs to provide modeling flexibility. (These constructs are summarized in Table 1 in Section 3.) The designers of this system found, however, that limiting the constructs to entity and function provide a direct and simple language for data definition and manipulation.

In the functional data model, functions can be used to define the aggregation of attributes used to form an entity. BOOK might be defined as

```
DECLARE BOOK( )==>> Entity
DECLARE Title (BOOK) ==> String
DECLARE Publisher (BOOK) ==>
    PUB_HOUSE
DECLARE Author (BOOK) ==>>
    PERSON
```

Since Author (BOOK) and BOOK( ) are multivalued, ==>> is used to indicate a multivalued function. Relationships are also represented functionally. The declaration of Author (BOOK) describes a relationship between BOOK and PERSON. Multiple argument functions such as

```
DECLARE REVIEW RATING (BOOK,
    REVIEWER)==> RATING
```

can be used to represent aggregate relationships between multiple entities. As a basis for comparison, the E–R model represents the above function as a relationship with attributes. REVIEW_RATING would be a relationship between BOOK and REVIEWER with a RATING attribute.

The model does not provide explicit means for generalization and classification, although the user may define functions representing these abstractions. The null function on an entity type returns theset of elements of that type and

therefore provides a form of association. For example, the set of all books in the library, is represented as BOOK( ).

Although the functional data model does not provide as great a variety of modeling techniques as other models, it is clear that the concise and clear representation of relationships between entities is an advantage. Buneman and Nikhil [1984] discuss the functional data language FQL, in which a small set of functionals is used to provide a collection of query operations for the manipulation of functional databases.

2.5 RM/T

RM/T (Tasmanian model) [Codd 1979] is an extension of Codd's relational model [Codd 1970], attempting to capture more meaning in a conceptual model through the introduction of relationships and integrity rules. The relational model, a brief illustration of which appears in Figure 3 in the Introduction, provides a tabular conceptual model in which all relationships between the tables (relations) are dynamically formed on the basis of data values in the tables.

After the introduction of the relational model, researchers devoted their energies to describing forms of the model that guaranteed high levels of consistency and protection from update anomalies [Codd 1970, 1972; Fagin 1977, 1979]. But the result of maintaining higher order normal forms is usually a collection of relations far more fragmented than originally defined by the user and thus less closely related to the user's conceptual model.

For example, if the user is inclined to define a BOOK relation containing tuples completely describing the book's author, the rules defining normal forms will force the separation of this relation into two relations, BOOK and WRITER, to provide a reasonable level of database consistency. The definition of two separate relations will permit a change in a field such as the author's address to occur in one place without having to update the tuples for all books written by the author. The user, however, still views these two relations as having a relationship or connection between them. The linkage between the relations does not explicitly appear in the relational scheme. For example, Figure 3 when considered by itself without any supporting documentation does not present an obvious link between BOOK and WRITER. The semantic relationship between BOOK.AUTHOR and WRITER.NAME exists only in the mind of the user or in the code of an application program. The relational join operator is used to associate tuples with matching values in those fields. In the relational model, however, a join could also occur between BOOK.TITLE and WRITER.NAME since both are text fields. The result would be technically valid, but not the realization of the relationship under consideration.

Although the relational model presents a well-defined and reasonably straightforward conceptual model, its semantic leaness places a substantial modeling burden on the end user or application programmer. RM/T represents a means of enhancing the semantic expressiveness of the relational model while maintaining its fundamental character. In this spirit, the RM/T model defines entity and relationship types and the corresponding existence constraints between them. Entity types are defined by E-relations, of which there exists one per type, and P-relations, which define the properties (attributes) of the type. An E-relation consists of a single column holding the systemwide, unique identifiers for each instance of the entity type. P-relations are directly associated with E-relations and hold the value for each property. Figure 10 presents the E-relation and a subset of the P-relations for the BOOK entity.

RM/T represents relationships using *associative entity types* for many-to-many relationships and *designative entity types* for many-to-one relationships. In our example, the relationship BORROWED between BOOK and PERSON, which is many-to-many, appears as an associative entity type defined in RM/T as appears below. The BORROWED relation contains references to the appropriate BOOK and PERSON tuples plus an attribute

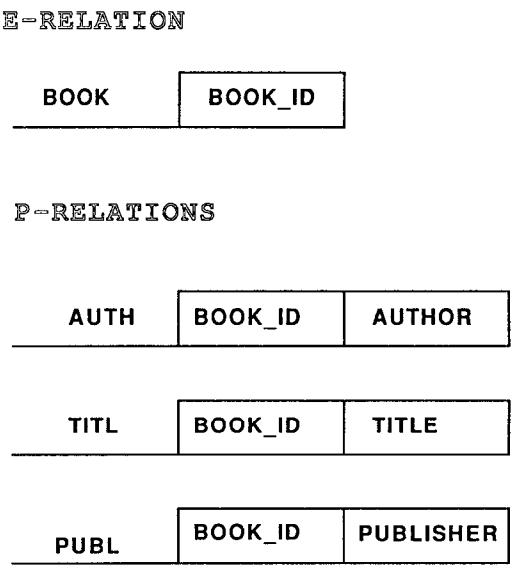

Figure 10. E-relation and P-relations.

indicating the number of copies of the book in the library.

```
CREATE E_RELATION HOLD
  ASSOCIATING (BOOK VIA BOOK_ID,
     LIBRARY VIA LIBRARY_ID);
CREATE P_RELATION PHOLD FOR
    E_RELATION HOLD
  PROPERTIES (SUR_BOOK
    SURROGATE FOR BOOK,
      SUR_LIB SURROGATE FOR
        LIBRARY);
CREATE P_RELATION PHOLD_COP
    FOR E_RELATION HOLD
  PROPERTIES (COPIES DOMAIN
    (COPIES));
```

The relationship between BOOK and WRITER can be represented by defining designative reference to entity type BOOK from entity type WRITER by adding the "DESIGNATING" phrase to the definition of the E_RELATION BOOK as shown below. BOOK is then considered a designative entity type.

```
CREATE E_RELATION BOOK
  DESIGNATING (AUTHOR VIA
    WRITER_ID);
```

RM/T provides numerous built-in integrity rules for the various entity types. The rule given below [Date 1983] applies to the BORROWED associative entity type. Both Codd [1979] and Date [1983] contain complete lists of the RM/T integrity rules.

> A given instance of BORROWED can exist in the database only if, for that instance, each E-attribute of BOOK and PERSON either has the null value, or identifies an existing entity of the appropriate type.

The E-attribute of an entity type is the internal, systemwide, unique tuple identifying attribute. An E-attribute roughly corresponds to object identifier in an object-oriented system. The general form of the above integrity rule holds for all associative entity types in the schema. The rule can be refined so as not to permit nulls. That option would apply to the BORROWED type defined here.

RM/T provides explicit support for type hierarchies through its SUBTYPE clause. In an RM/T type hierarchy, subtypes are distinguished by the values of specified attributes per Figure 11. The relationship DATABASE_BOOK IS-A BOOK is expressed in the definition of the E_RELATION for DATABASE_BOOK as shown in the following:

```
CREATE E_RELATION DATABASE_
  BOOK SUBTYPE OF BOOK PER
    CATEGORY TOPIC;
```

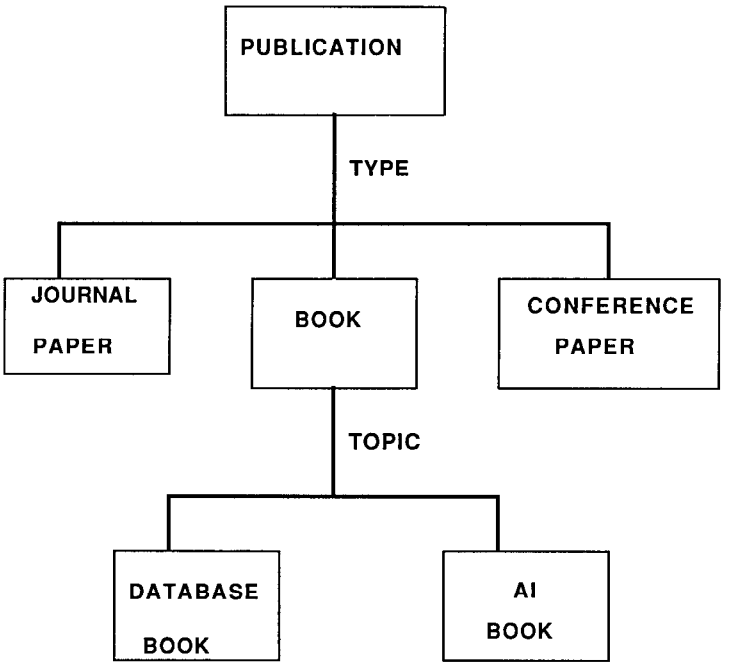

Figure 11. RM/T type hierarchy.

The phrase "PER CATEGORY TOPIC" indicates that the value of the TOPIC attribute determines membership in the DATABASE_BOOK subtype.

Multiple inheritance may occur in an RM/T type hierarchy. In a situation of that nature, RM/T makes the tacit assumption that no naming conflicts will arise since none of the explicit integrity rules address that case.

The orientation of this model is slightly different from that of other semantic models. RM/T arose out of the desire to handle database inconsistencies resulting from the insertion and deletion of tuples connected through interrelational dependencies. Thus the model is information structure oriented, as is the relational model. Most other semantic data models provide similar modeling abstractions, but at the higher, conceptual level distinct from the underlying information structures. RM/T presents itself as an enhanced relational model as opposed to a new approach to conceptual modeling. RM/T, however, still qualifies as a semantic model since the definition of these semantics not only gives more meaning to these relationships, but provides the data structures necessary to utilize the usual data modeling abstractions.

2.6 SAM*: A Semantic Association Model

SAM* [Su 1983] is a semantic model designed originally for scientific–statistical databases and later extended to explicitly support computer-intergrated manufacturing applications. Since SAM* was designed for a particular set of applications, it is characterized by its support for nontraditional object types and relationships lending themselves easily to the design of these databases. The general organization of a model constructed with SAM* is a network of atomic and nonatomic concepts (or objects). The nonatomic constructs are designed through recursive nesting to provide a well-structured and semantically consistent approach to object type definition. On the lowest level, concepts are represented with a set of abstract data types. The expansion to this set of types from the usual strings and numerals was done for the

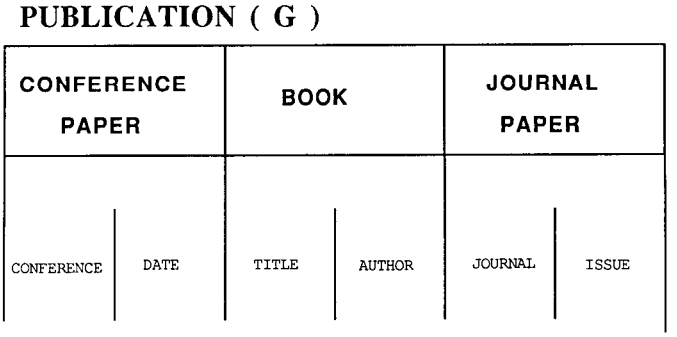

Figure 12. G-relation.

following reasons:

(1) To provide object types that correspond exactly to those seen by the user as the most primitive units of the application.
(2) To provide well-defined operations on the units of information that correspond to the actions occurring in the problem domain.

Both of the above provide more efficient use of the designer's energies since time does not have to be spent building the high-level constructs from more primitive types.

As a result of an analysis of the requirements of the computer-integrated manufacturing environment [Su 1986], the following types and their operations are built into SAM*:

(1) sets (ordered and unordered),
(2) vectors and matrices,
(3) time and time series,
(4) text,
(5) G-relations (generalized relations).

The G-relation of SAM* is an extended relation whose attributes may be of any valid SAM* type, including relation. Subtyping of G-relations occurs by organizing types hierarchically into a semantic network. The mapping involves implicitly embedding G-relation types within the definition of other G-relation types. For example, the G-relation for PUBLICATION pictured in Figure 12 captures the fact that CONFERENCE PAPER, JOURNAL PAPER, and BOOK relations are specializations of PUBLICATION.

SAM* provides the designer with seven built-in relationships (associations in SAM* terminology) that can be organized into networks to model the semantics of a particular enterprise. These seven relationships appear below [Su 1986]. Several have appeared in prior models and therefore receive only cursory mention here.

(1) Membership: "is member of."
(2) Aggregation: Defined in the Introduction.
(3) Interaction: Used to model arbitrary relationships, in the E–R sense. Provides cardinality and referential integrity constraints.
(4) Generalization: Enhanced version of generalization with mutual exclusion, set equality, set–subset, and set intersection constraints on the sibling types of a parent in the generalization hierarchy. Figure 13 illustrates the mutual exclusion constraint, which indicates that a PUBLICATION must be one of either a BOOK, JOURNAL_PAPER or a CONFERENCE_PAPER, and the set intersection constraint, which indicates that a BOOK could be considered both an AI_BOOK and a DATABASE_BOOK. In the diagram, a $\mathscr{G}$ node represents generalization, and an $\mathscr{A}$ node connotes aggregation.
(5) Composition: "is part of."
(6) Cross product: Grouping among types whose instances are the result of taking the cross product of the instances of the component types in order to support statistical analysis by the summarization association defined next.
(7) Summarization: Supports statistical aggregation and disaggregation. By

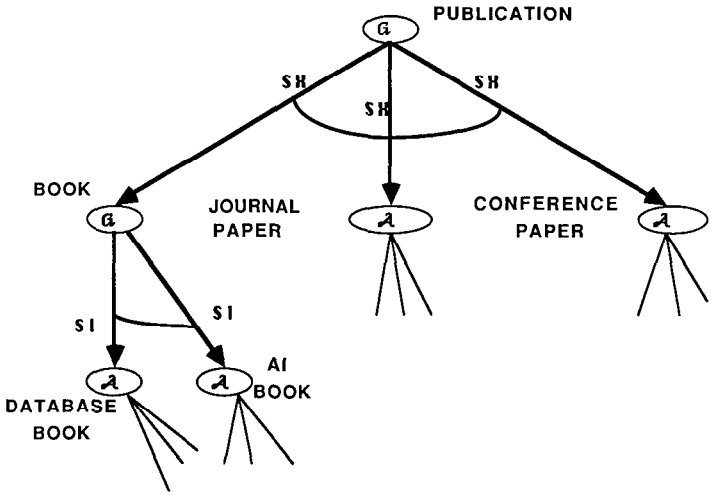

Figure 13. SAM* generalization hierarchy.

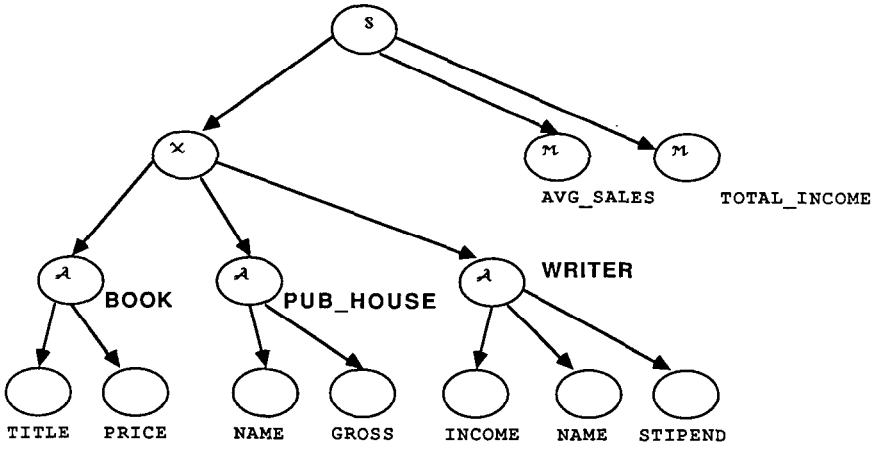

Figure 14. Cross product and summarization.

combining cross product and summarization, the designer can create statistical entities using attributes from several distinct entity types. Figure 14 illustrates a grouping that permits the gathering of summary statistics on financial aspects of publishing.

SAM* is oriented toward applications such as statistical databases and CAD/CAM, which involve nontraditional data. A number of distinct data types exist as primitives in SAM*. Furthermore, the G-relation provides explicit support for the representation of a wide diversity of information. The modeling constructs of SAM* were selected after an evaluation of the requirements of the conceptual modeling needs of the CAD/CAM environment. The developers of this model have implemented a prototype database management system IDMAS, based upon the SAM* model in order to evaluate the applicability of the modeling techniques to the computer-integrated manufacturing application domain [Krishnamurthy et al. 1987]. An enhanced model and system are presently at the design stage.

2.7 The Event Model

The event model [King and McLeod 1984, 1986] is a semantic data model providing support for generalization via functions, and aggregation through attributes. An interesting aspect of this model is its approach to dynamic modeling.

In the event model, a subtype relationship is used to organize the static schema into a set of hierarchies. Membership in a subtype is defined using predicates evaluated on attributes. For example, if DATABASE_BOOK is a subtype of BOOK, then a predicate would be specified within BOOK, defining the members of BOOK that qualify. For example,

Subtype: DATABASE_BOOK
all BOOK where topic = "database".

The attribute constraints listed below are built into the model [Farmer et al. 1985]:

(1) Primary: The attribute uniquely identifies the object.
(2) Ordered: The attribute may serve as a sort key.
(3) Single valued: The attribute has an atomic element as its value.
(4) Multivalued: The attribute may have a set as its value.

Thus, the semantics of an attribute depend on the constraints imposed by the schema designer in the definition of the attribute type. This provides the designer an additional degree of freedom. The definition of the BOOK type illustrates the static modeling facility of the event model:

TYPE: BOOK
primary attributes: ISBN—string (singled valued, nonnull)
dependent attributes:
AUTHOR—WRITER (multivalued, nonnull, ordered)
SALES—integer (single valued, ordered)
TITLE—string (single valued, nonnull, ordered)
PUBLISHER—PUB_HOUSE (single valued, nonnull, ordered)
BORROWER_PERSON (multivalued)
REVIEWERS—REVIEWER (multivalued)
REVIEWS—REVIEW (multivalued)
TOPIC—string (multivalued, ordered)

The methodology for interactive dynamic model design consists of a sequence of design phases that describe the dynamic structures (events) of the application. In the first phase of event design, a process of stepwise refinement is used to define the functions of the application environment. These are represented as a hierarchy of process and function links. Process events correspond to units of processing within the application environment. Function links are used to represent a hierarchy of events. A function link is defined between two events if one individual person or procedure is responsible for both or if one is logically embedded in the other. Figure 15 provides an example of events in the procurement of books. This methodology bears some resemblance to many EDP systems analysis techniques. The integration of this systems analysis approach with methods of conceptual data modeling distinguishes the event model from both other semantic models and classical EDP approaches.

Next, directed communication links are used to form paths indicating the flow of information in the database, as in Figure 16. The resulting diagrams resemble state diagrams or Petri nets, but they are presented at a more concrete level. Figure 16 shows the information flow for Obtain_publication. The last step includes resolving ambiguities and refining the design.

As the authors point out, this model is not appropriate for applications in which the flow of information is not fixed or routine. This model, however, does present a higher level modeling methodology for the dynamic aspects of databases when the behavior of the model is reasonably predictable. The design philosophy of this model, indicating not only the mechanism but the sequence by which information is extracted from the designer, could be extended to static modeling as well. If it aids a designer in specifying the semantics of an application, such a philosophy is well worth including in the model definition.

Ongoing work using the event model include Sedaco [Farmer et al. 1984, 1985], a tool for semantic database construction

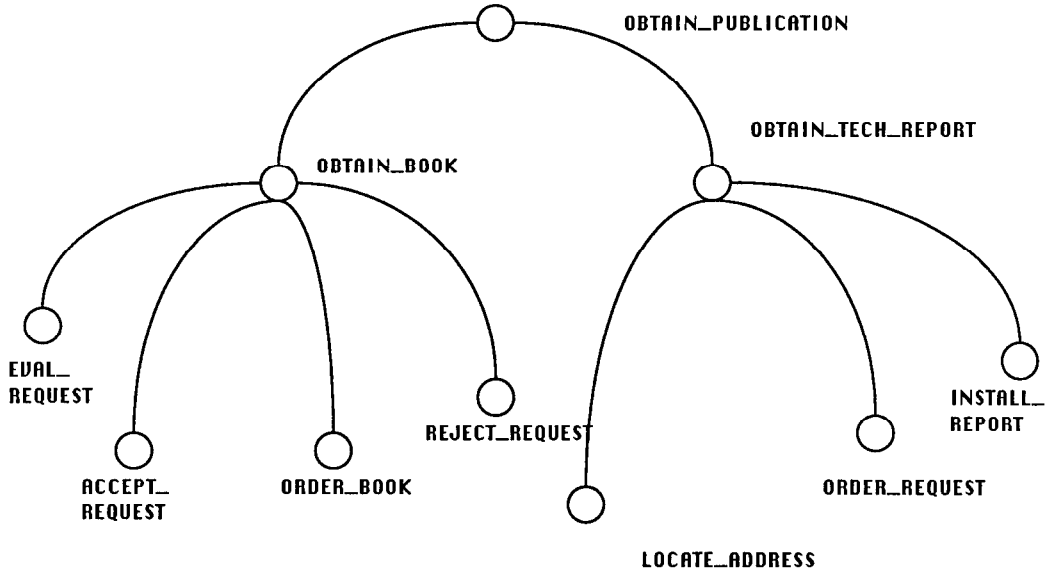

Figure 15. Specification of functional links.

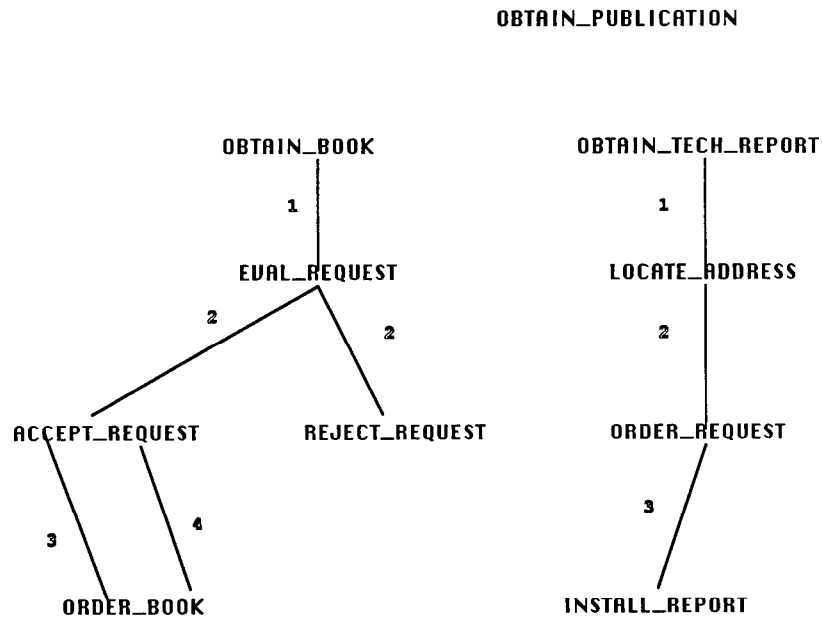

Figure 16. Flow of information.

identifying primitives to be used in database implementation, and Sembase [King 1984], a semantic database management system utilizing a graphics-based user interface.

2.8 SHM+

Like the previous model, SHM+ [Brodie 1984; Brodie and Ridjanovic 1984] addresses the problem of modeling both the

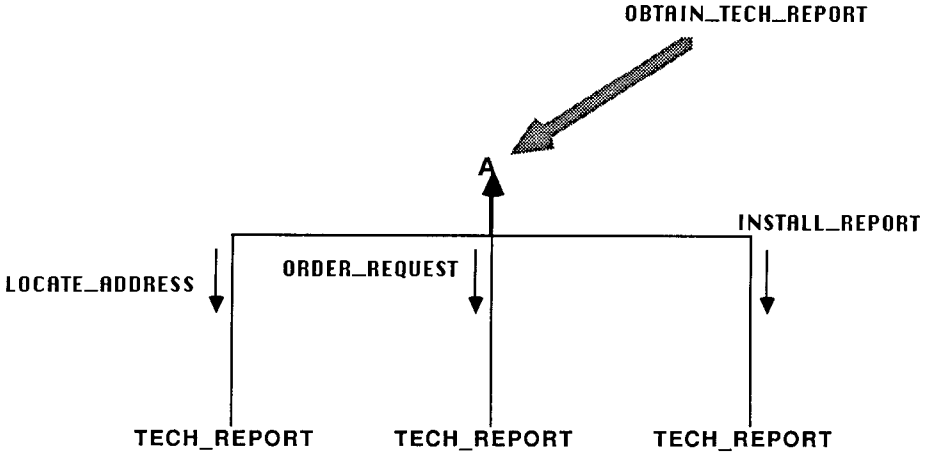

Figure 17. Sequential scheme.

static and dynamic portions of an application. Specification of data objects and associated transactions are performed using an abstract data type philosophy and a related modeling methodology. Object and behavior schemes, an alternative to relational tables and entity–relationship diagrams, are used to capture the object and operation schemas from the designer. These schemes are used to model both higher and lower level data, relationships, and operation objects, thus providing a unified structure for the expression of the semantics of the application.

The basic modeling constructs of SHM+ are primitive objects and operations, composition rules for hierarchically forming more complex objects and operations, and constraints to be applied to all primitives, composition rules, and hierarchies.

To model the static structure of the application, the gross properties of the objects and structural relationships are collected by use of object schemes. An example of a scheme utilizing aggregation is shown in Figure 2 in the Introduction. Similar schemes exist for generalization and association.

SHM+ supports inheritance through relationships other than those representing generalization. If we assume AUTHOR in Figure 2 refers to another object type, then the remaining attributes of that type can be inherited by the aggregate object PUBLICATION. This model not only encourages this perspective but permits the user to specify to what extent inheritance should be carried out (i.e., which attributes of AUTHOR should be thought of as being part of PUBLICATION). When a generalization hierarchy appears in a design, SHM+ enforces strict inheritance.

Behavior schemes are the explicit graphical representation of the gross properties of SHM+ transactions. The nodes of a behavior scheme represent the objects participating in the transaction, while the arcs are labeled by the operations applied to the objects. The structure of the graph defines the control flow of the transaction. Figures 17 and 18 illustrate the representation of a sequential computation and a caselike decision, respectively. The graphical representation of these control abstractions is identical to that used to represent the structural abstractions of aggregation and generalization. Thus, SHM+ offers a unified modeling methodology for both static and dynamic objects. Figure 19 presents the SHM+ definition of the OBTAIN_BOOK transaction.

The most important aspects of the SHM+ model are its contribution to dynamic data modeling and a consistent modeling methodology for both dynamic and static schemas. The similarity of the

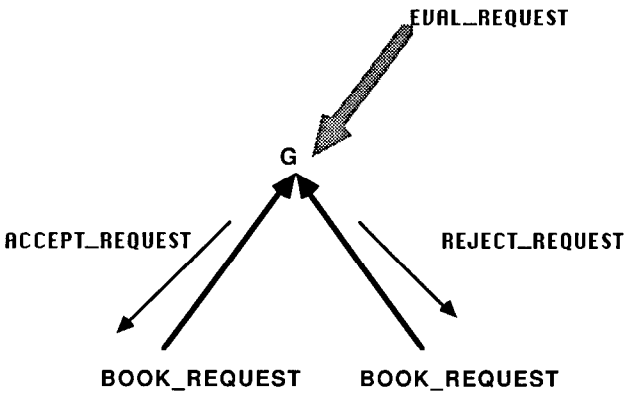

Figure 18. Case scheme.

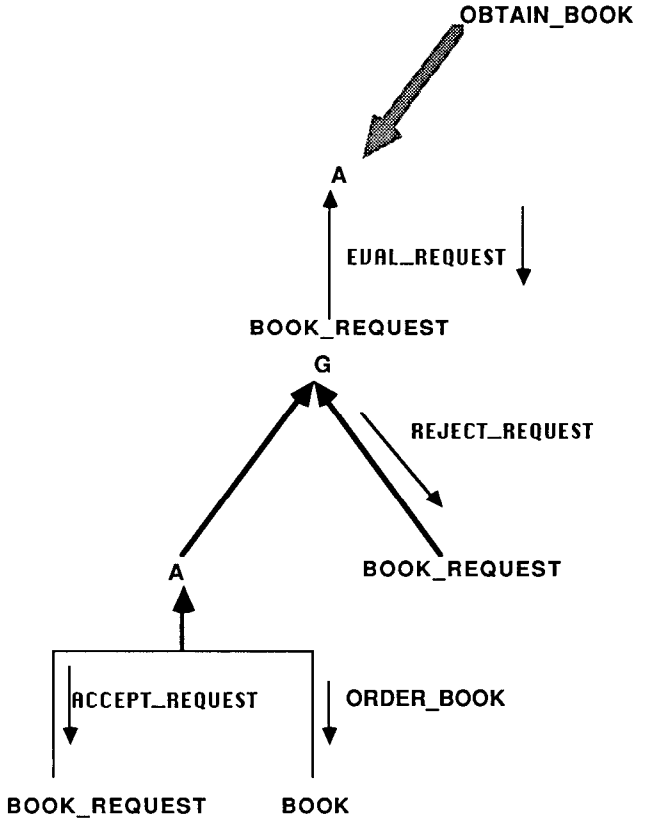

Figure 19. Behavior scheme.

constructs in Figures 2 and 17 illustrate the commonality of the approaches to modeling data and transactions in SHM+. Since the semantics of database objects always specify constraints on behavior of the database, it seems reasonable that the definition of transactions (dynamic objects also specifying database behavior) should be considered an integral part of database modeling.

3. COMPARISON OF MODELS

In this section, the objectives and offerings of the models described in Section 2 are

compared using the parameters suggested in Section 1. A technique for assessing the relative merits of these models is also given. Since the collection of data models is not uniform in terms of scope or objective, the evaluation contains a certain amount of subjectivity. Nevertheless, we present each model in terms of our earlier definitions of a "semantic model."

3.1 Tabular Comparison

Table 1 summarizes the features of the models surveyed. Owing to the diversity of models chosen, the information in the table is intentionally more general than specific. More specific parameters would leave many of the entries empty and would not provide a good means of comparison, especially with respect to dynamic modeling. A survey that limits the types of models compared to those providing some degree of dynamic modeling but provides separate detailed tables for static, dynamic, and temporal modeling constructs appears in Urban and Delcambre [1986].

An explanation of the terms used in the table but not defined in Section 1 follows:

(1) Unstructured object representation is classified as limited or enhanced, depending on the degree to which the system provides nontraditional data types.

(2) Relationship representation is considered to be independent, entities, tables, functions, or attributes, depending on the manner in which the model presents the relationships to the user. As mentioned before, many systems offer multiple relationship views to the user in the interest of semantic relativism.

(3) We classify the means by which insertion/deletion constraints are offered to the database designer within a given model as follows: If a user is provided a set of relationship types, each of which includes insertion/deletion rules, then we shall consider these rules built in. If these rules are further maintained automatically by the system, then we shall consider them automatic. If the user is provided with a choice of insertion/deletion semantics for each relationship type, then these constraints are user specified.

Notice that within the table, one finds a wide variety of modeling techniques. For example, whereas TAXIS emphasizes a generalization hierarchy as the major organizational feature for objects and operations, the entity–relationship model emphasizes networks. In this case, the subtle differences in the properties of hierarchy and a network have an impact on the application of the modeling methodologies. The absence of cycles in a hierarchy exactly corresponds to a prohibition against circular inheritance in TAXIS. Some other observations follow:

(1) Almost all models include generalization and aggregation as modeling abstractions. Classification and especially association seem to be not so widely acknowledged as fundamental modeling concepts. Most models tend to build around a single abstraction. For instance, TAXIS stresses generalization, whereas SDM defines all other properties within the class definition.

(2) Dynamic modeling as an integral part of the modeling process has gained the interest of only a handful of semantic model designers. In two of the three models capable of handling dynamic modeling, (TAXIS, SHM+, and the event model), a specific design methodology seems to be an integral part of the model; that is, the model not only provides modeling constructs but gives the designer a particular sequence of modeling activities used to specify the database model.

(3) There is great variety in the definition of insertion/deletion constraints. Some models (SAM*, SDM) specify these constraints as a part of the relationship definitions, others (functional data model, entity–relationship model) permit the user to specify which rules are desired, and still others (RM/T) permit a mixture of built-in constraints and user-defined constraints.

(4) There is also great variation in the support of derivation. Many models offer inheritance through a generalization relationship as the only form of derivation, whereas a model such as SDM provides many different varieties.

Table 1. Data Model Features

	Unstructured object representation	Relationship representation	Standard abstraction	Network versus hierarchy	Derivation/ inheritance	Insertion/ deletion constraints	Relationship semantics	Dynamic modeling
E–R	Limited	Independent and tables	Aggregation	Strong network	No	User specific	User selectable	No
TAXIS	Limited	Entity (classes)	Generalization Aggregation Classification	Strong hierarchy	Inheritance	User specific except in IS-A hierarchy	Predefined	Transaction modeling, object oriented
SDM	Limited	Independent and entity (classes)	Generalization Aggregation Classification Association	General hierarchy present	Elaborate and varied	Automatic	User defined	No
Functional	Limited	Functions	Association Aggregation Classification	No direct support for either	Functional	User specific	User defined	No
RM/T	Limited	Independent	Generalization Aggregation Classification Association	General hierarchy present	Inheritance	Automatic and user specific	Predefined	No
SAM*	Enhanced (special-purpose set built in)	Independent	Generalization Aggregation Classification Association	Network	Summation over classes inheritance	Automatic	Predefined	Object oriented
Event	Limited	Attributes	Generalization Aggregation	Hierarchy	No	Automatic	Predefined	Transaction modeling
SHM+	Limited	Attributes, entities, separate	Generalization Aggregation Association	General hierarchy present	Inheritance over generalization and association hierarchies	Built in	Predefined	Transaction modeling

(5) Only the most recent or the most specialized models provide rich sets of data types for modeling information that cannot be simply coded in ASCII. As the table indicates, this particular sampling of data models provides few good examples. There is evidence, however, that work in this direction is continuing [Copeland and Maier 1984; Woelk et al. 1986].

That the modeling philosophy is very different for each model is evidenced by the sections describing each model. Table 1 serves to emphasize these differences further. Although most models have been defined to address the general problem of conceptual modeling, many have been exercised in particular environments and have thus assumed characteristics best suited for those environments. In attempting to evaluate further the collection of models, one must decide on the purpose of the evaluation. Such an evaluation could proceed from one of two points of view:

(1) categorization of the models with the intention of identifying similarities and differences or
(2) selection of a best model for a specific environment.

In practice, a designer would face the latter decision. Given the general nature of this survey, however, we present parameters for categorizing semantic models and also discuss an approach to picking "the best model" for a given task.

3.2 Evaluating Semantic Models

Although substantial diversity of conceptual modeling approaches exists among the semantic data models considered, all of the models' authors seem to agree that the main objective is to facilitate the modeling of and the use of databases. Most agree that the following contribute to this objective:

(1) A semantic model should provide relationships between data objects that support the manner in which the user perceives the real-world enterprise.
(2) For these relationships a semantic model should contain semantics that specify the acceptable states, transitions, and responses of the database system.

The model developers vary in their perceptions of the following:

(1) Whether models should be application independent or targeted toward specific environments, such as SAM*.
(2) Whether the relationships should be highly developed packages, with all semantics (insertion/deletion constraints, cardinality constraints, etc) built in, as in SDM or SAM*, or whether the database designer should have the option to specify the semantics of each relationship in an explicit manner. The entity–relationship model provides the latter option, since for every relationship the designer indicates properties such as cardinality and insertion/deletion constraints.
(3) Whether relationships should be complex or primitive in their structure. Binary models [Abrial 1974; Azmoodeh et al. 1986; Bachman 1983; Bracci et al. 1976], for example, are data models in which all relationships are constructed from elementary binary relationships. Brodie [1984] classifies these as irreducible models since the information in the model is expressed with atomic rather than complex groups of facts. Tsichritzis and Lochovsky [1982] characterize binary models as the elementary graph-oriented models since the relationships are usually presented in graph form (the relational model being the elementary table-oriented model). To illustrate that it is possible to represent complex relationships using elementary binary relationships, consider an aggregation relationship used to form entities. For example, suppose a BOOK entity is composed of the attributes Title, Page-length, Author (reference to another entity), and Publisher (reference to another entity). One might perceive this as a 1:4 relationship between BOOK and the four attributes, relating each BOOK object to one occurrence of each of the attributes from the four binary relationships: BOOK–Title, BOOK–Page_length, and so on. Although it is possible to construct complex relationships using a binary model, it can be said that the

relationships presented to the user are elementary. This approach is different from that of a model such as SDM, which offers built-in higher order relationships.

(4) Whether relationships are distinguished from entities at the conceptual level. In some models, that is, E–R models, relationships act as primary modeling elements with different semantics than entities. On the other hand, in the functional model, both entities and relationships are represented as functions.

(5) Which abstractions should be emphasized. Certain models stress one or two abstractions as their primary modeling tools. SDM, for example, makes heavy use of classification and aggregation, whereas TAXIS stresses generalization hierarchies. Other approaches, such as SHM+, offer a wider range of abstractions. The developers of the models that provide a small number of primary modeling concepts believe that offering a few powerful options results in a more straightforward modeling process.

(6) What approach to dynamic modeling should be followed. SAM* provides abstract data types, in contrast to the generalization hierarchy approach in TAXIS and the flow of information approach of the event model.

The above criteria can serve as differentiating characteristics of semantic data models for a model evaluation. If the purpose of the evaluation is to identify similarities and differences among the models for the purpose of classification, then these characteristics can form the columns of a table such as Table 1. Using information of this nature, one may wish to organize the models into groups as in Brodie [1984] or to develop a continuum such as that presented in Figure 20.

The parameters of Table 1 can also be utilized to identify the model of choice for a particular task. In addition, consider the following three criteria [March et al. 1984], which can measure the ease of application of a given model to a particular problem domain by a specific designer:

(1) User's ability to understand the meaning (or semantics) of the modeling constructs provided. If the user has difficulty understanding or misunderstands the modeling tools, then the advantages of the semantic model are completely lost.

(2) Ease of query formulation. One impetus for the rise of semantic models was the awkwardness of manipulating the database on the physical and data structure levels.

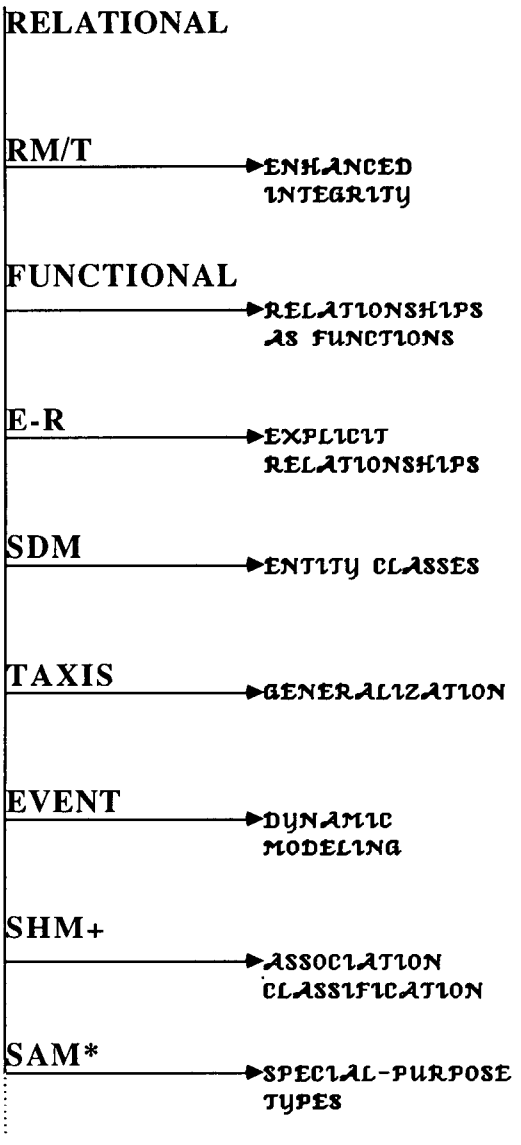

Figure 20. Data model continuum.

Table 2. Weighted Features

	Unstructured objects	Relationship representation	Standard abstraction	Network versus hierarchy	Derivation/ inheritance	Insertion/ deletion constraints	Relationship semantics	Dynamic modeling
Weight	0	25	15	10	15	10	20	5

Table 3. Data Model Comparison

	Unstructured objects	Relationship representation	Standard abstraction	Network versus hierarchy	Derivation/ inheritance	Insertion/ deletion constraints	Relationship semantics	Dynamic modeling	Total
Weight	0	25	15	10	15	10	20	5	—
M1	—	10	7	5	8	7	9	3	790
M2	—	6	8	5	6	7	4	8	600

(3) Ease of specification and maintenance of the semantics of the modeling constructs. Whether the offering of the model is a simple collection of insertion/deletion constraints or a sophisticated set of constraint definition tools, the objective should be the same: The semantics should be easily specified and automatically maintained.

One commonly used evaluation strategy involves the attachment of weights to the columns of the table, as in Table 2 where the weights indicate the relative importance of each metric to the problem domain. The rows of the table then hold the score indicating the merit of that model for the feature corresponding to each column. Table 3 presents the evaluation of hypothetical models M1 and M2 for a particular application environment. M1's higher total score reflects the evaluator's preference for the manner in which it describes relationships and a general feeling by the evaluator that M1 is "easier to understand."

Subjectivity is inherent in this evaluation process. When, however, one recalls that one of the goals of semantic data models is the production of a representation of the enterprise that closely parallels the user's perception, it is appropriate that the user's perception play a major role in the evaluation of a model for a given enterprise. Involving multiple evaluators in the process will mitigate the subjectivity to some degree.

4. WHERE DO WE GO NOW?

Although the formulation of abstractions and the support of relationships was of early and prime importance to researchers, there are other issues that have arisen within the realm of semantic data modeling. It is still important to explore the variety of meanings for the semantics of the objects, operations, and relationships of a model. There are, however, other ideas that also might satisfy the requirements outlined in the previous section and hence produce conceptual models that simplify the task of database design. The following paragraphs give (an eclectic) collection of ideas that will continue to be addressed by contemporary semantic data model researchers.

SAM* addresses the issue of supporting nontraditional data by providing data types that are useful in statistical applications and CAD/CAM. Other researchers feel that it is more useful to provide primitives to support the design of special types [Copeland and Maier 1984]. This provides a more flexible model—one that is not targeted to specific applications but that may require more complexity in the expression of specialized types. Modeling primitives for the design of text, sound (including voice), digitized images, and complex vector graphics appear in the approaches to the processing of multimedia information [Christodoulakis et al. 1986; Woelk et al. 1986] that utilize a meld of semantic data modeling principles and object-oriented language philosophies to provide support for these diverse and nonstandard database applications.

Two other areas in which substantive results are limited are those of dynamic and temporal modeling (meaning the inclusion of explicit time and event concepts). All semantic systems provide support for static modeling, but although researchers have discussed the importance of temporal [Snodgrass and Ahn 1986] and dynamic modeling, one finds only a few models that include these capabilities. Urban and Delcambre [1986] survey five models, RM/T, TAXIS, SDM, SHM+, and the event model, most providing dynamic and some providing limited temporal modeling features. A column containing information on temporal modeling in Table 1 would be very sparse indeed. Nevertheless, if conceptual modeling tools are to grow in their ability to describe an enterprise in a complete manner, this description must encompass behavior of a temporal nature as well as the representation of events.

A variety of methodologies for the representation of time has been proposed. Ariav [1986] has defined TODM, the temporally oriented data model, which effectively adds temporal data represented in cubic form into the relational model. Ariav's work extends the relational operations to include temporal data. Snodgrass and Ahn

[1985, 1986] have also used the relational model as the starting point for the representation of temporal data. They distinguish between transaction time, which is the time when information was stored in the database, and valid time, the time during which the information is useful. Using these two concepts, they define static rollback relations and historical relations. The approach taken by Shoshani and Kawagoe [1986] differs from the others in that they utilize time sequences, which are ordered sequences of time values, as the primary organizing concept. They also discuss the issue of physical organization for temporal data. Schiel [1983] proposes a temporal–hierarchic data model (THM) that incorporates time concepts as one of its fundamental abstractions. His work details the impact of temporal considerations on the semantics of generalization and specialization. Castilho et al. [1982] have developed a language for the inclusion of temporal semantics in the specification of a database schema. One of the key concepts found in several studies of temporal data is that of temporal constraints, which permit the user to include time factors, as well as data values, in the constraint conditions [Ariav 1986; Kung 1984].

To a large extent, the conceptual modeling requirements in new areas of information processing have driven the development of semantic data models. This survey has presented several semantic models that have the goal of satisfying the modeling needs of a wide range of enterprises. An alternative approach to addressing the need for extended data modeling capabilities is to generate data models automatically to fit the requirements of specific application environments. The Data Model Compiler [Maryanski et al. 1986, 1987] and EXODUS [Carey et al. 1986; Richardson and Carey 1987] are investigating this approach. The key conceptual issue in the development of a viable data model generator is the formal specification and representation of the semantics of the conceptual data models. A first step involves the identification of primitive abstractions for the representation of entities, relationships, operations, and constraints of the model. The language used in the Data Model Compiler project for the representation of data model semantics appears in Hong and Maryanski [1988]. The EXODUS project has developed the E programming language [Richardson and Carey 1987] as a primary specification mechanism. If the data model generator idea were to prove feasible, it would make possible the tailoring of semantic data models to the requirements of an enterprise, thus reducing the gap between the model in the mind of the designer and its representation within the database system.

As evidenced by the models presented here, semantic data modeling is presently in the research stage. The commercial marketplace will not move beyond the relational model until semantic database systems with reasonable performance characteristics emerge. Chronologically, semantic data models postdate the relational model by approximately 7 years. Fledgling products have begun to emerge; mature systems can be expected in a few years. The target marketplace for these new products is not expected to be commercial data processing but rather the management of scientific, engineering, and manufacturing data. The impact of semantic data models on the commercial market is likely to be limited by the following.

(1) Inertia in the commercial market—the process of converting enormous amounts of data to a new model will not occur until the expected benefit clearly exceeds the cost.
(2) Opportunity in the scientific/engineering/manufacturing market—a well-known need for advanced data models already exists in this space.

A reasonable forecast for the conceptual modeling future would project an emulation of the developments in programming languages. Some languages appear to be forever entrenched among practitioners (i.e., FORTRAN and COBOL), with new languages constantly surfacing, many having a substantial impact on both researchers and application programmers. A designer will always seek the best model for the task.

Since the complexity of the applications will continue to increase, the designer's requirements of a conceptual model will similarly heighten, and hence new models will continue to emerge. The collection of models surveyed in this paper is representative of the next, but not the last, generation of data models.

ACKNOWLEDGMENTS

The work of J. Peckham and F. Maryanski was partially supported by grant ECS-8401487 from the National Science Foundation. The work of J. Peckham was partially supported by an IBM teaching fellowship. The work of F. Maryanski was partially supported by grant IRI-8704042 from the National Science Foundation.

REFERENCES

ABRIAL, J. R. 1974. Data semantics. In *Data Base Management*, J. W. Klimbie and K. L. Koffemen, Eds. North-Holland, Amsterdam, pp. 1–59.

ARIAV, G. 1986. Temporally oriented data models. *ACM Trans. Database Syst. 11*, 4 (Dec.), 499–527.

AZMOODEH, M., LAVINGTON, S. H., AND STANDRING, M. 1986. The semantic binary relationship model of information. In *Research and Development in Information Retrieval, Proceedings of the 3rd Joint BCS and ACM Symposium*. Cambridge University Press, Cambridge, UK.

BACHMAN, C. W. 1983. The structuring capabilities of the molecular data model. In *Entity–Relationship Approach to Software Engineering, Proceedings of the 3rd International Conference on Entity–Relationship Approach* (Anaheim, Calif.), C. G. Davis et al., Eds. North-Holland, Amsterdam.

BORGIDA, A., MYLOPOULOS, J., WONG, H. K. T. 1984. Generalization/specialization as a basis for software specification. In *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*, M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, Eds. Springer-Verlag, New York, pp. 87–114.

BRACCI, G., PAOLINI, P., AND PELAGATTI, G. 1976. Binary logical associations in data modelling. In *Modelling in Database Management Systems, Proceedings of IFIP TC2 Conference* (Freudenstadt, W. Germany), G. M. Nijssen, Ed. North-Holland, Amsterdam, pp. 125–148.

BRAEGGER, R. P., DUD ZER, A., REBSAMEN, J., AND ZEHNDER, C. 1985. Gambit: An interactive database design tool for data structures, integrity constraints, and transactions. *IEEE Trans. Softw. Eng. SE-11*, 7, 574–582.

BRODIE, M. L. 1984. On the development of data models. In *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*, M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, Eds. Springer-Verlag, New York, pp. 19–48.

BRODIE, M. L., AND RIDJANOVIC, D. 1984. On the design and specification of database transactions. In *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*, M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, Eds. Springer-Verlag, New York, pp. 277–32.

BRODIE, M. L., AND SILVA, E. O. 1983. Active and passive component modelling: ACM/PCM. In *Proceedings of the IFIP WG8.1 Working Conference*, T. W. Olle, et al., Eds. North-Holland, Amsterdam, pp. 41–92.

BRYCE, D., AND HULL, R. 1986. SNAP: A graphics-based schema manager. In *Proceedings of the International Conference on Data Engineering* (Los Angeles, Calif.). IEEE, New York, pp. 151–164.

BUNEMAN, O. P., AND NIKHIL, R. 1984. The functional data model and its uses for interaction with databases. In *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*, M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, Eds. Springer-Verlag, New York, pp. 359–380.

BURNS, T., FONG, E., JEFFERSON, E., KNOX, R., MARK, L., REEDY, C., REICH, L., ROUSSOPOULOS, N., AND TRUSZOWSKI, N. 1986. Reference Model for DBMS Standardization. Database Architecture Framework Task Group (DAFTG) of the ANSI/X3/SPARC Database System Study Group. *ACM Sigmod Rec. 15*, 1, 19–58.

CAREY, M., DE WITT, D., FRANK, D., GRAEFE, G., MURAZIKRISHRA, H., RICHARDSON, J., AND SHEKITA, E. 1986. The architecture of the EXODUS extensible database system. In *Proceedings of the International Workshop on Object-Oriented Database Systems* (Pacific Grove, Calif.). IEEE, New York, pp. 52–65.

CASTILHO, J. M. V., CASANOVA, M. A., AND FURTADO, M. L. 1982. A temporal framework for database specifications. In *Proceedings of the 8th International Conference on Very Large Data Bases* (Mexico City). Very Large Data Base Endowment, Saratoga, Calif., pp. 280–291.

CHEN, P. 1976. The entity–relationship model: Toward a unified view of data. *ACM Trans. Database Syst. 1*, 1 (Mar.), 9–36.

CHEN, P., Ed. 1985. *Entity–Relationship Approach: The Use of the ER Concept in Knowledge Representation*. North-Holland, Amsterdam.

CHRISTODOULAKIS, S., HO, F., AND THEODORIDOU, M. 1986. The multimedia object presentation manager of MINOS: A symmetric approach. In *Proceedings of the ACM SIGMOD Conference* (Washington, D.C.). ACM, New York, pp. 295–310.

CODD, E. F. 1970. A relational model of data for large shared data banks. *Commun. ACM 13*, 6 (June), 377–387.

CODD, E. F. 1972. Further normalization of the database relational model. In *Data Base Systems*, R. Rustin, Ed. Prentice-Hall, Englewood Cliffs, N.J., pp. 33–64.

CODD, E. F. 1979. Extending the database relational model to capture more meaning. *ACM Trans. Database Syst. 4*, 4 (Dec.), 397–434.

COPELAND, G., AND MAIER, D. 1984. Making Smalltalk a database system. In *Proceedings of the ACM SIGMOD Conference* (Boston, Mass.). ACM, New York, pp. 316–325.

DATE, C. 1983. *Introduction to Database Systems*, vol. 2. Addison-Wesley, Reading, Mass.

DAYAL, U., AND DITTRICH, K., Eds. 1986. In *Proceedings of International Workshop on Object-Oriented Database Systems* (Pacific Grove, Calif.). IEEE, New York.

DITTRICH, K. 1986. Object-oriented database systems: The notions and the issues. In *Proceedings of the International Workshop on Object-Oriented Database Systems* (Pacific Grove, Calif.). IEEE, New York, pp. 2–4.

FAGIN, R. 1977. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst. 2*, 3 (Sept.), 262–278.

FAGIN, R. 1979. Normal forms and relational database operators. In *Proceedings of the ACM SIGMOD Conference* (Boston, Mass.). ACM, New York, pp. 153–160.

FARMER, D., KING, R., AND MYERS, D. 1984. A tool for the implementation of databases. In *Proceedings of the International Conference on Data Engineering* (Los Angeles, Calif.). IEEE, New York, pp. 386–393.

FARMER, D., KING, R., AND MYERS, D. 1985. The semantic database constructor. *IEEE Trans. Softw. Eng. SE-11*, 7, 583–591.

HAMMER, M., AND MCLEOD, D. 1981. Database description with SDM: A semantic database model. *ACM Trans. Database Syst. 6*, 3 (Sept.), 351–386.

HONG, S., AND MARYANSKI, F. 1988. Representation of object-oriented data models. *Inf. Sci.* To be published.

HULL, R., AND KING, R. 1987. Semantic database modeling: Survey, applications, and research issues. *ACM Comput. Surv. 19*, 3 (Sept.), 201–260.

JARDINE, D. A., Ed. 1977. *The ANSI/SPARC DBMS Model.* North-Holland, The Netherlands.

KERSCHBERG, L., KLUG, A., AND TSICHRITZIS, D. 1976. A taxonomy of data models. In *Systems for Large Data Bases*, P. C. Lockemann and E. J. Neuhold., Eds. North-Holland, Amsterdam, pp. 43–64.

KING, R. 1986. A database management system based on an object oriented model. In *Proceedings of the International Workshop on Expert Database Systems* (Charleston, S.C.). University of South Carolina, pp. 443–468.

KING, R., AND MCLEOD, D. 1986. The event database specification model. In *Proceedings of the 2nd International Conference on Databases: Improving Usability and Responsiveness* (Jerusalem, Israel). IIPA, pp. 299–322.

KING, R., AND MCLEOD, D. 1984. A unified model and methodology for conceptual database design. In *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*, M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, Eds. Springer-Verlag, New York, pp. 313–327.

KRISHNAMURTHY, V., SU, S., LAM, H., MITCHELL, Z., AND BANCMEYER, E. 1987. A distributed database architecture for an integrated manufacturing facility. In *Proceedings of International Conference on Data and Knowledge Systems for Manufacturing and Engineering* (Hartford, Conn.). IEEE, New York, pp. 4–13.

KUNG, H. 1984. A temporal framework for database specification and verification. In *Proceedings of the 10th International Conference on Very Large Data Bases* (Singapore). Very Large Database Endowment, Saratoga, Calif., pp. 91–99.

MARCH, S. T., Ed. 1988. In *Proceedings of the 6th Entity–Relationship Conference.* North-Holland, Amsterdam. To be published.

MARCH, S. T., RIDANOVIC, D., AND PRIETULA, F. 1984. On the effects of normalization on the quality of relational database designs or being normal is not enough. In *Proceedings: Trends and Applications 1984, Making Databases Work* (Gaithersburg, Md.). National Bureau of Standards, pp. 257–261.

MARYANSKI, F. 1986. The data model compiler: A tool for generating object-oriented database systems. In *Proceedings of the Workshop on Object-Oriented Database Systems* (Pacific Grove, Calif.). IEEE, New York, pp. 73–84.

MARYANSKI, F., FRANCIS, S., HONG, S., AND PECKHAM, J. 1987. Generation of conceptual data models. *Data and Knowledge Engineering.* To be published.

MYLOPOULOS, J., BERNSTEIN, P. A., WONG, H. K. T. 1980. A language facility for designing database-intensive applications. *ACM Trans. Database Syst. 5*, 2 (June), 185–207.

NIXON, B., CHUNG, L., LAUZEN, I., BORGIDA, A., MYLOPOULOS, J., AND STANLEY, M. 1987. Implementation of a compiler for a semantic data model: Experience with Taxis. In *Proceedings of the ACM SIGMOD Conference* (San Francisco, Calif.). ACM, New York, pp. 118–131.

O'BRIEN, P. 1983. An integrated interactive design environment for Taxis. In *Proceedings of SOFTFAIR: A Conference on Software Development*

Tools, Techniques, and Alternatives (Silver Spring, Md.). IEEE, New York, pp. 298–306.

OLLE, T. W. et al., Eds. 1982. Information systems design methodologies: A comparative review. In *Proceedings of the IFIP WG 8.1 Working Conference*. North-Holland, Amsterdam.

OLLE, T. W. et al., Eds. 1983. Information systems design methodologies: A feature analysis. In *Proceedings of the IFIP WG 8.1 Working Conference*. North-Holland, Amsterdam.

OLLE, T. W. et al., Eds. 1986. Information systems design methodologies: Improving the practice. In *Proceedings of the IFIP WG 8.1 Working Conference*. North-Holland, Amsterdam.

PIROTTE, A. 1977. The entity–association model: An information oriented data base model. In *Proceedings of the International Computing Symposium*. North-Holland, Amsterdam, pp. 581–597.

RICHARDSON, J. E., AND CAREY, M. J. 1987. Programming constructs for database system implementation in EXODUS Databases. In *Proceedings of the SIGMOD Conference* (San Francisco, Calif.). ACM, New York, pp. 208–219.

SCHIEL, U. 1983. An abstract introduction to the temporal-hierarchic data model (THM). In *Proceedings of the 9th International Conference on Very Large Data Bases* (Florence Italy). Very Large Database Endowment, Saratoga, Calif., pp. 322–330.

SCHMID, H. A., AND SWENSON, J. R. 1975. On the semantics of the relational data model. In *Proceedings of the ACM SIGMOD Conference* (San Jose, Calif.). ACM, New York, pp. 211–223.

SHIPMAN, D. W. 1981. The functional data model and the data language DAPLEX. *ACM Trans. Database Syst. 6*, 1 (Mar.), 140–173.

SHOSHANI, A., AND KAWAGOE, K. 1986. Temporal data management. In *Proceedings of the 12th International Conference on Very Large Data Bases* (Kyoto, Japan). Morgan Kaufman, Los Altos, Calif., pp. 79–88.

SMITH, J. M., AND SMITH, D. C. P. 1977. Database abstractions: Aggregation and generalization. *ACM Trans. Database Syst. 2*, 2 (Mar.), 105–133.

SNODGRASS, R., AND AHN, I. 1985. A taxonomy of time in databases. In *Proceedings of the ACM SIGMOD Conference* (Austin, Tex.). ACM, New York, pp. 236–246.

SNODGRASS, R., AND AHN, I. 1986. Temporal databases. *Computer 19*, 9, 35–46.

SPACCAPIETRA, S., Ed. 1987. *Entity–Relationship Approach: Ten Years of Experience*. North-Holland, Amsterdam.

SU, S. Y. W. 1983. SAM*: A semantic association model for corporate and scientific-statistical databases. *Inf. Sci. 29*, 151–199.

SU, S. Y. W. 1986. Modeling integrated manufacturing data with SAM*. *Computer 19*, 1, 34–49.

TAYLOR, R. W., AND FRANK, R. L. 1976. CODASYL data-base management systems. *ACM Comput. Surv. 8*, 1 (Mar.), 67–103.

TEOREY, T. J., YANG, D., AND FRY, J. P. 1986. A logical design methodology for relational databases using the extended entity relationship model. *ACM Comput. Surv. 18*, 2 (June), 197–222.

TSICHRITZIS, D., AND LOCHOVSKY, F. 1976. Hierarchical database management: A survey. *ACM Comput. Surv. 8*, 1 (Mar.), 105–123.

TSICHRITZIS, D., AND LOCHOVSKY, F. 1982. *Data Models*. Prentice-Hall, Englewood Cliffs, N.J.

URBAN, S. D., AND DELCAMBRE, L. M. L. 1986. An analysis of the structural, dynamic and temporal aspects of semantic data models. In *Proceedings of the International Conference on Data Engineering* (Los Angeles, Calif.). IEEE, New York, pp. 382–389.

VERHEIJEN, G. M. A., AND VAN BEKKUM, J. 1982. NIAM: An information analysis method. In *Proceedings of the IFIP WG 8.1 Working Conference*, T. W. Olle et al., Eds., North-Holland, Amsterdam.

WOELK, D., KIM, W., AND LUTHER, W. 1986. An object-oriented approach to multimedia databases. In *Proceedings of the ACM SIGMOD Conference* (Washington, D.C.). ACM, New York, pp. 311–325.

Received November 1986; final revision accepted October 1987.