Abstraction in Databases

Dennis McLeod

University of Southern California

John Miles Smith

Computer Corporation of America

Abstract

This paper surveys current research and practice concerning abstraction in database systems. Classical and semantic database models are reviewed and emphasized, as fundamental database abstraction mechanisms.

1. Introduction

The purpose of this paper is to provide an overview of the goals, issues, and technical results of database systems research concerning data abstraction and conceptual data modelling. This paper specifically reviews the following:

1. the essential concepts and goals (as related to data abstraction) used by researchers in the database systems area,
2. the role of abstraction in database systems research and database management technology,
3. the purpose and nature of conceptual database modelling formalisms (semantic database models).

To reach a consensus among the active researchers on these matters is obviously impossible. Our goal, then, is to summarize the main ideas and concepts, to provide a framework for further discussion and explication. In particular, the emphasis here is on establishing a view of database research that might facilitate the exchange of ideas among researchers who are specifically concerned with:

- *conceptual (semantic) database modelling techniques*, in the database systems research area,
- *abstract datatypes* and *program specification techniques*, in the programming language domain,
- *knowledge representation*, in the artificial intelligence community.

This research was supported, in part, by the Joint Services Electronics Program through the Air Force Office of Scientific Research under contract F44620-76-C-0061.

2. The Scope of Database Technology

The vast majority of research in the database systems area concerns techniques to support *database management systems (DBMSs)*. A DBMS is a general-purpose tool that accommodates the logical structuring, physical storage, and control of data, and that provides access interface(s) to databases. A DBMS is a general-purpose facility, in the sense that it can be directly applied to a variety of application domains. Of course, most of the techniques developed to support general-purpose DBMSs can be directly applied to a database-intensive application system (a *database system*) that is tailored to a specific *application environment*; however, it is important to understand this domain-independent orientation of database systems research, as it has a strong impact on the nature of the techniques that have been developed.

Database management systems represent a highly successful technology, which is rapidly advancing. While contemporary DBMSs can be indispensible components of overall information systems, a DBMS is not a panacea for all types of information management problems. Specifically, contemporary database management systems impose restrictions on the types of databases they are suited to accommodate [Hammer+McLeod 1979]. In a DBMS, databases are generally limited to "structured" or "formatted" data, i.e., the data is logically organized in the form of discrete records, each containing a specified number of atomic fields. Examples of collections of information that do not directly satisfy this requirement are text/document files, and aggregates of signal data; both of these are examples of "continuous" or "unstructured" data. It is also assumed that a database managed by a DBMS is considerably larger than its description; it is intended that a database contain a comparatively small number of different kinds of data, and many instances of each kind. In typical DBMS terminology, this means that the number of record types is much smaller than the number of record occurrences. This is in contradistinction to so-called "knowledge bases", which typically contain a great many "types" of information, and one or a few instances of each type.

By implicitly making these assumptions about the nature of databases, contemporary DBMSs typically have an important but limited applicability within an overall information system. The presence or absence of specific kinds of features in a DBMS further constrain its applicability and usefulness. For example, contemporary DBMSs do not provide adequate facilities for manipulating text/document files (e.g., for accomplishing full text search); current technology in "information retrieval" must be called upon to address this problem.

3. Abstraction in Database Systems

A fundamental focus of much recent research in the database systems area has concerned *database abstraction*. As a working definition, *abstraction* is a knowledge manipulation technique, the essential purpose of which is to substitute a description of the essence of a concept for the concept itself. As such, the process of abstraction maps a detailed, specific model of a concept into a higher-level, less detailed one.

Database abstraction refers to abstraction applied in the database context. A fundamental concern of database abstraction focuses on *representation independence*. In the context of database systems, the desire to separate the meaning of data from its computer-oriented representation has given rise to the notion of *(physical) data independence*. The principal motivation for data independence is twofold:

1. to accommodate database evolution, viz., the ability to change the physical organization of a database without changing its logical structure,

2. to free the user from the burden of dealing with the physical storage and access detail associated with a database.

The mechanism normally used to support data independence is a (logical/conceptual) *database model*. A database model is a formalism for expressing the logical structure of a database, and for providing a basis for manipulating such a database; the actual representation of the data is separately specified, at a lower level of abstraction. Specifically, a database model consists of four logical components:

1. a *data space*, which consists of a set of atomic *elements*, and certain *relationships* among them,

2. *type definition constraints*, which specify restrictions on the relationships in the data space,

3. *manipulation operations*, which allow elements to be created and destroyed, and their relationships modified,

4. a *predicate language*, which allows individual elements to be identified by their logical properties (and selected from the database).

The ANSI/X3/SPARC database system architecture [ANSI/X3/SPARC 1975] is an attempt at generalizing the various approaches that have been taken to describing data in a DBMS. In this architecture, a database is specified on three levels:

1. The *conceptual schema* is the central level of specification, and is intended to specify the meaning of the data; it is constructed using a (logical) database model.

2. Beneath the conceptual schema sits the database's *physical design*, which includes the storage structures and access methods used to represent and access data within a computer system.

3. Finally, atop the conceptual schema are one or more *external schemas*, which define reorganizations or reformattings of data described in the conceptual schema.

These three levels represent levels of data abstraction in a database system. While the issues associated with defining each of these levels and the mappings between such levels can be usefully examined in considerable detail, the remainder of this paper focuses on the nature of the database model used to define a conceptual schema; in our view this is the most fundamental concern with regard to database abstraction.

Much recent database system research has focused on the development and use of *semantic database models*. The issue here is that even though all representation detail has been abstracted away, there remains sufficient complexity in the logical structure as to require further abstraction; it is a point of controversy as to whether the same abstraction mechanisms are suited to both physical and logical abstraction. The purpose of a semantic database model is to provide user-understandable specifications of data, and to capture a substantial portion of the meaning of the data (in order to convey it to users).

4. Classical Database Models

To appreciate the significance of semantic database models, it is helpful to briefly review the development of their precursors - the "classical" database models. These models can be divided into three groups: the *hierarchic* group [Tsichritzis+Lochovsky 1976], the *network* group [Taylor+Frank 1976], and the *relational* group [Chamberlin 1976]. The development of these models has been largely driven by a single problem: to represent in an acceptably efficient manner the activities of a business enterprise. The models contain, to varying extents, both a physical aspect (which relates to implementation efficiency), and a logical aspect (which relates to modelling semantics). As noted above, this paper focuses on the logical aspect; the physical aspect is not further considered here.

The data space components (consisting of a set of atomic elements, and certain relationships among them) of the classical database models are identical. The "intended" interpretation of a data space is that each individual thing in the application environment maps onto one and only one element in the data space; moreover, all classical models *partition* the data space into two classes of elements: *types* and *instances*. An unlimited

number of named relationships is allowed between instances. However, elements that are types are only allowed to participate in a single relationship, viz., the "member-of" relationship between an instance and a type.

The type definition constraints supported by the various classical models are similar in a number of ways, but there are also some important differences among the models. All classical models allow only one-to-many (i.e., functional) relationships to be expressed. The expression of many-to-many relationships requires a nonstandard interpretation where multiple elements represent a single individual. Such nonstandard interpretations require the user/programmer to assume certain responsibilities for correctness otherwise born by the database system. The restriction to functional relationships applies particularly to the "member-of" relationship; this means that elements can only belong to one type (and thus precludes the creation of subtypes).

The differences among the type definition constraints of the classical database models relate to the limitations on the topology of relationships. Consider a directed graph in which types are nodes and there is an edge from type T1 to type T2 if there is a functional relationship from the elements of T1 to the elements of T2. The hierarchic models require that each node has at most one out-going edge and that no cycles occur. Neither the network models nor the relational models make any restriction. The primary motivation for the restriction in the hierarchic model is to allow an efficient implementation scheme to be utilized; this is an example of the strong impact that implementation considerations have had on the nature of classical database models.

All classical models support manipulation operations for instance creation and destruction, and for relationship modification. Facilities for manipulating the types are separate, and are usually provided in the form of commands for the database designer and administrator (e.g., "create type", "delete type", etc.); in general, specifications and operations on the data (instances) are separate from those regarding the schema (types, constraints, etc.).[1] One specific way in which the classical models differ with regard to manipulation operations is due to the presence or absence of requirements for functional relationships to be *total*. Hierarchic models require all functions to be total; this means that the creation/destruction of one instance may be coupled to the simultaneous creation/destruction of another. Network models allow functions to be explicitly declared as either "necessarily total" or "allowed to be partial". The relational models treat all functions as "allowed to be partial"; in this case, totality must be imposed as an external constraint when necessary.

[1] In fact, this separation is sometimes reflected in the architecture of the DBMS; a *data dictionary*, separate from the database itself, is sometimes used to record information on the schema.

The predicate language of a classical database model supports the identification of individuals (in a database) by their logical properties; this provides a capability for retrieving selected individuals from the database. Significantly, there are major differences between the classical models with regard to predicate language. The hierarchic and network models only allow boolean predicates to be formed, while the relational models allow any first-order predicate. In the first-order predicate language, variables are only allowed to range over instances, not over types. This implies, among other things, that types cannot be output as the result of a query.

5. Semantic Database Models

Semantic database models attempt to remove many of the restrictions imposed by the classical models, and also attempt to develop higher-level primitives suitable for special application environments. In this section, the main concepts underlying recent developments in semantic database models are reviewed. In the discussion, a number of specific semantic database models and research efforts are referenced; this discussion is not exhaustive, but is rather intended to highlight and exemplify the concepts.

The data space component of a semantic database model differs in several ways from that associated with classical database models. Neutrality and relativism are dominant themes in recent semantic models [Hammer+McLeod 1979, Hammer+McLeod 1980, McLeod+King 1979, Smith+Smith 1978, Smith+Smith 1979]. The point is that there is no rigid distinction between a type and an instance, or between an individual and a set of individuals; interpretation depends upon use. For example, a "ship-type" may be viewed as an individual or as a type.[2] Three main types of relationships have been identified as basic in semantic models [Codd 1979, Hammer+McLeod 1978, Hammer+McLeod 1980, Lee+Gerritsen 1978, Mylopoulos et. al 1978, Palmer 1979, Smith+Smith 1977a, Smith+Smith 1977b, Smith+Smith 1979]:

1. The "has-subtype" relationship associates an element (viewed as a type) with another (a subtype of the former element); for example, the type "tankers" is a subtype of the type "ships". Another way of looking at this same relationship type is as the "is-a" relationship among elements (viewed as instances), e.g., a "tanker" is a "ship".

2. The "has-attribute" relationship associates an element (viewed as an instance) with another, e.g., a "married person" has a "spouse". A relationship defined in this way can also be viewed as an element itself, e.g., "marriage".

3. The "has-instance" relationship logically connects an element (viewed as a type) with another element (viewed as an instance); the latter is an instance of the former, e.g., a particular individual is an instance of the type "person". When viewing a type as a set

[2] Moreover, it may even be reasonable to also view a type as a set of instances (the set of elements that are instances of the type).

of instances, the "has-instance" relationship type is "has-member".

These three kinds of relationships among elements model the basic ways in which types and instances can be interrelated: "has-subtype" relates a type with a type; "has-attribute" relates and instance with other instances; "has-instance" relates a type with instances.

In order to provide structure on the data space, semantic database models provide facilities for imposing constraints on the relationships in the data space. The key goal of this structuring is abstraction: in this case, abstracting relationships into generic kinds. For example, generic kinds of "has-attribute" relationships are associated with specific types; e.g., type "married person" may have a relationship kind "married-to", which associates each "married person" instance with another "married person" instance. Note that relationship occurrences can themselves be viewed as elements (instances); this is an example of the trend in semantic database models towards the unification of schema and data (in contradistinction to the classical database models and the DBMSs that support them).

As a further specific point of departure from classical database models, semantic models have recognized the fundamental importance of supporting many-to-many relationships among elements. One-to-many and one-to-one relationships are viewed as special cases of many-to-many, and a generic kind of relationship can be constrained to be one-to-many or one-to-one. Clearly, some mechanism is required to support the specification of these (and other) kinds of important constraints; two main alternative approaches have been proposed for handling constraint specification:

1. A general-purpose constraint specification language can be provided [Eswaran+Chamberlin 1975, Hammer+McLeod 1975, Stonebraker 1974].

2. The most important types of constraints can be identified,[3] and built into the database model. This approach is the one that has been mostly followed for semantic database models. Examples of the types of constraints that are built into semantic database models are: functionality (constraining a relationship to be one-to-many), totality, etc.

Most semantic database models have an associated set of manipulation operations, which support the creation, deletion, and modification of elements. One of the most important issues here concerns the level of specification and implicitness of modification dependencies. Suppose for instance, that "married person" is defined as a subtype of "person", and that a new married person is added to the database; is a new person instance automatically created, or must an explicit operation invocation be supplied? The central notion here is that of "oneness" [Kent 1978, Kent 1979]; is the thing in the database simultaneously a person and a married person, or are there two things that are related? While specific approaches on this matter vary, the consensus seems to be that the user ought to see a single individual as an instance of several types; this is in distinction to the classical database model approach, where the intended interpretation is that a single thing in the application environment maps onto a single instance of a given type in the database.

To support the identification and selection (retrieval) of elements from a database, the recent trend in semantic database models is to provide a unified treatment of instances and types. That is, a general-purpose predicate language and/or a set of primitive operations is provided to support the selection of elements. This is another example of the trend toward the unification of schema and data, and represents a significant point of departure from classical database models.

One of the most important ways in which current semantic database models are lacking is in their support of specification of operations specific to an application environment: facilities for combining the data selection and manipulation primitives are limited in most semantic models. This is a consequence of the emphasis in database semantic modelling research on data, rather than process.

A critical issue in extending semantic models to accommodate the specification of procedures is the careful meshing of the operations with the schema; this means that the operation specification language must be integrated into the semantic model's specification language, and that the operations must be organized in some way that relates them to the data.[4] A particular approach that has been taken to accommodating application environment specific retrieval operations is to view them as derived data, e.g., as relationships whose values are calculated using some declarative derivation specification [Hammer+McLeod 1980] or function definition [Buneman+Frankel 1979, Shipman 1980]. While it is clear that the abstract datatype approach is of considerable relevance here, there remain a number of essential problems that are topics of current research [Hammer 1976]; for example, abstract datatypes require an operation to be associated with a single type, but semantic database models seem to require operations involving multiple elements (instances of different types) to be logically associated with several types. In sum, the integration of operations with data in the context of semantic database models is an important area for further research.

[3] There is certainly no consensus at the present time as to which constraints are most important and fundamental.

[4] One current approach to this meshing is provided in [Hammer+Berkowitz 1980].

[Lum et. al. 1979]
Lum, V., S. Ghosh, M. Schkolnick, D. Jefferson, S. Su, J. Fry, T. Teorey, and B. Yao, *1978 New Orleans Data Base Design Workshop Report*, IBM Research Report RJ554, San Jose CA, 13 July 1979.

[McLeod 1977]
McLeod, D. J., "High Level Definition of Abstract Domains in a Relational Database System", *Journal of Computer Languages*, Volume 2, Number 3, 1977.

[McLeod+King 1979]
McLeod, D. and R. King, "Applying a Semantic Database Model", *Proceedings of International Conference on the Entity-Relationship Approach to Systems Analysis and Design*, Los Angeles CA, 10-12 December 1979.

[Mylopoulos et. al. 1978]
Mylopoulos, J., P. A. Bernstein, and H. K. T. Wong, "A Language Facility for Designing Interactive Database-Intensive Applications", *Proceedings of ACM SIGMOD International Conference on the Management of Data*, Austin TX, 31 May - 2 June 1978.

[Palmer 1978]
Palmer, I., "Record Subtype Facilities in Database Systems", *Proceedings of the Fourth International Conference on Very Large Databases*, West Berlin, West Germany, 13-15 September 1978.

[Pirotte 1977]
Pirotte, A., *The Entity - Property - Association Model: An Information-Oriented Database Model*, Technical Report, M.B.L.E. Research Laboratory, Brussels, Belgium, 1977.

[Roussopoulos 1977]
Roussopoulos, N., "ADD: Algebraic Data Definition", *Proceedings of Sixth Texas Conference on Computing Systems*, Austin TX, 14-15 November 1977.

[Schmid+Swenson 1975]
Schmid, H. A. and J. R. Swenson, "On the Semantics of the Relational Data Model", *Proceedings of ACM SIGMOD International Conference on the Management of Data*, San Jose CA, 14-16 May 1975.

[Senko 1975]
Senko, M. E., "Information Systems: Records, Relations, Sets, Entities, and Things", *Information Systems*, Volume 1, Number 1, Pages 3-14, 1975.

[Senko 1977]
Senko, M. E., "Conceptual Schemas, Abstract Data Structures, Enterprise Descriptions", *Proceedings of ACM International Computing Symposium*, Belgium, April 1977.

[Shipman 1980]
Shipman, D., "The Functional Data Model and the Data Language DAPLEX", to appear in *ACM Transactions on Database Systems*, 1980.

[Smith+Smith 1977a]
Smith, J. M. and D. C. P. Smith, "Database Abstractions: Aggregation", *Communications of the ACM*, Volume 20, Number 6, Pages 405-413, June 1977.

[Smith+Smith 1977b]
Smith, J. M. and D. C. P. Smith, "Database Abstractions: Aggregation and Generalization", *ACM Transactions on Database Systems*, Volume 2, Number 2, Pages 105-133, June 1977.

[Smith+Smith 1978]
Smith, J. M. and D. C. P. Smith, "Principles of Conceptual Database Design", *Proceedings of NYU Symposium on Database Design*, New York NY, 18-19 May 1978.

[Smith+Smith 1979]
Smith, J. M. and D. C. P. Smith, *A Database Approach to Software Specification*, Technical Report CCA-79-17, Computer Corporation of America, Cambridge MA, April 1979.

[Solvberg 1979]
Solvberg, A., "A Contribution to the Definition of Concepts for Expressing Users' Information System Requirements", *Proceedings of International Conference on the Entity-Relationship Approach to Systems Analysis and Design*, Los Angeles CA, 10-12 December 1979.

[Stonebraker 1974]
Stonebraker, M. R., *High Level Integrity Assurance in Relational Database Management Systems*, Electronics Research Laboratory Report ERL-M473, University of California, Berkeley CA, 16 August 1974.

[Su+Lo 1979]
Su, S. Y. W. and D. H. Lo, "A Semantic Association Model for Conceptual Database Design", *Proceedings of International Conference on the Entity-Relationship Approach to Systems Analysis and Design*, Los Angeles CA, 10-12 December 1979.

[Taylor+Frank 1976]
Taylor, R. W. and R. L. Frank, "CODASYL Database Management Systems", *Computing Surveys*, Volume 8, Number 1, March 1976.

[Tsichritzis+Lochovsky 1976]
Tsichritzis, D. C. and F. H. Lochovsky, "Hierarchical Database Management: A Survey", *Computing Surveys*, Volume 8, Number 1, March 1976.

Bibliography

[Abrial 1974]
Abrial, J. R., "Data Semantics", in J. Klimbie and K. Koffeman (eds.), *Database Management*, North Holland, 1974.

[ANSI/X3/SPARC 1975]
ANSI/X3/SPARC (Standards Planning and Requirements Committee), "Interim Report from the Study Group on Database Management Systems", *FDT* (Bulletin of ACM SIGMOD), Volume 7, Number 2, 1975.

[Bachman 1977]
Bachman, C. W., "The Role Concept in Data Models", *Proceedings of International Conference on Very Large Databases*, Tokyo, Japan, 6-8 October 1977.

[Biller+Neuhold 1978]
Biller, H. and E. J. Neuhold, "Semantics of Databases: The Semantics of Data Models", *Information Systems*, Volume 3, Number 1, Pages 11-30, 1978.

[Buneman+Frankel 1979]
Buneman, P. and R. E. Frankel, "FQL - A Functional Query Language", *Proceedings of ACM SIGMOD International Conference on the Management of Data*, Boston MA, 30 May - 1 June 1979.

[Chamberlin 1976]
Chamberlin, D. D., "Relational Database Management Systems", *Computing Surveys*, Volume 8, Number 1, March 1976.

[Chang 1975]
Chang, C. L., *A Hyper-Relational Model of Databases*, IBM Research Report RJ1634, San Jose CA, 22 August 1975.

[Chen 1976]
Chen, P. P. S., "The Entity - Relationship Model: Toward a Unified View of Data", *ACM Transactions on Database Systems*, Volume 1, Number 1, Pages 9-36, March 1976.

[Chen 1978]
Chen, P. P. S., *The Entity-Relationship Approach to Logical Database Design*, Monograph Number 6, QED Information Sciences, Wellesley MA, 1978.

[Codasyl 1971]
Codasyl Committee on Data System Languages, *Codasyl Database Task Group Report*, ACM, New York NY, 1971.

[Codd 1970]
Codd, E. F., "A Relational Model for Large Shared Data Banks", *Communications of the ACM*, Volume 13, Number 6, June 1970.

[Codd 1971]
Codd, E. F., "Further Normalization of the Database Relational Model", *Database Systems* (editor Rustin, R.), Prentice Hall, 1971.

[Codd 1979]
Codd, E. F., "Extending the Database Relational Model", *ACM Transactions on Database Systems*, Volume 4, Number 4, December 1979.

[Eswaran+Chamberlin 1975]
Eswaran, K. P. and D. D. Chamberlin, "Functional Specifications of a Subsystem for Database Integrity", *Proceedings of International Conference on Very Large Databases*, Framingham MA, 22-24 September 1975.

[Hammer 1976]
Hammer, M., "Abstraction in Databases", *Proceedings of ACM SIGMOD-SIGPLAN Workshop on Data Abstraction*, Salt Lake City UT, 1976.

[Hammer+Berkowitz 1980]
Hammer, M. and B. Berkowitz, "DIAL: A Programming Language for Data-Intensive Applications", *Proceedings of the ACM-SIGMOD International Conference on the Management of Data*, Los Angeles CA, May 1980.

[Hammer+McLeod 1976]
Hammer, M. and D. McLeod, "A Framework for Database Semantic Integrity", *Proceedings of Second International Conference on Software Engineering*, San Francisco CA, 13-15 October 1976.

[Hammer+McLeod 1978]
Hammer, M. and D. McLeod, "The Semantic Data Model: A Modelling Mechanism for Database Applications", *Proceedings of ACM SIGMOD International Conference on the Management of Data*, Austin TX, 31 May - 2 June 1978.

[Hammer+McLeod 1979]
Hammer, M. and D. McLeod, "On the Architecture of Database Management Systems", *Infotech State-of-the-Art Report on Data Design*, 1979.

[Hammer+McLeod 1980]
Hammer, M. and D. McLeod, "Database Description with SDM: A Semantic Database Model", to appear in *ACM Transactions on Database Systems*, 1980.

[Kent 1978]
Kent, W., *Data and Reality*, North Holland, 1978.

[Kent 1979]
Kent, W., "Limitations of Record-Based Information Models", *ACM Transactions on Database Systems*, Volume 4, Number 1, Pages 107-131, March 1979.

[Lee+Gerritsen 1978]
Lee, R. M. and R. Gerritsen, "Extended Semantics for Generalization Hierarchies", *Proceedings of ACM SIGMOD International Conference on the Management of Data*, Austin TX, 31 May - 2 June 1978.

[Wiederhold+El-Masri 1979]
Wiederhold, G. and R. El-Masri, "Structural Model for Database Design", *Proceedings of International Conference on the Entity-Relationship Approach to Systems Analysis and Design*, Los Angeles CA, 10-12 December 1979.

[Wong+Mylopoulos 1977]
Wong, H. K. T. and J. Mylopoulos, "Two Views of Data Semantics: A Survey of Data Models in Artificial Intelligence and Database Management", *Infor*, Volume 15, Number 3, Pages 344-382, October 1977.