Heterogeneous Databases and High Level Abstraction

Randy H. Katz[1]
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720

1. Introduction

A heterogeneous database management system combines multiple dissimilar models of data within a single integrated system. The objective is to allow a user to access data independently of how it is actually organized. For example, a user may access a database as though it were stored relationally (i.e., in tables) [CODD70], even though it is actually stored as a CODASYL/DBTG or network database [CODA71]. In addition, different subpieces of the database may be organized under different data models. The heterogeneous database system must presnt these to the user as an integrated whole. The user's model of his data data may be different from any of the models chosen to implement it.

Rather than construct a new database system from scratch, we are interested in constructing a heterogeneous system out of existing systems. The key difficulties with this approach are: (1) the formulation of database design methods that are applicable to a variety of different data models, and (2) the development of techniques to translate programs and data between dissimilar data models. In this paper, we briefly describe how high level abstraction has been applied to these problems. The use of abstraction in database systems is related to the application of abstraction techniques in programming languages and artificial intelligence research.

2. Logical Database Design [WONG79]

A single semantic description of the user's data is needed in a heterogeneous environment. Our approach to logical design is to map an abstract specification of the database into a logical schema for any of the underlying data models.

Our model of the real world consists of interrelated objects of interest. These are called "entities" and their interrelationships are called "relationships" [CHEN77]. One of the shortcomings of today's data models is their inability to conveniently express knowledge about the real world. For example, the relational model is defined in terms of subsets of Cartesian products over sets of values, while the CODASYL model is based on record types and their interconnections. Researchers in artificial intelligence, on the other hand, have developed rich models for encoding real world knowledge. Neither the relational nor the CODASYL model provide semantic primitives that aid in describing the real world, and thus do not provide the appropriate framework for logical design.

Consider a database whish stores information about suppliers and the parts that they supply. Suppliers and Parts are entity sets, and the knowledge that a supplier supplies a particular part is an instance of a relationship between two entities. Entities (and relationships) can be further described by their properties, such as a supplier's name and location, a part's name and color, or the number of a particular part supplied by a particular supplier.

Relationships can be used to model certain semantic integrity constraints imposed by the real world. For example, a binary relationship can be an identity (1:1), a function (n:1), or many-to-many (n:m). Further, it can represent compulsory membership. If an employee is only of interest as long as he works for some department, then he must work for some department to be representedin the database. If his department is deleted, then he should be deleted as well. This kind of relationship is called an "existence dependency."

A database is specified in terms of entities and their relationships and properties. Rules are formulated to map the specification into structures supported by the underlying models, e.g., tuples in the relational model or record types in the CODASYL model. The correct behavior of objects under update must be preserved by these rules. Semantic objects (e.g., an individual supplier) must be updated atomically. A record instance or relational tuple is chosen to represent an object to provide this atomicity. In addition, update side-effects must be controlled. Deletion of an object must be propagated to all dependent objects, and no others. This can be supported structurally

---

[1] Current Address: BBN Information Management, 50 Moulton St., Cambridge, MA 02238

in some models, but must be supported procedurally in others.

Similar work has been purseued by researchers in abstract data types. In the above, we introduced an abstract model of the world which is supported by primitive data organizations. ADTs build complex data structures (e.g., queues, stacks, etc.) on top of primitive programming language constructs. The validation of the correctness of the design mapping rules is closely related to the research being pursued on verifying the correctness of an implementation of an abstract data type.

If two schemas in different data models have been derived from the same specification, then translation between them is possible. Rules are formulated to map between the different representations of the same semantic object within target schemas of dissimilar data models. Stronger, more rigorous definitions of equivalence may be needed, because not all of the semantics can be structurally represented by the data models. Some semantics may be represented procedurally within users' programs. For example, the relational model does not provide explicit constructs to support existence dependencies, while the CODASYL model does (Mandatory/Automatic set membership). These are supported by the programs that perform the updates.

3. Physical Database Design [KATZ80a]

Physical design is the phase of the design process which is most sensitive to the particular data model and database system. Abstraction techniques are used to make physical database design more system independent.

Storage structure can be described in terms of abstract concepts of data placement and data interconnection. The properties of data interconnection include (1) rapid traversal from one piece of data to another (e.g., from an employee to the department he works in), and (2) rapid traversal from one piece of data to a set of related data (e.g., from a department to all the employees who work in that department). The properties of data placement include (1) placement of data near each other (e.g., employee near department), and (2) placement of related data near each other (e.g. all employees within a given department are near each other). Some of these properties conflict while others reinforce each other. Physical design proceeds by assigning properties to logical access paths (derived from relationships) which insure that the most heavily travelled paths receive the most favorable combinations. This assignment is used as a specification for realizing a physical database schema for a particular model and system.

Similar approaches have been explored in automatic programming. A specification of what a data structure does and usage information about how it is to be accessed are used to choose among alternative implementations of the data structure [GOTL74]. High level abstraction of storage structure properties is needed to allow flexibility in choosing an implementation.

4. Program Translation [KATZ80b]

Methods are needed in a heterogeneous system to map between database queries expressed in the languages supported by the different data models. For example, a relational calculus query must be mapped into a sequence of CODASYL DML if the underlying database is actually represented as a database organized under that model. The mapping often involves more than a simple transliteration of operations. In the above, there is a disparity in the expressive powers of the manipulation languages involved, akin to the differences between assembly language and high level programming languages.

Decompilation is the process of mapping a sequence of procedural statements into a single non-procedural query. The program is analyzed to determine which sequences of procedural operations implement a single semantic access. For example, a sequence of CODASYL Find statements which enu,erate the member records of a particular CODASYL set can be identified as performing a traversal from a particular department (represented by the owner record) to all employees (represented as the member records) who work there. Once the semantic meaning of the operations is understood, these can be mapped into a high level query.

Compilation translates a non-procedural query into a sequence of procedural operations applied to access paths. The approach is similar to the traditional compilation of high level languages into lower level languages. The optimization of compiled queries depends the infor-ation about storage structure support assigned to access paths during the physical database design.

5. Conclusions

Researchers in heterogeneous database management are confronted with difficult problems in attempting to construct such systems. An abstraction approach can be exploited to help solve some of these.

Much can be learned from work in the related areas of artificial intelligence and programming languages. There is a need to raise the semantic level of database models to make them more suitable for sophisticated database design and schema translation. Some of the work done by researchers in formulating systems for representing real world knowledge can be applied here. Mechanisms for separating the abstraction (e.g., the design schema) from the implementation (e.g., the physical schema) can be borrowed from programming language researchers. Work is still needed in integrating pieces of a single database represented in differentdata models. An abstraction oriented view should prove fruitful here as well.

6. Acknowledgement

I wish to acknowledge the support of the Army Research Office, Contract DAAG29-76-G-0245, the Honeywell Corporation, and the I.B.M. Corporation for their predoctoral fellowship. Also I acknowledge the continued support and encouragement of Professor Eugene Wong.

7. References

[CHEN76] Chen, P. P., "The Entity-Relationship Model - Towards a Unified View of Data," A.C.M. Trans. on

Data Base Sys., V 1, N 1, (Mar 76).

[CODA71] CODASYL Data Base Task Group, April 1971 Report, A.C.M., (Apr 71).

[CODD70] Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," Comm. A.C.M., V 13, N 6, (Jun 70).

[GOTL74] Gotlieb, C. C., Tompa, F. W., "Choosing a Storage Schema," Acta Informatica, V 3, pp. 297 - 319, 1974.

[KATZ80a] Katz, R. H., Wong, E., "An Access Path Model for Physical Database Design," A.C.M. SIGMOD Conf., (May 1980).

[KATZ80b] Katz, R. H., Wong, E., "An Access Path Model for Program Decompilation," submitted for publication to A.C.M. Trans. on Data Base Sys.

[WONG79] Wong, E., Katz, R. H., "Logical Design and Schema Conversion for Relational and DBTG Databases," Proc. Intl. Conf. on Entity-Relationship Approach to Systems Analysis and Design, (Dec 79).