DATABASE RESEARCH: SOME COMMENTS ON FUTURE DIRECTIONS

by

W. T. Hardgrave
E. H. Sibley
Department of Information Systems Management
University of Maryland

1.0 The need for a Formalism for Database Analysis

We have only to look at the 1974 Debate on Data Models (1) to see that there is no common basis for communicating ideas about database concepts. Each year we see dozens of articles proposing new data models, new database designs, and new techniques. Each paper describes the innovation, gives a sample database and application and argues that this new approach will solve the stated problem. Usually references are given to previous work, but rarely does an article even attempt to present a rigorous comparison of a new data model with an existing and accepted data model.

The omission of adequate comparisons is not the fault of the authors individually; they have no tools for comparision. The problem is the lack of an accepted formalism for describing and comparing database requirements and database systems. The situation may be remedied (to a large extent) by the selection of a common formalism for describing and documenting data models. Most sciences use mathematics in their discussions, and there have already been suggestions that we use mathematics in data modelling: e.g. J. T. Schwartz (2) says:

> "..... one aspect of the data base situation seems to me not to pose significant difficulties at all. This nonproblem is the choice, at the abstract level, of sufficiently powerful modes of expression. Data base problems are generally quite simple from the logical point of view and easily formulated in set theoretic terms."

As other examples: Codd (3,4) has presented most of his work in terms of set theory; Sibley (5) in the 1974 SIGFIDET (SIGMOD) Workshop uses set theory extensively as a metalanguage for describing data/storage management structures; graph theory is formally described in set theoretic terms. It therefore seems natural to propose mathematical set theory as the formalism for database analysis.

1.1 The CODASYL DBTG Data Model as an example

One data model that has not been described in mathematical terms is the CODASYL DBTG (6) model. Although this model has been proposed as a standard, it is not yet clear whether the "DBTG SET"

is intended to be a mathematical set, a mathematical sequence, or something entirely different; see Hardgrave (7) for details. Briefly, this distinction is important for two reasons: first, sequences may have a "next" operator and sets may not (although sets may have a "give me one" operator), and second, although sets have a natural definition for intersection and other operations, sequences generally do not.

Conceptually, set operations are necessary for processing Boolean query languages or query languages taken from the predicate calculus. Departures from the above constraints can lead to a breakdown of the Boolean algebra and consequently can result in incorrect response to queries. Therefore, a comprehensive comparison of the relational and DBTG models is impossible until we have a formal (e.g. mathematical) description of the DBTG model. Furthermore, although Bachman (8) suggested that the DBTG and Relational models are equal in capability, this hypothesis can only be proved after a formal description of the DBTG model is available - this is not the case now.

1.2 A Problem with Set Theory

As a vehicle for database analysis, set theory has one major pitfall, which will be described after a brief discussion on terminology. The pitfall involves the notion of "data structures" but we prefer the term "set-based structures" to avoid possible confusion. Most people working in the database and file structure area have a feeling for the term, data structure, but the term is seldom adequately defined. We shall not attempt to precisely define either term here. However, the term set-based structure is often used in database systems to denote

- A mathematical set of atoms
- A mathematical tuple (a sequence) of atoms
- A set or tuple that consists of members that are themselves sets or tuples.

The notion of a relation is a good example of a set-based structure.

In classical set theory, the concept of a tuple is ill-defined. In particular, the tuple definition relies on the Kuratowski definition of the ordered pair plus the notion of nesting a tuple (either to the right or left) into ordered pairs. Details of this mathematical "kludge" may be found in Skolem (9). Even if classical set theory were an adequate description language, a major criticism of it is that it could never be implemented with any hope of efficiency. This follows because the added complexity that comes with nesting ordered pairs makes classical set theory very cumbersome and overburdened with seemingly unnecessary structure.

1.3 Extended Set Theory as a Proposed Formalism

Fortunately, Childs (10) recognized this problem with set theory, and has developed an "extended set theory", that serves as a foundation for the straightforward and elegant definition of set-based structures. In particular, extended set theory provides for the definition of the mathematical set and the mathematical tuple independently and at the same level of abstraction. This is in contrast to classical set theory where the tuple is defined in terms of the ordered pair and the ordered pair is defined in terms of the set.

The authors of this note consider extended set theory to be the best formalism now available for database analysis. Having extended set theory as a basis, we may use the tools of classical set theory (e.g. the set and the tuple) without fear of ambiguity.

2.0 Some New Directions Using the Formalism

Having selected a formalism for analysis, the database community may proceed with research in an orderly manner. As a minimum, this should consist of:

- Providing an exact definition for "data model".
- Enumerating existing data models and their associated implementations.
- Classifying database characteristics, applications, and requirements.
- Matching data applications with data models

2.1 What is a "Data Model"?

It is important to develop an exact definition for the general concept of a "data model" as soon as possible. Currently, opinions on the criteria for determining a data model differ. Certainly the logical data structure, the user access operations (i.e., language), the data definition facilities, and the database administrator facilities are all factors. The objective of this note is not to produce a definition for "data model", but rather to emphasize the need for a rigorous definition. As a practitioner in the database area, I am engulfed in the chaos that results from the lack of a generally accepted definition. Certainly DBTG and relational systems are accepted as bonafide data models, but what about IMS (11), SYSTEM 2000 (12), ADABAS (13), MODEL 204 (14) etc.? The trouble with the adoption of so many systems as "data models" is that they are only (exactly) defined by their implementation - and that is constantly changing! It is therefore impossible to publish comprehensive comparisons of such "data models".

If an exact definition for "data model" is established, it will be possible to recognize a bonafide data model. Equally important, it will be possible to determine which systems are implementations of a specific data model.

2.2 Classifying Applications

It is also important to classify database characteristics, applications, and requirements. Hopefully, it is possible to isolate classes of database requirements. For example, some applications require frequent on-line update; others do not. Some applications require a comprehensive query capability. Some databases have many recurring values, while others consist mostly of unique values. Current database requirements span the range of magnitudes from 10^4 to 10^{12} or more characters do these factors suggest different data models?

The point here is that no single data model appears to be a panacea for all applications. Moreover this will probably be true in the foreseeable future. Therefore, we must attempt to match the application to a suitable data model which has the potential to solve the given problem. Only then should the administrator attempt to select an implementation of the data model. Some work (e.g. the CODASYL Systems Committee's Feature Analysis (15) has been attempted in this vein, but this work must be continued, and it requires a structured environment imposed by using a formalism.

2.3 Comparing efficiency among systems

Finally, there is the issue of "efficiency", "performance", and the comparison of different systems. Potential efficiency is certainly one of the most important aspects of data model comparison. However, it is difficult to attempt to compare efficiency among systems that are constructed to solve different problems, and it is fruitless to compare two systems solving different problems. Efficiency can only be addressed after it has been established that comparisons are made between effective implementations of data models addressing the same problem.

3.0 Conclusions

1. It is important to establish a formalism for analysis of database systems. This will provide a common language and a baseline education level that may be taught to students and newcomers.
2. Our preference for a formalism is extended set theory.
3. It is necessary to define exactly what is meant by the term "data model." Producing a generally acceptable definition will be a difficult task that will require participation from a significant portion of the database community; however, progressing without it will be essentially impossible.
4. The data models and data applications must be classified according to their characteristics.
5. Only after these problems have been solved can we return to the business of producing usable database models and designs. Predicting the efficiency and usability of very large database systems before large sums of money are spent on development hinges on such a structured approach to database design.

References:

1. Data Models: Data-Structure-Set versus Relational, Rustin, R., (ed.), ACM, New York, 1975.

2. Schwartz, J. T., "Abstract and Concrete Problems in The Theory of Files," Data Base Systems, Courant Computer Science Symposium 6, Prentice-Hall, Englewood Cliffs, N.J., 1971.

3. Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," CACM, Vol 13, No. 6, June 1970.

4. Codd, E. G., "A Data Base Sublanguage Founded on the Relational Calculus." Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access and Control. ACM, New York, 1971.

5. Sibley, E. H. "On the Equivalences of Data Based Systems," Data Models: Data-Structure-Set Versus Relational, ACM, New York, 1975

6. April 71 Report of Data Base Task Group (DBTG) CODASYL Systems Committee Technical Report, ACM, New York, 1971.

7. Hardgrave, W. T., Set Processing in a Network Environment, ICASE Report 75-7, Hampton, VA, 1974.

8. Bachman, C. W., "The Data-Structure-Set Model", Data Models: Data-Structure-Set versus Relational, ACM, New York, 1975.

9. Skolem, T. "Two Remarks on Set Theory", Math. Scand. 5, 1957.

10. Childs, D. L. "Extended Set Theory: A Formalism for the Design, Implementation and Operation of Information Systems." Volume IV, Current Trends on Programming Methodology, edited by R. T. Yeh, Prentice-Hall, expected publication in early 1977.

11. Information Management System/360 Version 2 General Information Manual. IBM Form No. GH20-0765.

12. System 2000 Reference Manual, MRI Systems Corporation, Austin, Texas, August 1974.

13. ADABAS Reference Manual, Software A. G., Reston, VA., January 1975.

14. Model 204 Reference Manual, Computer Corporation of America, Cambridge, MA, 1975.

15. Feature Analysis of Generalized Data Base Management Systems, CODASYL Systems Committee Technical Report, ACM, New York, 1971.