SETS AS A MODEL FOR DATABASE REPRESENTATION: MUCH ADO ABOUT SOMETHING

E. Henry Beitz, Consultant
2563 Esch Avenue
Ann Arbor, Michigan 48104-6255
(734) 973-0906

A database models real-world objects and ideas. The power and flexibility of set theory are ideally suited to the problems of modelling the real world without altering it. By reviewing some of the basic concepts involved we will explore the possibility of representing large and complex models in the environment of a digital computer, in what this author hopes will prove to be a suggestive and innovative way.

THE CONCEPTUAL DATABASE

Here we will examine the information environment without regard to machine representation. A database may conveniently be thought of as a collection of objects. Each object exists by virtue of its being described as one of the objects comprising the database.

Every object is represented by some subset of its properties, (where property is used as described by Mealy[1]). The use for which the database is being kept will usually dictate which subset of properties is relevant.

Neither the set of objects **O**, nor the set of properties **P**, is our database. In fact the database **DB** is the binary relation from the set of properties **P** to the set of objects **O**. Or more formally –

$$\mathbf{DB} \subset (\mathbf{P} \times \mathbf{O}) = \{\langle p, o\rangle : (p \in \mathbf{P})(o \in \mathbf{O})\}.$$

Figure 1 shows the database **DB** as a subset of a cartesian plane. The collection of intersections marked with an **x** represents the database **DB**. In general each object **o** in the database is represented by a set of properties

$$o = \{ p : p \text{ is a property of } o \}.$$

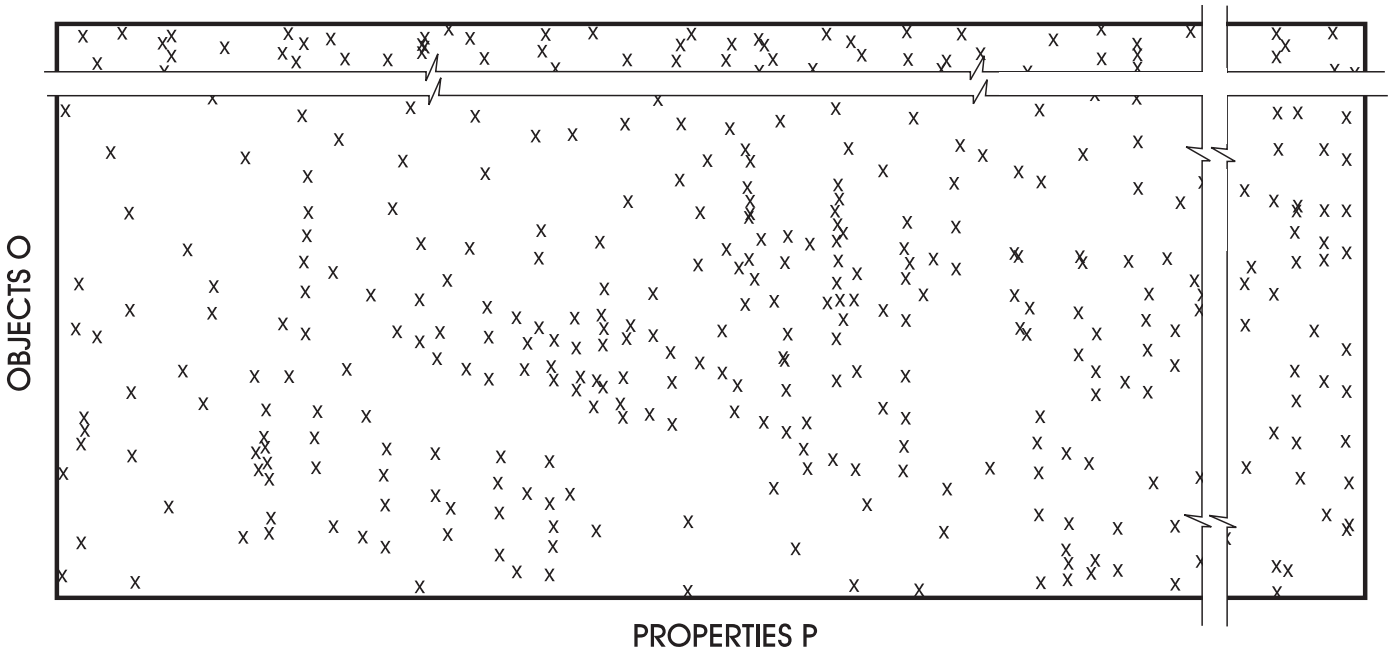

FIGURE 1 - THE CONCEPTUAL DATABASE DB

USING THE DATABASE

Two primary functions can be identified in order to use a database. The first we will call selection. This has to do with choosing a set of objects from the database. Having selected a set of objects the second of our two functions becomes necessary. Let us call it the extraction function. Extraction is the process of obtaining values for a specified set of attributes, from the set of properties that represent each of the objects selected in a prior selection function.

Basically, selection always involves one or more of the properties. The set of objects having a particular property **p**, is given by:

$$DB[\{p\}] = \{o : < p, o > \in DB\}.$$

Regardless of how many properties are involved, this choice is always determined by evaluating a boolean expression whose operands are from the set:

$$\{DB[\{p\}] : p \in P\}.$$

To extract properties from a set of selected objects it is desirable to define another set of sets – the set of sets of properties attributed to specific objects:

$$\{\overline{DB}[\{o\}] : o \in O\}.$$

Since the set of properties **P**, is a binary relation from a set of attributes **A** to a set of values **V**, the pertinent values could be extracted from each of the selected sets of properties by specifying the desired subset of **A**, the attributes. Let us call this subset **RA** – the required attributes.

The resulting set of values, **RV**, for a selected object **o**, would be:

$$RV = (\overline{DB}[\{o\}])[RA]$$

$$= \{v : (< a, v > \in \overline{DB}[\{o\}])(a \in RA)\}$$

We are not concerned with the use to which the set **RV**, of resulting values, is put. The resulting values may be displayed, averaged, divided by 7.694, or whatever. The main purpose here is to be able to access whichever part of the database is needed.

SETS AS MEMBERS OF A SET

There is a need to allow sets of properties to be included as members of a set of properties that represent an object. Because it is proposed that each property be explicitly represented this does not constitute a problem. A simple case is, for example, when the set of properties is a subset of:

$$P|\{a_i\} = \{< a, v > : (a = a_i)(< a, v > \in P)\}$$

that is, every property in the set that is a member of another set, has a common attribute.

A slightly more complex example would be the case where the nested set has both a common attribute and is ordered, or is a tuple. Because of the ambiguities discussed by Childs[2, 3, 4], it will be necessary to explicitly represent the order as an attribute-qualifying ordinal. This may be thought of as follows:

$$<< a, v_d >, < a, v_e >, < a, v_f >>$$
$$= \{< a.2, v_e >, < a.1, v_d >, < a.3, v_f >\}$$

The sets representing objects may only consist of properties, the atomic elements of the scheme. A property may only take a single value in any single context (where the context is defined by the attribute).

RELATIONSHIPS

To make it possible to represent sets as related to other sets, we must look more closely at the notion of a property. The set of attributes needs to include names for relations (and their inverse relations).

An appropriate set of object identifiers must be defined as the set of values that each of these roles, as we will call the names for relations, may take.

Consider a relation **R** from $\mathbf{O_a}$, a subset of the set of objects **O**, to $\mathbf{O_b}$, another subset of **O**. (The two subsets $\mathbf{O_a}$ and $\mathbf{O_b}$ are not necessarily disjoint.) If $\mathbf{x} \in \mathbf{O_a}$ is **R**-related to $\mathbf{y} \in \mathbf{O_b}$, then the set of properties that represent **y** should include $< \mathbf{ar}, \mathbf{v_x} >$, and the set that represents **x** should include $< \mathbf{\overline{ar}}, \mathbf{v_y} >$. ($\mathbf{v_x}$ and $\mathbf{v_y}$ are the object identifiers of **x** and **y** respectively, and **ar** and $\mathbf{\overline{ar}}$ are the roles from $\mathbf{O_a}$ to $\mathbf{O_b}$ and $\mathbf{O_b}$ to $\mathbf{O_a}$ respectively).

The reason for including this seemingly facile description of a relationship is to establish the concept of object identifiers. In the relational model[5, 6] each subset of the partition of the set of objects and every other relation requires a primary key, which essentially serves the same purpose.

The reason for not simply nesting the properties of the related object, suitably qualified, in those representing the object to which it bears the relationship, is that the same object may be the subject of a number of relationships. This implies redundant representation of some objects, which in turn implies multiple updating of these same objects' sets of properties.

Because it is often necessary to process the relationships without immediate regard for the objects that are the subject of the relationship, the relationships may themselves be treated as objects. The proposed model deals with objects represented by sets of properties, and because of the concept of a property, complex relationships may be handled.

A simple binary relation such as the one described above may be represented as a set of objects. Each object that is a member of **R** would look like this:

$$\{< \mathbf{ar}, \mathbf{v_x} >, < \mathbf{\overline{ar}}, \mathbf{v_y} >\}$$

where $\mathbf{v_x}$ and $\mathbf{v_y}$ are object identifiers for some object that is a member of $\mathbf{O_a}$, and some object that is a member of $\mathbf{O_b}$, respectively. The set **R**, of objects representing the relationship, is itself a subset of the set of objects comprising the database. Each object that is a member of **R** will also have an object identifier.

A selection function will suffice to find all the objects **R** that have a property $< \mathbf{ar}, \mathbf{v_x} >$. Having selected a set of relationships from **R** we can invoke the extraction function to obtain the values of the properties having the attribute **ar**, for each of the selected relationship objects. Determining which object or objects in the one subset are related (in a particular role) to which specific object or objects of the other subset, is a simple process.

There is no reason why each instance of an object's inclusion in a relationship would have to be separately represented. The constraints that apply to each of the role-names would determine this. For example, let $\mathbf{O_a}$ be the set of vehicles, $\mathbf{O_b}$ the set of people, and **R** the relationship of **OWNERSHIP**. One person owning two vehicles would be the object –

$$\{<\text{VEHICLE, y}>, <\text{OWNER, x}>, <\text{VEHICLE, z}>\}$$

in **R**, where $\mathbf{(y, z \in O_a)}$ and $\mathbf{(x \in O_b)}$. Two people jointly owning a number of vehicles would also be easy to cope with. Because we are dealing with a set of properties, the order of the properties is not relevant. Of course, this depends on the fact that the role-name is explicitly recorded in each property.

The concept can be extended to cover much more complex relationships:

$$<< \text{PROFESSOR, COURSE, DAY, TIME, ROOM}>, \text{STUDENT, GRADE} >,$$

for example. It is important to realize that the range of a property of a relationship might be identical for more than one of the roles of that

relationship. The role-name is what makes a distinction possible. Take an example from genealogy – **x IS THE PARENT OF y.** Although both have as their range the set of people, the role-name associated with **x** might be **PARENT** while that associated with **y** is **CHILD**. (There are of course people who will never appear in the relationship as a parent, those who will never appear as a child, and even some who won't appear in the relationship at all.)

The original kind of property may also be included among the properties of an object that represents a relationship. This simply means that the sum total of what may be said about the role is that it has a value. (See **DAY, TIME** and **GRADE** in the example above. In an integrated database it is highly likely that there will be much more information about each of the other items!)

THE MACHINE ENVIRONMENT

The model described above was conceived with the characteristics of the modern digital processor constantly in mind. Rapid execution of the two primary functions, selection and extraction, makes it necessary to have two complementary representations of the database. To ensure flexibility and to map each of the two sets of sets to the memory that is accessible in a particular environment, a well-ordered and compact renaming of each set is needed. This new internal name of each set must remain invariant for the duration of its existence.

Two identity functions are desirable; one that uniquely names every object and one that uniquely names every property. The simplest function for either set would be from an index–set of integers to the members of the set. (Each object's index ordinal will also serve as that object's identifier in the relationships described above.)

The importance of freeing the representation of each set of properties and each set of objects from their storage addresses cannot be emphasized sufficiently! Two indices need to be provided. Each may comprise a block of logically contiguous storage addresses and every entry should be uniform in size. This means that there are four distinct parts to the representation of a database: the two indices and the two sets of sets

$$(\forall p \in P)(DB[\{p\}]) \quad \text{and} \quad (\forall o \in O)(\overline{DB}[\{o\}])$$

The indices serve to map the sets of property–names and object–names to storage addresses as depicted in Figure 2.

Each index and its related sets of images represents the entire database. This makes it possible to reproduce either representation from the other. (To insure that at least one of the two representations is intact, it is advisable to store them on physically disjoint sets of mass storage units.)

The high information density could result in a greatly reduced volume of data transfer. At any instant in time the size of both the sets of properties and the sets of objects is finite. It should be possible, therefore, to represent the set of objects having a specific property, by a string of fixed-length object identifiers; the size of each object identifier being a function of the total number of objects in the environment. In a similar manner, each set of properties could be represented as a string of fixed-length property identifiers. It is only at the user interface that it will be necessary to convert these codes to readable strings of graphic symbols.

Because sets of properties and objects are represented as subsets of their index-sets a simple well-ordering of each set is readily available. By virtue of this encoding, operations such as union, intersection and difference of these well-ordered strings are simply realized.

In the case of extremely large sets, particularly of object identifiers of objects having a specific

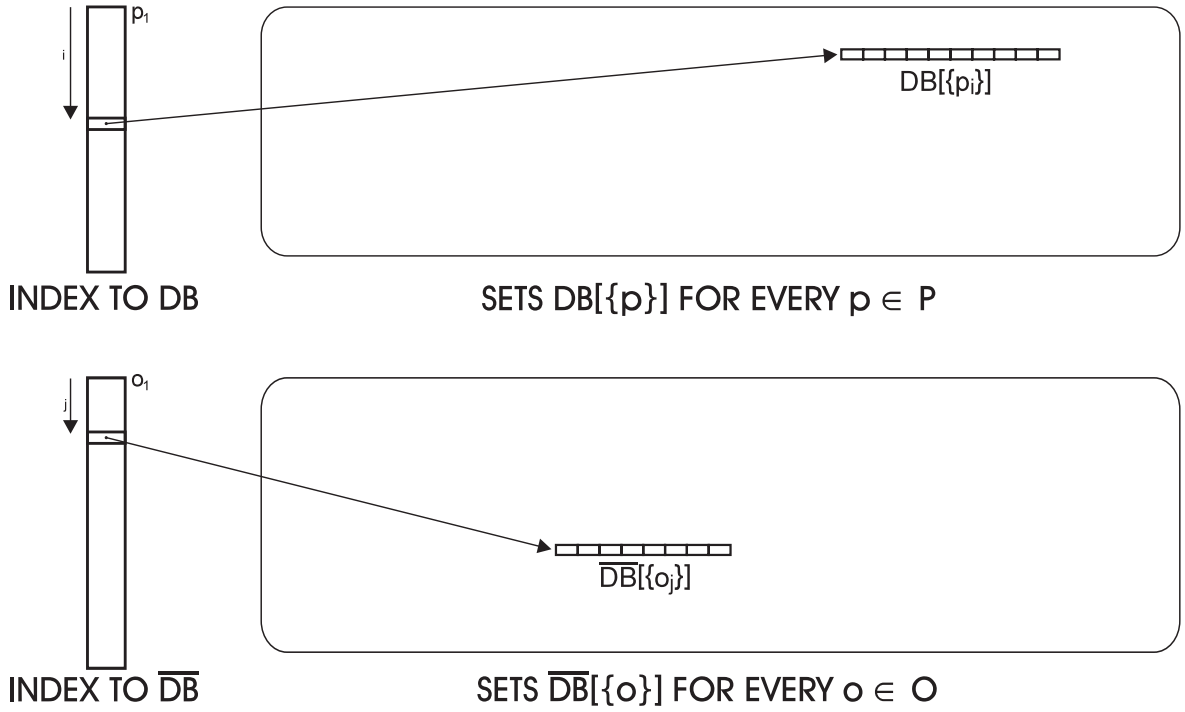

FIGURE 2 - THE TWO REPRESENTATIONS (DB & $\overline{DB}$) OF THE DATABASE

property, numerous different representations are possible[7, 8, 9].

Any property may be used to gain access to the database. Having gained access to the database, the information space may be traversed by way of the relationships. The length of the path from the point of entry to the desired objects and their properties will depend on precisely which relationships are explicitly represented.

Some relationships may be derived from others. For example, **MATERNAL-GRANDFATHER / GRANDCHILD** may be derived from **MOTHER / CHILD** and **FATHER / CHILD**. When the former relationship is heavily used it should be explicitly represented (and will therefore have to be maintained).

Any changes that effect the relationships from which this derived, redundant one is formed, must initiate a procedure to determine whether the derived relationship should be updated. The set of objects representing the new relationship may be abandoned at any moment in time. The only consequence will be that the access path will become longer.

In current practice what is usually called structure is implicitly represented in the arrangement of stored data. In fact, where it is, often defines, what it is. Structural information should be represented explicitly so as not to preclude any view of the environment being modelled.

PARTITIONING THE DATABASE

The environments that we normally model have characteristics that allow us to partition the set of objects **O** in very useful ways. (Partitioning is used in its strict sense, that is, a set of non-empty subsets that are disjoint and whose union is the set of objects **O**.)

How a database is partitioned depends on its contents and the manner in which it is intended to be used. The reasons for partitioning relate directly to the methods of representing the database and to the nature of the medium being used for modelling it. The

basis for partitioning the set of objects would be a common property or perhaps even a set of common properties. The modern digital computer can very rapidly carry out quite complex operations on a small database. What is meant by small depends only on current technology. Sensible partitioning should also make it possible for the digital computer to handle large, complex databases proficiently.

The subsets of the partition are objects in their own right. Each such subset must be described as an object that has those properties which are common to all the database objects belonging to that subset. This means that each of the database objects must have as one of the properties that describe it, a property that indicates which subset of the partition it is a member of. From this one property it may be inferred that the object also has all of the properties of its partition. When large populations are involved this can mean a substantial reduction in the amount of storage required for the database. (But a word of warning: too fine a partition would increase the amount of processing needed to answer even a simple query.)

The set of attributes that have a meaning for every member of a subset of the partition can be specifically enumerated. Further qualification of each of the attributes would also make it possible to control data integrity: the set of values that constitute valid properties of members of the subset in a particular context, may be precisely described; the fact that a specified number of properties from a defined subset must be included in each object's set of properties, may be stated; the case where each member of a subset of the partition must take a unique value in a particular context, can be dealt with; etc..

CONCLUSIONS

Current and projected technology has been considered the prime reason for adopting the point of view presented above. All the constructs of the relational model[5] may be represented using the view described here. Many of the phenomena observed by Dewey[10], Shannon[11], Zipf[12] and Mandelbrot[13] can be exploited in an implementation using this model as its basis. Of the three approaches to information discussed by Kolmogorov[14] the proposed model, approximates the combinatoric one most closely.

I would like to acknowledge the contribution of my colleagues and other mentors in putting this together. In particular I would like to thank Jane Jodeit who helped me sort it all out, and Ho-Nien Liu who encouraged me to write this paper.

CITED REFERENCES

1. **Mealy, G. H.**, Another look at data.
   AFIPS Conference Proceedings, FJCC (1967), pp 525 - 534

2. **Childs, D. L.**, Description of a set-theoretic data structure.
   AFIPS Conference Proceedings, FJCC (1968), pp 557 - 564

3. **Childs, D. L.**, Feasibility of a Set-Theoretic Data Structure – a general structure based on a reconstituted definition of relation.
   Information Processing 68, North Holland, Amsterdam (1969)

4. **Childs, D. L.**, Extended set theory.
   STIS Corporation, Ann Arbor, Michigan (1974), (distributed by author).

5. **Codd, E. F.**, A Relational Model of Data for Large Shared Data Banks.
   Communications ACM 13, 6 (June 1970), pp 377 - 387.

6. **Date, C. J.**, An Introduction to Database Systems.
   Addison Wesley (1975)

7. **Bloom, B. H.**, Some Techniques and Trade-offs Affecting Large Data Base Retrieval Times.
   Proc. 24th National Conf. ACM (1969), pp 83 - 95.

8. **Hardgrave, W. T.**, The Prospects for Large Capacity Set Support Systems Imbedded within Generalized Data Management Systems.
   Presented at the International Computing Symposium (1973), Davos, Switzerland.

9. **King D. R.**, The Binary Vector as the Basis of an Inverted Index File.
   Journal of Library Automation, 7, 4 (December 1974), pp 307 - 314

10. **Dewey G.**, Relativ Frequency of English Speech Sounds.
    Harvard University Press (1923)

11. **Shannon, C. E.**, Prediction and Entropy of Printed English.
    Bell System Tech. J. 30 (1951), pp 54 - 58

12. **Zipf G. K.**, Human Behavior and the Principle of Least Effort.
    Addison Wesley (1949)

13. **Mandelbrot, B.**, Contribution à la théorie mathématique des jeux de communication.
    Publ. de l'Inst. de Statistique de l'Univ. de Paris, Vol. 2, fasc. 1, 2 (1953), pp 80 - 102

14. **Kolmogorov A. N.**, Three Approaches to the Quantitative Definition of Information.
    International J. of Comp. Math. Vol. 2, (1968), pp 157 - 168