AN EXTENSION OF RELATIONAL DATABASE MODEL TO PROBABILISTIC DATA

An approach based on generalized arrays.

Renzo Beltrame

CNUCE - Inst. of CNR
36, Via S. Maria
56100 PISA - Italy
(050) 593 237 - Telex 506371

Abstract: In many applications, for example in representing data concerning historical "objects", alternatives are the rule; so we are led to a probabilistic scheme in their representation. A deterministic approach, which underlies actual DBMS types, leads to very poor results, and, in certain cases, to misleading ones too. An extension of relational model is proposed to take into account these aspects, and the effects of a non-deterministic approach on operations concerning both relations and database are examined. A formal description of this extension is presented with reference to the internal representation of data. The effects on the queries are outlined. A representation as a general array is discussed which preserves the typical properties of the relational model with deterministic data, and the form of the operations.

1.0 INTRODUCTION.

When data concerning historical "objects" are represented, uncertainty and alternatives are the rule. If a deterministic approach is adopted, which underlies actual DBMS types, very poor, and, in some cases, misleading results are obtained.

If we consider, for instance, the history of fine arts, the attribution of a painting to a certain author and the period of its execution are often discussed, and different solutions are proposed by various historians. When we force a deterministic approach to data representation we impose a choice either of a single datum or of more but equiprobable data.

In both cases we have two side-effects:

- a loss of information which may occur either because possible information is not introduced in the database, or because the weight of the registered information is different from the expected one;
- a distortion in the information content because we force two information to be treated in the same way despite their different probabilities: so the basic principle does not hold which ensures that data retrieved according to a certain characteristic are homogeneous when retrieving characteristic is used as comparison criterion.

As an example we can look at the following cases referring to Raffaello's works.

The table "Lo sposalizio della Vergine" at Brera Pinacoteque in Milan is signed and dated 1503. The so called portrait of Elisabetta Gonzaga at the Uffizi Gallery in Florence was attributed to: Raffaello, Caroto, and Francia environment, and with different degrees of certainty. Furthermore the identification as the portrait of Elisabetta Gonzaga is discussed too. For the "Scuola di Atene" fresco in Vatican Stanze, there is a discussion about a possible participation of Bramante in architectures perspective (see "Came62" and "Brus70").

We remark that the last case, using a relational model of database, shows an alternative between simple and non-simple domains. Collaboration, in fact, has natural description as a particular relation.

More complicated situations arise for the Loggia of Psiche or Vatican Logge decors.

Apart from technical discussions among fine arts historians about the data examplifyed above, the introduction of a probability information in database relational model opens the model itself in two main directions; they can be informally pointed out as follows.

In a n-tuple of a relation R:

1. more "objects" can be connected to one through the same conceptual relation with different probabilities, because we can have a measure space as an element instead of a single item;
2. two "objects" can be connected together through different conceptual relations with different probabilities; for example we could have the same person as the unique author of a painting with a certain probability value and, with another probability value, as the leader in a collaboration with other painters.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

1984 ACM 0-89791-137-7/84/006/0017 $00.75

tuple	j-component	its probability
T1	a	1
T2	a	p1
	b	p2
	c	p3

Fig. 1. Statement of the problem.

We remark that the same structure can support other types of applications, for example when we catalogue books or papers by subject we can add to each classification item a number describing the degree of pertinence of the classified object to the subject considered.

Clearly the degree of pertinence does not have a probability structure but rather a fuzzy set one. Nevertheless the treatment of the two problems may become very similar, althogh different approaches were proposed (see for example "Salt83").

2.0 FIRST EXTENSION OF THE RELATIONAL MODEL

As a first step in this work we shall discuss a relational model extension considering situation in which domains are all simple. So we shall consider a relational model in its first normal form: with relations domains all simple. We can summarize the problems in the table of Fig. 1, which covers the first two situations discussed in the introduction.

These assumptions are justified by a discussion of extensions oriented to the internal representation of a database rather than to the conceptual schema. For the same reason we do not consider here the effects of a probabilistic approach on the static and dynamic constraints representation, on rules, and on other related topics.

2.1 FORMAL DESCRIPTION

To describe a relation having elements to which a probability value is added, we introduce in the relational model the following definitions which are sligthly different from the classical ones:

1. we start from a certain number M of **domains of basic type** Bi distinct but not necessarily disjoint; the elements of these domains are the domain values of the usual relational model; we call these elements objects;
2. for each domain of basic type Bi we define a space of probabilities Ai(Bi,Bi), where Bi is a sigma-ring of subsets of Bi. Each space Ai is a space of positive measures defined on (Bi,Bi), such that either fi(Bi)=1 or fi(Bi)=0 for every fi belonging to Ai. Clearly if fi(Bi)=0 fi is identically null;
3. **a simple domain** Sj is defined by a space Aj;
4. **a value belonging to a simple domain** Sj is defined by a function fi belonging to Aj, where Aj is the space of probability defining Sj.

The definitions of the universe U as the Cartesian product S1xS2x.....xSn of n domains, of relation as subset of U in which we shall refer to Sj as the jth domain of R, and of relationship as an equivalence class of those relations which are equivalent under permutation of domains (see "Codd70"), remain unchanged.

We remark that a simple domain in the extended model is quite different from a non simple domain in the deterministic one, because a measure space has a field structure.

We remark too that the new definition of simple domain value allows to treat with the same formalism null values and nonnull ones.

2.2 RELATIONAL ALGEBRA EXTENSIONS.

The new definitions affect in a different way basic operations of relational algebra introduced for non-inferential data systems. For these basic operation we refer to Ullman's definitions given in chapter 4 of "Ullm79."

Permutation: Permutation is unaffected by the new definitions because it requires only a permutation of components in the tuples that represent relation.

Projection: Projection is affected by new definitions when we remove any duplication in tuples resulting from selection of certain domains in a relation. Hence we must

object	probability	result
different	equal or different	false
equal	equal	true
equal	different	false

Fig. 2. Comparison table of values

redefine equality between values belonging to the same domain, but in different tuples, taking into account their probabilities.

In projection equality is rather an identity, or, if we prefer, a sort of someness. Then the table of values of Fig. 2 holds.

We must consider not equal two tuples having the same object in a certain domain, but different probability values associated to it, and so we will maintain both. They can be processed later in different ways depending on differences in the queries. We will discuss this aspect of the extension in "2.3.2 Query."

In every case equality between probability values is an event of low probability, and we must introduce a **comparison tolerance.**

Formally, let fi and fj be two values of the same domain, because fi and fj are two functions, equality holds iff the distance between the two functions is smaller than or equal to a real number **ct** which acts as a comparison tolerance, and this number must be supplied when projection is performed. One of the ways to define the equality is to assume as a distance between the two functions fi and fj the sup $|fi(E)-fj(E)|$, where the supremum is computed over the sets E belonging to **Bi**; but other definitions can be used depending on the particular applications.

As we shall see in discussing queries processing (see "2.3.2 Query"), the comparison tolerance can become a very critical parameter.

Union: We can consider only two relations, which must have the same arity, because the union has a semigroup structure. Union is affected by new definitions like projection when we remove any duplication in tuples which result from the catenation of the tuples of the two relations involved in union.

Difference: We can consider only two relations, which must have the same arity, because the difference has a semigroup structure. Difference is affected by new definitions like projection when we compare two tuples to choose the tuples that are present in the first relation but not in the second one.

Cartesian product: We can consider only two relations, because the cartesian product has a semigroup structure. Cartesian product is affected by new definitions like projection when we remove any duplication in tuples which result from the catenation of the domains of the tuples of the two relations involved in cartesian product.

Selection: Selection gives the set of tuples in a relation R such that a given relation holds among the values of stated domains (see "Ullm79" at p. 106). So we have to redefine relational operations on objects belonging to different domains and in different tuples taking into account their probabilities.

We consider the simplest form of selection - involving two domains or a domain and a value as constant - because selection has a semigroup structure.

Let fi belonging to Ai and gj belonging to Aj be the two components of tuples involved as operand in relational operation required by selection, and (fi x gj) their product measure. We are interested in (fi x gj) measure of the set (E x E′) of pairs (s,s′), with s belonging to Bi and s′ belonging to Bj, for which the considered relation - equal, less, greater, etc. - holds; this measure gives the probability that the relation holds. E belongs to **Bi** and E′ belongs to **Bj**, so $(fi \times gj)(E \times E') = fi(E)gj(E')$.

For each relational operation involved in selection a probability value cl - **confidence level** - must be supplied, and the result will be true if the product measure is greater than or equal to the supplied value; that is if $fi(E)\ gj(E') >= cl$ holds.

We remark that a "constant" involved as operand in selection implies either $fi(E)=1$ or $gj(E')=1$. Furthermore in many cases we have $E=\{s\}$ and/or $E'=\{s'\}$, that is they can contain only one point.

Join: Join, in its general form too (see "Ullm79" at pp. 108-9), has a semigroup

structure, and so we can consider its simplest form: the join of two relations R and S is given by those tuples in Cartesian product RxS such that a given relation holds between the i-th component of R and the j-th component of S. We must so redefine relational operations on values belonging to different domains and in different tuples taking into account their probabilities; but the problem is the same we considered in selection; cartesian product also does not need any extension different from those considered above. So no new extension is required.

Other operations on relations can be expressed as sequences of the operations described above.

2.3 OPERATIONS ON DATABASE

We consider here the effects of this extension on the operations performed on database such as query, insertion, deletion, and update of information.

2.3.1 Insertion, deletion, and updating.

If we insert or delete a tuple we have no difference. When insertion, deletion, or update concern the components of a tuple they are strongly affected by the new definitions introduced, because every component of a tuple is now a set function, and so we are led to typical problems of representing and updating set functions preserving normalization properties required by the probability concept.

Any maintenance operation requires in fact insertion, deletion, or substitution of an element belonging to a space of measure (see "2.1 Formal description") with another belonging to the same space. And at this point we have various possibilities.

In the following we will propose an approach based on general arrays which is the natural extension of the deterministic one (see "3.0 A representation as general array"); it is particularly suitable when the basic domains Bi are finite, discrete sets.

When the basic domains are one or more bounded intervals of R1, other approaches may be more suitable.

Furthermore they may be more attractive, even formally, in describing constraints. Consider, in fact, that in manipulating generalized functions instead of presences or absences of "objects", we are led to the domain of mathematical analysis rather than that of the logic.

2.3.2 Query

The query philosophy of an information system in which a certain probability value is added to every atom of information is quite different from that of a deterministic one.

We have two kinds of differences. The first concerns relational algebraic operations that the query involves. The second concerns the type of result we plan to obtain from the query.

As discussed above (see "2.2 Relational algebra extensions.") selections and joins require that a confidence level must be added; the comparison tolerance to be used in identity check between tuples can be supplied either as a database characteristic, or as a query one, the choice depending on the particular application.

The second kind of differences leads to more subtle and spread-out consequences, because of the large spectrum of possibilities offered by a probabilistic approach.

We give here two examples to outline the type of problems that can arise. As we will see, post-processing of results obtained by relational algebraic operations is required, and this post-processing is strictly related to probability calculus. So the two aspects will be treated separately in processing a query.

Let us suppose we will obtain from our database the names of the authors who produced a certain type of works, for example a certain type of objects, in a certain place, let's say Florence, and in a prefixed period in the past.

A confidence level must be supplied for each condition. It states the minimum level of probability at which we will accept the condition as true (see "2.2 Relational algebra extensions."). Clearly these values depend strongly on the problem that originated the query.

Fig. 3.

Intermediate result of the query

obj_type	period	place	author
O o1	P p1	F f1	A1 a1 A2 a2 A3 a3
O o2	P p2	F f2	A2 b1 A4 b2
....	....	....	.....

author	probability
A1	o1p1f1a1
A2	max(o1p1f1a2,o2p2f2b1,..)
A3	o1p1f1a3
A4	o2p2f2b2
....	....

Fig. 4. Final answer to the first query.

Applying the relational algebraic operations involved by the query, selections and projection, we will obtain a table as that of Fig. 3.

In this table o1 is the probability that the first object in the list was of the required type; p1 the probability to be produced in the required period of time; f1 the probability to be produced in Florence; A1,A2,A3, the attributions with the related probabilities a1,a2,a3; and so on for the other items of the list.

A post-processing of the table of Fig. 3 would give a result as that illustrated in Fig. 4, which can be considered a good way to answer the query, because it gives, for each author, the maximum value of the probability that the query conditions were all fulfilled.

Let us now suppose we would obtain a comparative evaluation of the quantity of work the various authors made, instead of their name only, all the other conditions being unchanged.

Let us perform the same relational algebraic operations and we will obtain a table like that illustrated in Fig. 3. As the first post-processing step, we calculate the probabilities that a certain number of the stated objects were produced. Then we compute in each case the mean value of the number of objects that can be attributed to the different authors.

The final answer could assume the form given in Fig. 5; where pn,...,p0 are the probabilities to have respectively Nmax,...,0 objects which fulfill the required conditions, and m1,...,m4 are the mean number of the objects that can be attributed to each author. Clearly this latter set of values is different in the various cases.

A distribution could be better than a mean value as the final result of authors work, depending on the application that originated the query.

It is beyond the scope of this paper to examine in detail the problems that an extension like that discussed above poses to the query philosophy. The fact we pointed out is the possibility to separate the query manipulation in two steps: the first strictly related to the extended relational algebraic operations, the second to the probability calculus.

objects number	authors work
Nmax pn	A1 m1 A2 m2 A3 m3 A4 m4
....	....
1 p1	A1 m1 A2 m2 A3 m3 A4 m4
0 p0	

Fig. 5. Final answer to the second query.

3.0 A REPRESENTATION AS GENERAL ARRAY

In many applications either Bi are finite countable sets, or they can be defined in this way without loss of information.

Our probabilities, in these cases, are defined on fields of subsets, rather than on sigma-field, and so we can continue to use an array representation of relations, respecting also usual implementation constraints about finiteness of indices.

Arrays become general arrays when we introduce extensions discussed above (see "2.0 First extension of the relational model"), but the structural properties remain unchanged.

We recall that a general array is an ordered collection of elements, which can be arrays themselves. The number of dimensions, or axes, of an array is called its rank, or valence. Each index axis has a length, which is the number of indices on the axis. An array containing one element is called a single. The collection of the lengths of all the dimensions of an array is called its shape. Lengths and indices may be any countable ordinal number; restriction to finite ordinal numbers is an implementation constraint. The elements of a general array can belong to the basic types and/or can be general arrays themselves. An array whose elements are all of the basic type is called a **simple array** (see "More73").

In the following we shall consider that both probability values and objects in the domains Bi were of basic type in general arrays representation. This assumption simplifies the discussion avoiding particulars which depend on the implementation of a programming language based on general arrays.

A general matrix which represents an n-ary relation R having probabilized data can be graphically indicated as in Fig. 6. It maintains the usual properties of this representation where only deterministic data are involved (see, for historical reasons, "Codd70"). In fact at the outermost level:

- each row represents an n-tuple of R;
- the ordering of the rows is immaterial;
- all rows are distinct;
- the ordering of columns is significant - it corresponds to the ordering of the domains on which R is defined;
- the significance of each column could be partially conveyed by labelling it with the name of the corresponding domain.

Considering a probabilistic information involving only discrete sets, if we look at the internal structure of each element of the general matrix, we can find:

- either a vector whose first element is a traditional value of a basic domain, and the second one its probability; in this case always 1;
- or a matrix of several rows where each row is a vector of two elements; this matrix describes the function fj that represents the value of the j-th component.

The elements will be simple or not depending on the representation of all the components as elements of the basic type in the implementation of the programming language that supports generalized arrays.

The solution here outlined records only the objects - the traditional domain values - to which nonzero probability is associated, and the related probability value. That is we restrict the representation to the support of fj. This solution corresponds, formally, to one of the simplest form to obtain a measure: starting from a real valued, non negative function of the points of the finite set Bi.

Any maintenance operation concerns both kinds of information: so, whenever we insert, delete, or update a value, the following operations must be done:

- the elements of the general matrix representing the n-ary relation must be retrieved;
- rows of each element, that is the basic domain value and its probability, must be inserted, deleted, or updated in the retrieved matrix;
- the measure values must be renormalized, that is an update is required of all the measure values in the retrieved matrix.

When we process the elements of the general matrix, operations on relations require an extension of the usual relational algebraic operations to take into account probability values in the form described above. But, in both cases at the outermost level, the usual approach of relational model with deterministic data is preserved, and the form of DBMS procedures too.

In particular, the result of applying the relational algebraic operations involved by the queries discussed above (see Fig. 3) will give a general matrix of the form presented in Fig. 7.

In this paper we limit ourselves to a graphical representation of general arrays which result from the approach discussed above, and we avoid presenting the form of the functions which process the elements of the general matrix that represents an n-ary relation.

There are a certain number of implementations of general arrays as extensions of the APL language, but they differ deeply in philosophy and require very different paths to obtain the same result. The choice of a standard set of operators and functions to manipulate general arrays is in fact a matter of study and debate in the APL community. For these reasons we have preferred to give only a graphical presentation.

Figure 6.
n-ary relation represented as general matrix

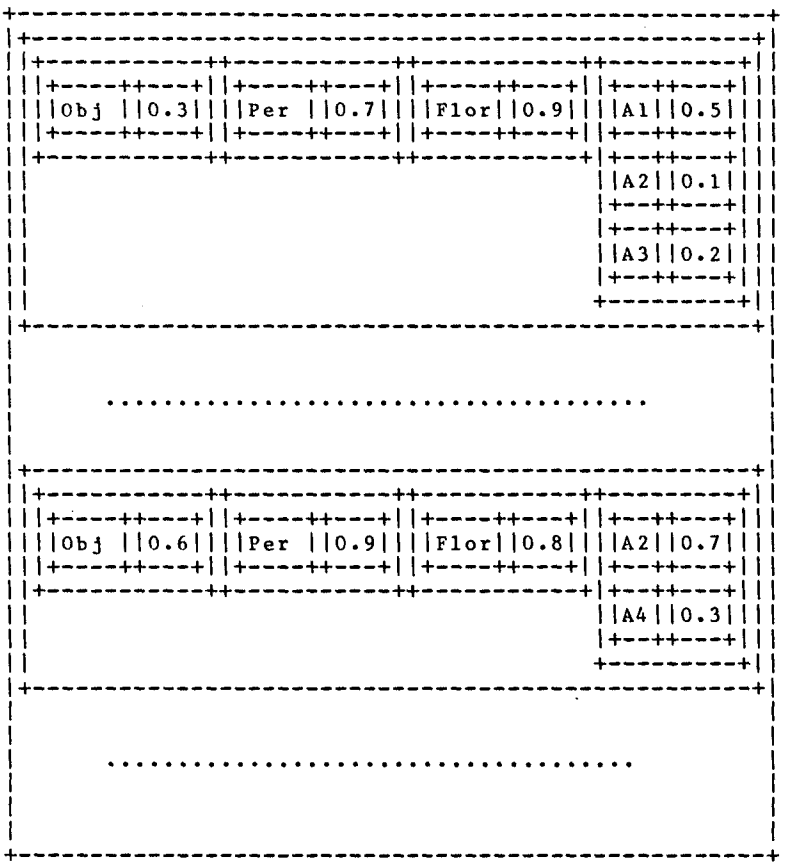
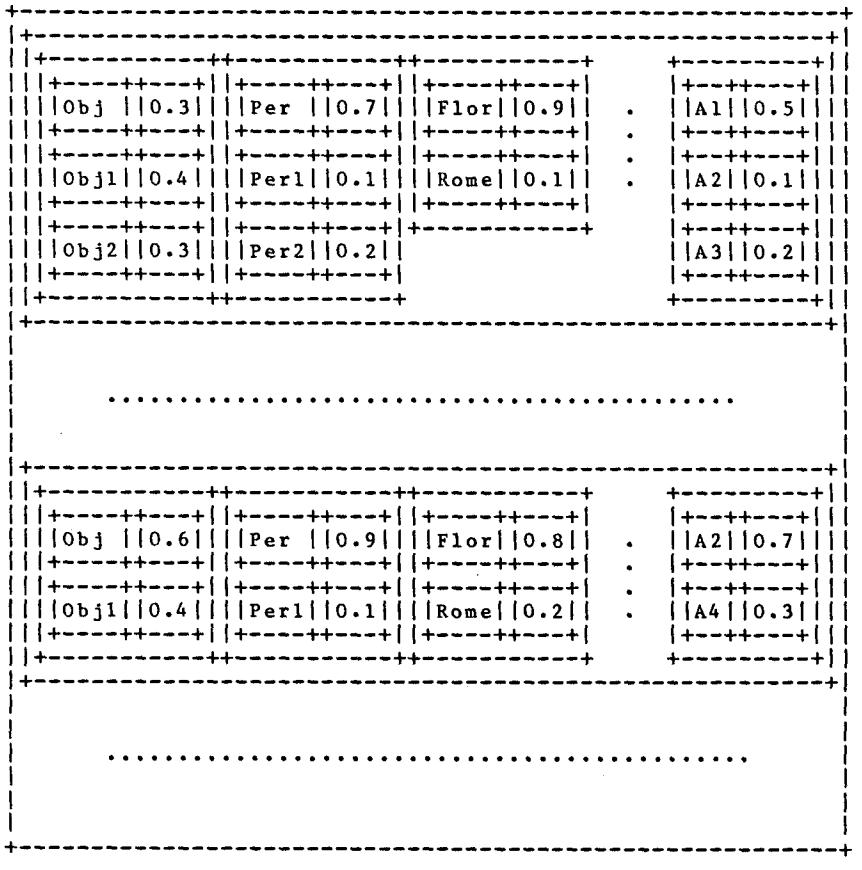


Figure 7.
Query intermediate result



REFERENCES

[Brus70] L. BRUSCHI, **Bramante**, Electa, Milano, 1970.

[Came62] E. CAMESASCA, **La pittura di Raffaello**, Rizzoli, Milano, 1962.

[Codd70] E.F.CODD, "A Relational Model of Data for Large Shared Data Banks", **Comm. ACM, 13**, .6 (1970), pp. 377-387.

[Ghan73] ?. GHANDOUR, J. MEZEI, "General Arrays, Operators and Functions", **IBM J. Res. Develop. 17**, (Jul. 1973), pp. 335-352.

[More73] T. MORE Jr., "Axioms and Theorems for a Theory of Arrays", **IBM J. Res. Develop. 17**, (Mar. 1973), pp. 135-175.

[Halm74] P.R. HALMOS, **Measure Theory**, Springer-Verlag, New York, 1974.

[Salt83] G. SALTON, E.A. FOX, H. WU, "Extended Boolean Information Retrieval", **Comm. ACM, 26**, 12 (1983), pp.1022-1036.

[Ullm79] J.D. ULLMAN, **Principles of Database System**, Computer Science Press, 1979.