Control and Cybernetics

vol. **26** (1997) No. 2

Architectural support issues for fuzzy databases

by

Frederick E. Petry

Center for Intelligent and Knowledge-Based Systems,
Department of Electrical Engineering and Computer Science,
Tulane University, New Orleans, LA 70118, USA

Abstract: In this paper, approaches to support general purpose fuzzy databases are considered. First, issues relative to the need for fuzzy databases are reviewed. This gives the perspective that hardware support is necessary to provide the performance needed for commercial systems. In particular, the concepts of database machines and the Bellcore Datacycle system is discussed. Then, preliminary steps to design of a backend processor for a similarity based fuzzy database is described.

1. Introduction

To begin the discussion of issues of architectural support for fuzzy databases, we must address the scope of acceptance and requirements for commercial fuzzy databases. The aspects of commercialization relevant to market issues that will dictate the future potential for fuzzy databases must be considered as these will determine the future directions of fuzzy databases and development of specialized architectures.

The database field is currently extending its paradigm from a mostly relational approach towards an object-oriented direction. It is not clear yet what forms these object-oriented extensions will take. Indeed there is little agreement upon which features are needed since this approach did not spring from a single source as the relational model did from Codd's research. Likewise, the scope of database capabilities is expanding to encompass multimedia, text, pictures, etc., and this is of significance in supporting complex applications such as CAD/CAM design systems and geographical information systems (GIS). Such a range of database approaches represents a part of a broad spectrum of information systems encompassing additionally the processing of large text-type files conventionally construed as the realm of information storage and retrieval systems. Various heuristic techniques, such as are possible with fuzzy logic and artificial intelligence, are needed to help with the analysis, visualization and high level interpretations for data of the enormous scales found in large scientific databases.

The determining factors for commercialization are demand, need and feasibility. First we need to consider if there has been an expressed demand for uncertainty representation and management that is currently not being provided so that a strong potential use of fuzzy logic in information systems could be anticipated in the near term. Fuzzy logic has been used in many expert systems as a mechanism for uncertainty management. However, in the database community there does not appear to be a similar demand. The simplest approach to imprecision is the use of null values in the relational model Codd (1979) and the conclusion is that there does not appear at present to be an expressed demand for uncertainty management past the simple approach of null values.

The majority of efforts in the use of fuzzy logic in databases have focused in most cases on the underlying data model of Buckles and Petry (1982), Prade and Testemale (1984), Umano (1982), Zemankova, Kandel (1985). Is this where there is the greatest current need for uncertainty management? There are three fundamentally independent aspects of databases that could utilize fuzzy logic: schema design, data modeling, and querying systems.

Relative to schema design, it is well accepted that there is a great deal of difficulty in capturing and modeling many real-world enterprises and a fuzzy linguistic schema specification would be mapped into an underlying crisp data model Buckles and Petry (1995).

A number of researchers have proposed approaches to fuzzy querying of crisp data bases. Again, this does not require the underlying data model to be fuzzy, just the representation of the user's understanding or specification of imprecision in their query, see Bosc, Gailbourg and Hamlin (1988).

The need for fuzziness in data modeling can be viewed as providing the capability to directly represent imprecision and also to support some aspects of fuzzy schemas and fuzzy queries more efficiently, however, there are significant feasibility problems for general purpose fuzzy data models because of database management systems' performance requirements.

The feasibility issues relative to AI and by extension to fuzzy logic compared with databases are apparent. In the former areas, a major issue has been the capability to enhance and extend the domain of representation. In databases, the issue is dominantly performance.

Because the direct inclusion of fuzzy data would usually entail the need for approaches such as non-first normal forms in the relational database model, it is likely that performance degradation would prohibit this at the present time. This makes it seem likely that front-end fuzzy querying systems have the greatest potential in the near-term based on performance criteria, see Kacprzyk and Zadrożny (1995), Nakajima, Sogoh and Arao (1993). One system has been developed by OMRON as an interface to the Oracle DBMS and the second, created at the Systems Research Institute in Poland, is intended for use on personal computers with systems such as Microsoft Access. Both use similar approaches with a fuzzy SQL, although in different system scales. In both fuzzy front-end systems, fuzzy SQL is pre-processed into SQL acceptable to the specific DBMS and then the query results that are retrieved must be post-processed to re-introduce the fuzzy structuring into the results.

Another way in which fuzzy set approaches have been used are in special purpose databases such as that at Bellcore Mansfield and Fleischman (1993). The situation in special purpose databases is that the approach can be tailored to enhance performance of the fuzzy data component since the system is not intended as a general purpose database system.

In this paper we want to consider approaches to support general purpose fuzzy databases. In particular, we will overview concepts of database machines and the Bellcore Datacycle system. Then we will describe preliminary steps to design of a backend processor for a similarity–based fuzzy database.

2. Database machines

There have been a number of attempts at classifying database machines. One includes backend systems, storage hierarchy, intelligent controllers and database computers, Champine (1978). Another uses the categories, cellular logic systems, backend computers, integrated database machines and high speed associative memory systems, Su et al. (1980). One common feature of the above is the use of special purpose systems to support database retrieval. This serves to remove a large percentage of database functions from the host machine and thus to provide the critical performance requirements needed for commercial systems. The overlap of processing with the host and specialized design to support specific retrieval functions are the aspects we shall describe in our design in the next section.

The Datacycle architecture which has been developed at Bellcore, was designed to permit a database processing system that uses filtering technology to perform an efficient, exhaustive search of an entire database. It has been observed that fuzzy queries place severe stress on the indexing and I/O subsystems of conventional database systems since they frequently involve the search of large numbers of records. The use of the Datacycle architecture eliminated the need for complex index structures, provided high-performance query throughput, permitted the use of ad hoc fuzzy membership functions and provided deterministic response time largely independent of query complexity and load as described by Fleischmann and Mansfield (1993).

A possible implementation technique for a fuzzy query is to utilize specialized database index structures that associate records to fuzzy sets, Bosc and Galibourg (1989). The database index structures avoid the complexity of evaluating the membership function against every tuple in the database during query processing. This would allow high-speed access for a number of predetermined fuzzy predicates. However, arbitrary queries involving derived data cannot efficiently make use of these index structures and force the run-time execution of membership functions. The performance penalty due to run-time membership function execution is further adversely affected by the need to perform set intersection and union operations involving large sets.

The Datacycle architecture, see Bowen et al. (1992) was developed to provide high-performance transaction processing, powerful query language capabilities, and high levels of concurrent access by multiple applications, all in a single architecture. The heart of this approach is a high speed, on-the-fly, data filtering operation, which in Datacycle supports enhanced query processing capabilities including their technique for fuzzy query processing. The approach permits the ad hoc definition of membership functions in the query grammar, arbitrary use of numeric attributes in the database, and high performance.

In this architecture, entire databases are broadcast over high bandwith communications facilities to specialized filtering hardware, see Lee, Matoba, Mak (1991), in order to perform complex data selection and aggregation operations needed for query evaluation. Their membership functions are represented in libraries as trapezoids with a common set of breakpoints (A, B, C, D) to define the range (support) of the membership function. During query parsing, breakpoints are substituted for the linguistic membership functions specified in the query. Due to the characteristics of the datafilter, membership functions are limited to piecewise linear functions because of the lack of multiply instruction in the VLSI datafilter.

This ability to dynamically scale the membership function from the statistical domain to the domain of an arbitrary subset of the data is a significant departure from approaches that depend on the static definition of the membership function and its index structures. It enables meaningful fuzzy query processing for a much larger set of applications, and reduces the amount of database specific knowledge required of a user.

3. Similarity query architecture

3.1. Similarity database concepts

The use of similarity relationships in a relational model attempts to generalize the concept of null and multiple-valued domains for implementation within an operational environment consistent with the relational algebra. The nonfuzzy relational database is a special case of this fuzzy relational database approach.

For each domain, j, in a relational database, a domain base set, D_j, is understood. Domains for fuzzy relational databases are either discrete scalars or discrete numbers drawn from either a finite or infinite set. An example of a finite scalar domain is a set of linguistic terms. For example consider a set of terms that can be used for subjective evaluation of the contamination of the sites in the environmental database: *critical, severe, poor, so-so, average, good, excellent.* The fuzzy model makes use of a similarity relationship to allow the comparison of these linguistic terms.

A similarity relation, $s(x, y)$, for given domain, D, is a mapping of every pair of elements in the domain onto the unit interval $[0, 1]$, which is, $x, y, z \in D$, Zadeh (1971):

1. Reflexive: $s_D(x, x) = 1$
2. Symmetric: $s_D(x, y) = s_D(y, x)$
3. Transitive: $s_D(x, z) \geq \max(\min[s_D(x, y), s_D(y, z)]) : (T1)$

This particular max-min form of transitivity is known as T1 transitivity. Another useful form is T2, also known as max-product:

3'. Transitive: $s_D(x, z) = \max([s_D(x, y) \cdot s_D(y, z)]) : (T2)$

where $\cdot$ is arithmetic multiplication.

An example of a similarity relation satisfying T2 transitivity is:

$$s_D(x, y) = e^{-b \cdot |y-x|}$$

where $b > 0$ is an arbitrary constant and $x, y \in D$.

The identity relation used in nonfuzzy relational databases induces equivalence classes (most frequently singleton sets) over a domain, D, which affect the results of certain operations and the removal of redundant tuples. The identity relation is replaced in this fuzzy relational database by an explicitly declared similarity relation of which the identity relation is a special case.

Next the basic concepts of fuzzy tuples and interpretations must be described. A key aspect of most fuzzy relational databases is that domain values need not be atomic. A domain value, d_i, where i is the index of the attribute in the tuple, is defined to be a subset of its domain base set, D_i. That is, any member of the power set may be a domain value except the null set. Let $\mathbf{P}(D_i)$ denote the power set of $D_i - \emptyset$.

A *fuzzy relation* R is a subset of the set cross product $\mathbf{P}(D_1) \times \mathbf{P}(D_2) \times \ldots \times \mathbf{P}(D_m)$. Membership in a specific relation, r, is determined by the underlying semantics of the relation. For instance, if D_1 is the set of major cities and D_2 is the set of countries, then (Paris, Belgium) $\in \mathbf{P}(D_1) \times \mathbf{P}(D_2)$ – but is not a member of the relation A (capital-city, country).

A *fuzzy tuple*, t, is any member of both r and $\mathbf{P}(D_1) \times \mathbf{P}(D_2) \times \ldots \times \mathbf{P}(D_m)$. An arbitrary tuple is of the form $t_i = [d_{i1}, d_{i2}, \ldots, d_{im}]$ where $D_j \subseteq d_{ij}$.

An *interpretation* $a = [a_1, a_2, \ldots, a_m]$ of a tuple $t_i = [d_{i1}, d_{i2}, \ldots, d_{im}]$ is any value assignment such that $a_j \in d_{ij}$ for all j.

In summary, the space of interpretations is the set cross product $D_1 \times D_2 \times \ldots \times D_m$. However, for any particular relation, the space is limited by the set of valid tuples. Valid tuples are determined by an underlying semantics of the relation. Note that in an ordinary relational database, a tuple is equivalent to its interpretation.

3.2. Similarity query formulation

To illustrate the process of query evaluation in the similarity database, we examine a generalized form of Boolean queries that may also be used to retrieve information, Petry (1996). The details of query evaluation can be seen more easily in this sort of queries.

A query $Q(a_i, a_h, \ldots, a_k)$ is an expression of one or more factors combined by disjunctive or conjunctive Boolean operators: V_i op V_h op $\ldots$ op V_k. In order to be well formed with respect to a relation r having domain sets $D_1, D_2 \ldots D_m$, each factor V_j must be

1. a domain element a, $a \in D_j$, where D_j is a domain set for r, or
2. a domain element modified by one or more linguistic modifiers, e.g. NOT, VERY, MORE-OR-LESS.

The relation r may be one of the original database relations or one obtained as a result of a series of fuzzy relational algebra operations. Fuzzy semantics apply to both operators and modifiers. An example query is

MORE-OR-LESS big and NOT VERY VERY heavy

where "big" is an abbreviation of the term (SIZE = big) in a relation having domain called SIZE. The value "heavy" is likewise an abbreviation. The linguistic hedge VERY can be interpreted as CON(F), concentration, and MORE-OR-LESS as DIL(F), dilation.

A membership value of a tuple in a response relation r is assigned according to the possibility of its matching the query specifications. Let $a \in D_j$, be an arbitrary element. The membership value $ma(b)$, $b \in D_j$, is defined based on the similarity relation, $s_j(a, b)$, over the domain. The query $Q(\diamondsuit)$ induces a membership value $m_Q(t)$ for a tuple to in the response r as follows:

1. Each interpretation $I = [a'_1, a'_2, \ldots, a'_m]$ of t determines a value $m_{aj}(a'_j)$ for each domain element a_j, of Q $(a_i, a_h, \ldots, a_k)$.
2. Evaluation of the modifiers and operators in $Q(\diamondsuit)$ over the membership values $m_{aj}(a'_j)$ yields $m_Q(I)$, the membership value of the interpretation with respect to the query.
3. Finally, $m_Q(t) = \max I$ of $t\{m_Q(I)\}$.

In short, the membership value of a tuple represents the best matching interpretation. The response relation is then the set of tuples having nonzero membership values. In practice, it may be more realistic to consider only the tuple with the highest value.

4. Similarity database architecture design

The major aspect we address here is the use of the basis of the similarity database – the similarity relation. The key will be the use of an associative memory to map the domain elements to a matrix to generate the similarity. This follows approaches to the use of associative processors for implementing maybe logic in the relational algebra, Hurson and Miller (1987), Miller and Hurson (1987).

Fig. 1 shows the high level description of the query evaluation steps discussed above. These steps are done in parallel across the interpretations of the matching

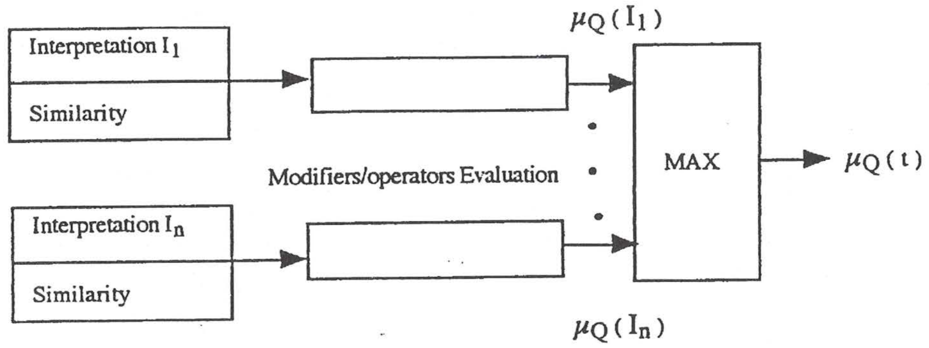

Figure 1. Query interpretation architecture design

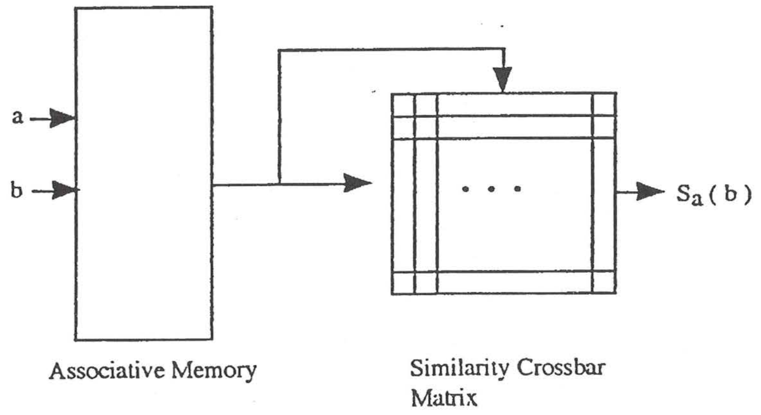

Figure 2. Similarity module for Fig. 1

tuples generated during the evaluation of the query relative to the similarity-based fuzzy database.

Fig. 2 shows the design of the similarity module. This is used to obtain the required similarity values to generate membership values for the interpretations. The linguistic terms corresponding to an attribute domain being evaluated are stored in the associative memory arrays. This facilitates direct indexing of the hardware similarity matrix from which the specific similarity values are then read.

5. Conclusions

If we desire to actually move the laboratory developments of fuzzy databases to full scale commercial systems, serious consideration of performance requirements must be made. Although many approaches using various indexing schemes and advances in processor and I/O speed and main memory capacity can make inroads into the performance problems, the overhead of general purpose fuzzy database still remains. The overhead of fuzzy processing has been successfully addressed in other applications by the development of fuzzy chips. Thus we have been led to begin the considerations of special purpose database hardware to enhance performance. The issues raised and preliminary design sketched in this paper are being further elaborated and a more complete design that can be simulated for testing and performance estimation is in development.

References

BOSC, P. and GALIBOURG, M. (1989) Indexing principles for a fuzzy database. *Inform. Systems*, **14**, 493-499.

BOSC, P., GAILBOURG, M. and HAMLIN, G. (1988) Fuzzy querying with SQL: extensions and implementation aspects. *Fuzzy Sets Syst.*, **28**, 333-39.

BOWEN, T., GOPAL, G., HERMAN, G., HICKEY, T., LEE, K., MANSFIELD, W., RAITZ, J. and WEINRIB, A. (1992) The Datacycle^{TM} architecture. *Communications of the ACM*, **35**, 71-81.

BUCKLES, B. and PETRY, F. (1982) A fuzzy representation of data for relational databases. *Fuzzy Sets Syst.*, **7**, 213-226.

BUCKLES, B. and PETRY, F. (1995) Fuzzy databases in the new era. *Proceedings of FUZZ-IEEE/IFES '95 Workshop on Fuzzy Database Systems and Information Retrieval*, 85-91.

CHAMPINE, G. (1978) Four approaches to a database computer. *Datamation*, 101-106, Dec.

CODD, E. (1979) Extending the database relational model to capture more meaning. *ACM Trans. on Database Systems*, **4**, 156-174.

HURSON, A. and MILLER, L. (1987) A database machine architecture for supporting incomplete information. *Jour. Comp. Sys and Science Eng.*, **2**, 231-239.

KACPRZYK, J. and ZADROŻNY, S. (1995) FQUERY for Access: fuzzy querying for windows-based DBMS. *Fuzziness in Database Management Systems*, P. Bosc and J. Kacprzyk, eds., Physica-Verlag, Heidelberg, 415-435.

LEE, K., MATOBA, T., MAK, V. (1991) VLSI accelerators for large database systems. *IEEE Micro*, **11**, 8-20.

MANSFIELD, W. and FLEISCHMAN, R. (1993) A high-performance, ad-hoc, fuzzy query processing system. *Jour. Intelligent Inform. Sys.*, **2**, 397-420.

MILLER, L. and HURSON, A. (1987) Supporting maybe algebra in the associative search language machine. *SIGMOD Record*, **16**, 61-80.

NAKAJIMA, H., SOGOH, T. and ARAO, M. (1993) Development of an efficient fuzzy SQL for a large scale fuzzy relational database. *Proc. of 5th IFSA World Congress*, 517-530, Seoul, Korea.

PRADE, H. and TESTEMALE, C. (1984) Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries. *Inf. Sci.*, **34**, 115-43.

PETRY, F. (1996) *Fuzzy Databases: Principles and Applications.* Kluwer Academic Publishers, Norwell, MA, USA, 78-80.

SU, S., CHANG, H., COPELAND, G., FISHER, P., LOWENTHAL, E., SCHUSTER, S. (1980) Database machines and some issues of DBMS standards. *Proc. NCC*, 191-208.

UMANO, M. (1982) FREEDOM-0: A Fuzzy Database System. *Fuzzy Information and Decision Processes*, M. Gupta and E. Sanchez, eds., North-Holland, Amsterdam, 339-347.

ZADEH, L. (1971) Similarity relations and fuzzy orderings. *Information Sciences*, **3**, 177-206.

ZEMANKOVA, M., KANDEL, A.(1985) Implementing imprecision in information systems. *Inf. Sci.*, **37**, 107-141.