Control and Cybernetics

vol. **24** (1995) No. 3

A general view on fuzzy relational database systems and querying

by

Włodzimierz Kwasowiec

ul. Gołuchowska 3/33,
01-485 Warsaw, Poland

A simple and general view of fuzzy querying in relational database systems is presented. Fuzzy database systems are defined basing on traditional ones and using a relation which more precisely reflects the internal structure of the database. Fuzzy querying in fuzzy database systems is proposed and discussed. Basic properties of these database systems are briefly examined.

Keywords: Relational database systems; fuzzy database systems; fuzzy querying; matching degree function.

1. Introduction

The purpose of the present paper is to simplify our understanding of relational database systems and to introduce in a most general way the fuzzy database systems basing on traditional ones. To this end we shall start with formulation of conventional databases and generalize them to fuzzy databases and infer main properties of fuzzy databases as set against conventional (non-fuzzy) ones.

There are various models of fuzzy databases (see e.g. Anvari and Rose 1987; Kacprzyk et al. 1990; Shenoi et al. 1990; Yazici et al. 1992). Some authors defined fuzzy database by means of an equivalence relation (e.g. Anvari used distinguishability relation, Shenoi used the one of similarity) and in this way they obtained a partition of the database domain. Tripathy and Saxena (1990) dealt with fuzzy databases by means of fuzzy relations containing membership grade function. We want to get a similar general model using simpler tools.

Kacprzyk et al. (1989;1986), Bosc and Pivert (1992) dealt with queries – a basic notion of database theory; Kacprzyk has proposed linguistic quantifiers for fuzzy queries. Zemankova and Kandel (1984) presented a model of fuzzy database systems and a query language. They discussed in detail the measures of imprecision and various operations and applied them to querying. We are going to introduce the notion of query as general as possible and we want to discuss its main features in traditional and fuzzy databases.

One important aspect which will be taken into consideration is imprecision of membership of contents of a database. We will not deal here with imprecision of values of particular attributes stored in the database and, moreover, we will not consider neither so called measure of nearness nor pairwise similarity. Some subjects are put off to more detailed considerations. We shall deal with processing of vague information in the sense of imprecise belonging to a database table. Therefore if a database system is used to model an information system of the real world, then by means of the degree of imprecision we indicate "how much" particular information of the system satisfies user's conditions.

We shall introduce conventional database systems and try to extend them in a way to give users more convenient ("human-consistent") tools for database management. First we shall define relational database structure by means of partition of a database onto tables and moreover we assume to have a distinguished relation specifying internal relationship between attributes of different tables. We shall introduce usual queries in traditional databases and we shall generalize them to fuzzy queries (again in traditional databases). Later we shall generalize traditional databases to fuzzy ones and define there fuzzy queries. We shall discuss querying and show its general characteristics in conventional and in fuzzy database systems.

2. Basic notation

Here we state some general assumptions.

Let $ATTR$ be a set of attributes (their names).

By $Dom(f)$ we denote the domain of the function f.

By $f|X$ we denote the restriction of the function f to the set $X \cap Dom(f)$.

Let DT be a set. Its elements are called types and each type is a set composed of (for now: non-fuzzy) values.

Since types are sets, sometimes we shall need to speak about their elements and therefore we adopt the denotations:

$Elem(DT)$ – for the set of elements of all types (the union of all types),

Therefore: $Elem(DT) = \{x : x \in Type \text{ for some } Type \in DT\}$

3. Traditional database systems

By a database we mean a set of tables and a relation

$$B = \langle TBL, LINK \rangle$$

where TBL is a set of tables (database components),
$LINK$ is a relation between attributes of different tables (to be described later).

$LINK$ does not show the relational structure of the database like in Codd (1970) – this relational structure is reflected here by table structure. The relation $LINK$ shall establish internal structural dependencies between parts of the database.

Now we define tables.

By a table T from TBL we mean a pair

$$T = \langle Str(T), Cont(T) \rangle$$

$Str(T)$ is a partial function (the structure of T) from $ATTR$ into DT.

Its domain instead of $Dom(Str(T))$ will be written shortly $Attr(T)$ because it is the set of attributes assigned to the table T. Therefore:

$$Attr(T) \subseteq ATTR,$$

There may happen that an attribute (its name) is used in many tables. We assume for such case that the same type is assigned to it by the function Str in all these tables (otherwise we might take a new name for an attribute with another type), i.e. we assume:

$$\text{if } a \in Attr(T1) \cap Attr(T2) \text{ then } Str(T1)(a) = Str(T2)(a)$$

$Cont(T)$ is a set of functions from $Attr(T)$ into elements of data types determined by the structure $Str(T)$. It consists of all current contents of the table T. These content functions are sometimes called tuples (e.g. in Shenoi et al. 1990), sometimes are called *subset*-values (e.g. in Tripathy and Saxena 1990). We assume:

$$\begin{aligned} Cont(T) \quad \subseteq \quad & \{c : ATTR \to Elem(DT) \text{ such that } Dom(c) = Attr(T) \ \& \\ & c(attr) \in Str(attr) \text{ for each } attr \in Attr(T)\} \end{aligned}$$

Because one attribute may be used in many tables, we shall often use pairs $\langle attribute, table \rangle$ to avoid confusion.

We denote: $Cont(B) = \{\langle\langle attr, T\rangle, val\rangle : T \in TBL \ \& \ \langle attr, val\rangle \in Cont(T)\}$.

Thus $Cont(B)$ is the union of all content functions of the whole database B with attributes specified as above pairs. Sometimes we shall write simply $c|X0$ for $c \in Cont(B)$ if it will not cause any confusion ($X0$ will consist of attributes of only one table).

Now come to the description of the relation $LINK$ – a feature which characterizes relational databases in more detail. Generally we avoid to duplicate information in the database. However sometimes we shall admit to assign the same pieces of information to attributes in different tables to fix a relationship between these tables. Various attributes with the same values may be used to identify elements of different tables (which is simillar to the idea of so called "primary" and "foreign" keys). To this end we define $LINK$ as a general relation between attributes of different tables, i.e. between pairs: an attribute and a table which contains that attribute. We used here the word "link" because we wish to link these attributes which may contain the same values.

So we assume:

$$LINK \subseteq (ATTR \times TBL) \times (ATTR \times TBL)$$

and it must satisfy:

(i) if $\langle a1, T1\rangle LINK\langle a2, T2\rangle$ then $T1 \neq T2$ and $a1 \in Attr(T1)$ and $a2 \in Attr(T2)$ and $Str(T1)(a1) = Str(T2)(a2)$

(ii) if $T1 \neq T2$ and $a \in Attr(T1) \cap Attr(T2)$ then $\langle a, T1\rangle LINK\langle a, T2\rangle$

$LINK$ may be symmetric and transitive relation but under (i) it may not be a reflexive one. Thus

PROPOSITION 3.1 *$LINK$ is not an equivalence relation.*

The relation $LINK$ shows an essential structural relationship between tables. It may happen that it is empty relation and it means that different tables are completely independent. Mostly interesting for us are database systems with reacher structure where $LINK$ is not empty. Thus from (ii) we obtain an obvious property concerning non-emptiness of the relation $LINK$.

PROPOSITION 3.2 *If there exists $a \in Attr(T1) \cap Attr(T2)$ for some $T1 \neq T2$ then $LINK$ is not empty.*

3.1. Example

We shall consider a simple database $STUDENT$ which contains tables: $PERSONAL$, $GRADE$, JOB and possibly some other ones. We are not going to present the structure of these tables in detail. We shall show only their attributes. Namely:

$$\begin{aligned} Attr(PERSONAL) &= \{Id\_nr, Name\} \\ Attr(GRADE) &= \{Id\_nr, Course, Grade\} \\ Attr(JOB) &= \{Id\_nr, Job, Salary\} \end{aligned}$$

We assume that student is uniquely determined by the value of its Identity number (Id_nr) and different students may have the same name. We can have various content functions, e.g.

$$\begin{aligned} (\langle Id\_nr, 1\rangle, \langle Name, Brown\rangle) &\in Cont(PERSONAL), \\ (\langle Id\_nr, 2\rangle, \langle Name, Kwasowiec\rangle) &\in Cont(PERSONAL). \\ (\langle Id\_nr, 1\rangle, \langle Course, Math\rangle, \langle Grade, 5\rangle) &\in Cont(GRADE), \\ (\langle Id\_nr, 2\rangle, \langle Course, Math\rangle, \langle Grade, 4\rangle) &\in Cont(GRADE). \end{aligned}$$

It denotes that Brown has got 5 and Kwasowiec has got 4 from Math but we are interested in finding more information using (involved) queries. Here $LINK$ relates (identifies types) the attribute Id_nr in all these tables and is defined to be symmetric and satisfying:

$$\begin{aligned}
\langle Id_nr, PERSONAL\rangle \quad & LINK \quad \langle Id_nr, GRADE\rangle \\
\langle Id_nr, PERSONAL\rangle \quad & LINK \quad \langle Id_nr, JOB\rangle \\
\langle Id_nr, JOB\rangle \quad & LINK \quad \langle Id_nr, GRADE\rangle.
\end{aligned}$$

4. Querying in traditional database systems

Users may require from a computer system to deliver various information originating from data stored in the database. To this end queries of database systems might constitute a good utility.

We are not going to discuss query language in detail, like it is e.g. in Ullman (1982) or Bosc and Pivert (1992) or Kacprzyk et al. (1989;1986). Moreover we are going to simplify considerations: in a query we shall look for information of only one table but that information may be determined by conditions concerning data of the whole database.

So a classic query (in its simplified version – for one table) is a 3-parameter formula:

$$Query0(T, X0, Cond0),$$

where T is a table,

$$X0 \subseteq Attr(T),$$

$Cond0$ is a condition – it involves some attributes and restricts their values.

This query $Query0$ should be understood in the following way:

"Find all values of attributes from $X0$ of the table T satisfying $Cond0$".

The condition $Cond0$ might be a simple condition restricting only values of attributes of the table T but also more involved one concerning attributes of many tables. It may be specified in various ways by means of any attributes from the set $ATTR$ and any values from the set $Elem(DT)$. We may search for concrete values, for ranges of values of some attributes but we may not yet use here so called "fuzzy values". Since the present approach is very general we are not going to deal with the structure of the $Cond0$ in detail.

A query is a formula which transmits our information demands to a computer system. The query should be processed and we are interested in the results obtained after such processing.

The result of the query $Query0(T, X0, Cond0)$ is the set of functions:

$$\begin{aligned}
Res(Query0) \quad = \quad & \{res : X0 \to Elem(DT) \text{ such that there exists} \\
& c \in Cont(B) \text{ which satisfies } Cond0 \;\&\; res = c|X0\}.
\end{aligned}$$

Hence query returns results which were required by users and are determined in parameters specified by users in a given condition (using values of some attributes). The returned results are partial database content functions containing interesting for users, current information taken from the database.

Remark. Queries should be specified carefully and properly. It might seem that we do not need information about many attributes in a query and therefore we choose a small set $X0$. However it may happen that a query returns restrictions of many contents as one partial function (restriction caused by too small subset $X0$) and then we get wrong information about e.g. the number of returned elements. If such information is important to us then we should specify query for larger set $X0$, even for the set of all attributes of that table.

4.1. Example

We adopt assumptions of the example 3.1. We may ask about names of students who has got the grade 5, i.e. we ask $Query1(T, X1, Cond1)$, where $T = PERSONAL$, $X1 = \{Name\}$, $Cond1 = (Cont(GRADE)(Grade) = 5)$

Then we obtain: $Res(Query1) = \{\langle Name, Brown\rangle\}$

To get better information we should specify: $X1 = \{Id\_nr, Name\}$ because otherwise when two students of the name Brown got 5 we would obtain the same result and this way we could loose essential information.

5. Fuzzy querying in traditional database systems

For fuzzy approach to databases we shall need fuzzy values, fuzzy relations, fuzzy quantifiers and a function which will enable us to determine adequate values that give "the best matching" for a given query (i.e. to find all information from the database which fit best to the specified query).

So we assume to have fuzzy objects:

F_val – a set of names of fuzzy values (e.g. "*high*")

F_rel – a set of names of fuzzy relations (e.g. "*is much greater than*")

F_qua – a set of names of fuzzy quantifiers (e.g. "*majority of ...*")

Now we introduce FQ to be the set of all so called "fuzzy queries". Fuzzy query is also (like query in the non-fuzzy case) a 3-parameter formula

$$Query0(T, X0, Cond0) :$$

where T is a table, $X0 \subseteq Attr(T)$, similarly to the non-fuzzy case

$Cond0$ is a condition (restricting some attributes) – but now it may contain fuzzy values, fuzzy relations and fuzzy quantifiers.

We assume also to have a function which will show the membership or the matching degree of some contents under "fuzzy conditions".

md – matching degree function with results in $[0, 1]$

Generally md is a function which depends on (fuzzy) queries and content functions:

$$md : FQ \times Cont(B) \rightarrow [0,1]$$

When we shall deal in future with the query $Query0$ in more detail, we shall decompose the condition $Cond0$ to small components containing either fuzzy value or fuzzy relation or fuzzy quantifier and then we shall define md by means of correspondent operators joining these components. Now we want to show the domain dependency of the function md for two components: F_val and F_rel. Fuzzy values are values of some (perhaps all) attributes and we must know which attribute is under consideration. Fuzzy relations concern pairs of attributes (perhaps of different tables).

$$md : Attr(T) \times F\_val \times Cont(B) \rightarrow [0,1]$$

$$md : Attr(T1) \times F\_rel \times Attr(T2) \times Cont(B) \rightarrow [0,1]$$

Fuzzy quantifiers, whose use was proposed by Kacprzyk and Ziółkowski (1986) concern a certain part of the condition and therefore they are applied to some indirect results (after evaluation of that part of the condition at contents of considered table). The domain dependency of the function md for F_qua is more involved and we put it off to future considerations. More detailed discussion of these fuzzy objects can be found in Kacprzyk et al. (1989).

The result of the query $Query0(T, X0, Cond0)$ is also the set of content functions (similarly to the non-fuzzy case) which now is additionally determined by md:

$$\begin{aligned} F\_Res(Query0) \quad = \quad & \{res : X0 \rightarrow Elem(DT) \text{ such that} \\ & \text{there exists } c \in Cont(B) \text{ satisfying} \\ & Cond0 \ \& \ res = c|X0 \ \& \ md(Query0, c) > 0\} \end{aligned}$$

5.1. Example

We continue our example. We may ask "to find all students who have better grades in main courses than their total earnings". First we must formally define the meaning of our strange words in this context (fuzzy value "main" and fuzzy relation "better") according to:

$$md : \{Course\} \times "main" \times Cont(PERSONAL) \rightarrow [0,1]$$

$$md : \{Grade\} \times "better" \times Salary \times Cont(PERSONAL) \rightarrow [0,1]$$

We use the table PERSONAL because we want to find numbers and names of students satisfying our condition. So $T2 = PERSONAL$ and $X2 = Attr(PERSONAL)$ in our $Query2(T2, X2, Cond2)$.

To use quantifiers we might ask $Query3$, e.g. "*to find all students for whom nearly all of (.....) hold*" where instead dots in brackets we write several conditions. We should define the meaning of the quantifier "*nearly all*" in the function md and then in the set of results $F\_Res(Query3)$ we could obtain all students (their identity_number and name) for whom the value of md is greater than 0.

In practice the set $F\_Res(Query0)$ will usually be shown as a set ordered by md in descending order (most interesting for us are information contained in elements with the highest matching degree). Moreover these results might depend on a given threshold but such considerations will be put off to fuzzy databases.

Usually for simplification of considerations we want to deal with values of all attributes of a given table for an arbitrary query and then we would have $X0 = Attr(T)$, i.e. the first parameter of $Query0$ would be determined by T. So a query might often be considered as a formula parameterized by T and then dependent only on a condition in the following way:

$$Query0_T(Cond0).$$

6. Fuzzy database systems

First we assume to adopt all above notations. Tripathy and Saxena (1990) have used fuzzy relations defined by means of tuples (here: content functions) and a membership grade function. We shall proceed similarly. Now we shall add to each content function a fixed attribute to be able to keep in it (as its value) membership grades of content functions.

Now we assume that a distinguished attribute Md_attr exists in $ATTR$ and a distinguished type composed of real numbers $Type = [0, 1]$ exists in DT.

We require that the attribute Md_attr belongs to $Attr(T)$ for each $T \in TBL$ and that $Str(T)(Md\_attr) = [0, 1]$ for each $T \in TBL$. This attribute will be used to keep the degree of membership (expressed here by means of a real number) of an arbitrary content function.

By a fuzzy database (general definition is similar to the traditional case) we shall mean a set of tables and a relation

$$B = \langle TBL, LINK \rangle$$

where similarly to traditional databases TBL is a set of tables (database components), $LINK$ is a relation between pairs: attributes and tables.

Similarly to traditional case a table T from TBL is a pair:

$$T = \langle Str(T), Cont(T) \rangle$$

where $Str(T)$ is a partial function from $ATTR$ into DT, with our main assumptions:

(i) $Md\_attr \in Attr(T)$ & $Str(T)(Md\_attr) = [0, 1]$ for each $T \in TBL$

(ii) if $a \in Attr(T1) \cap Attr(T2)$ then $Str(T1)(a) = Str(T2)(a)$

and $Cont(T)$ is a set of functions from $Attr(T)$ into elements of data types determined by the structure $Str(T)$ (it describes all current contents of the table T) with the assumption concerning Md_attr:

$$Cont(T) \subseteq \{c : ATTR \to Elem(DT) \text{ such that } Dom(c) = Attr(T) \;\&\; c(attr) \in Str(attr) \text{ for each } attr \in Attr(T)\}$$

So each content c is defined at Md_attr and this value shows us "the degree of belongingness" to the database table (higher $c(Md\_attr)$ means better belongingness of c to our database). If $c(Md\_attr) = 0$ then we understand that actually this c does not belong to the database.

One could think that we should restrict $Cont(T)$ only to such content functions c that $c(Md\_attr) > 0$. We do not remove from the database the contents which equal 0 at Md_attr but we will not consider them when dealing with current state of the database. They will show a piece of "the history of the behaviour of our database".

Moreover like in traditional case we denote:

$$Cont(B) = \{\langle\langle attr, T\rangle, val\rangle : T \in TBL \;\&\; \langle attr, val\rangle \in Cont(T)\}.$$

Now we define $LINK$ (similarly to the traditional case except for Md_attr) as a relation between pairs of attributes and tables:

$$LINK \subseteq (ATTR \times TBL) \times (ATTR \times TBL)$$

and it must satisfy:

(i) if $\langle a1, T1\rangle LINK \langle a2, T2\rangle$ then $T1 \neq T2$ and $a1 \in Attr(T1)$ and $a2 \in Attr(T2)$ and $Str(T1)(a1) = Str(T2)(a2)$

(ii) if $T1 \neq T2$ and $a \in Attr(T1) \cap Attr(T2)$ then $\langle a, T1\rangle LINK \langle a, T2\rangle$

We emphasize that though $LINK$ is defined here using the same conditions as before, we must take into account the attribute Md_attr which is situated in all tables of the database and therefore the type of the attribute Md_attr is the same in all tables. Consequently we have the following properties:

PROPOSITION 6.1 *If TBL consists of more than one table then $\langle Md\_attr, T1\rangle LINK \langle Md\_attr, T2\rangle$ for all different tables $T1, T2 \in TBL$.*

COROLLARY 6.1 *If TBL consists of more than one table then $LINK$ is not empty.*

6.1. Example

We adopt assumptions of the example 3.1. To get fuzzy database we must add the attribute Md_attr to all tables. Its current value may be interpreted: how much somebody (e.g. the dean) considers given person to be a student.

7. Fuzzy querying in fuzzy database systems

In section 5 we defined fuzzy querying in traditional database systems. This definition with using a formula $Query0(T, X0, Cond0)$ remains the same but now it is applied to extended specification of databases. The formula $Query0$ is parameterized by a table T which is now enriched by the attribute Md_attr, by a set of attributes $X0$ which may now contain a new attribute Md_attr and by a condition $Cond0$ which now also might be more powerfull.

Of course for fuzzy databases we may use fuzzy values from F_val, fuzzy relations from F_rel, fuzzy quantifiers from F_qua. We will need also the function md because it will evaluate the matching degree for a given query. Its definition is similar to traditional case:

$$md : FQ \times Cont(B) \to [0, 1]$$

Now the structure of tables is extended and therefore the function md depends additionally on values of the distinguished attribute Md_attr. We can keep an information about actual membership grade of a given content in Md_attr and modify it consecutively by means of currently gained values of md.

For example: in the condition $Cond0$ of the fuzzy query $Query0$ we may use also the attribute Md_attr to require all content functions with the value of Md_attr not less then a given threshold.

Similarly to the non-fuzzy case the result of the query $Query0(T, X0, Cond0)$ is the set of functions which are now determined also by md:

$$\begin{aligned} F\_Res(Query0) \quad = \quad & \{res : X0 \to Elem(DT) \text{ such that} \\ & \text{there exists } c \in Cont(B) \text{ satisfying} \\ & Cond0 \;\&\; res = c|X0 \;\&\; c(Md\_attr) > 0 \;\&\; \\ & md(Query0, c) > 0\} \end{aligned}$$

The set $F\_Res(Query0)$ is usually shown as a set ordered by md in descending order and most interesting for users are information contained in elements (partial content functions) with the highest matching degree.

Traditional databases correspond to some fuzzy databases. Namely to get fuzzy database system we must extend traditional database (all its tables) by the attribute Md_attr and define:

$$c(Md\_attr) = 1 \text{ for each table } T \text{ and for all } c \in Cont(T).$$

Therefore we obtain:

PROPOSITION 7.1 *Traditional databases are particular cases of fuzzy database systems.*

In future this property will be useful in the description of an implementation of fuzzy querying in traditional databases.

We could generalize $F\_Res(Query0)$ to $F\_Res(Query0, thres)$, where $thres$ is a fixed real number (it is the threshold) from the set $(0, 1]$ which might be different for different queries. Then we define:

$$\begin{aligned} F\_Res(Query0, thres) \quad = \quad & \{res : X0 \to Elem(DT) \text{ such that} \\ & \text{there exists } c \in Cont(B) \text{ which satisfies} \\ & Cond0 \ \& \ res = c|X0 \ \& \\ & c(Md\_attr) \geq thres \ \& \\ & md(Query0, c) \geq thres\} \end{aligned}$$

Remark. The set $X0$ could be adopted in $Query0$ more generally as the union of attributes of various tables.

In this paper we discussed fuzzy queries in general terms and we assumed that the fundamentals for dealing with them in more detail.

References

ANVARI M., ROSE G.F., (1987) Fuzzy relational databases, in *The Analysis of Fuzzy Information* (J. Bezdek, Ed.), CRC Press, Boca Raton.

BOSC P., PIVERT O., (1992) Fuzzy querying in conventional databases, in *Fuzzy Logic for the Management of Uncertainty*, (L. Zadeh & J. Kacprzyk Ed.), Wiley & Sons.

BUCKLES B.P., PETRY F.E., (1987) Generalized Database and information systems, in *The Analysis of Fuzzy Information* (J. Bezdek, Ed.), CRC Press, Boca Raton.

CODD E.F., (1970) A relational model for large shared databases, *Comm. Assoc. Comput. Mach.*, **13**(6), 377-387.

KACPRZYK J., ZADROŻNY S., ZIÓŁKOWSKI A., (1989) Fquery III: A "human-consistent" database querying system based on fuzzy logic with linguistic quantifiers, *Information Systems*, **14**, 443-453.

KACPRZYK J., ZIÓŁKOWSKI A., (1986) Database queries with fuzzy linguistic quantifiers, *IEEE Trans. Systems, Man and Cybernetics*, **11**, 474-479.

KACPRZYK J., BUCKLES B.P., PETRY F.E., (1990) Fuzzy information and database systems, *Fuzzy Sets and Systems*, **38**, 133-135.

SHENOI S., MELTON A., FAN L.T., (1990) An equivalence classes model of fuzzy relational databases, *Fuzzy Sets and Systems*, **38**, 153-170

TRIPATHY R.C., SAXENA P.C., (1990) Multivalued dependencies in fuzzy relational databases, *Fuzzy Sets and Systems*, **38**, 267-279.

ULLMAN J. D., (1982) *Principles of database systems*, Computer Science Press, Rockville, Maryland.

YAZICI A., GEORGE R., BUCKLES B.P., PETRY F.E., (1992) A survey of conceptual and logical data models for uncertainty management, in *Fuzzy Logic for the Management of Uncertainty*, (L. Zadeh & J. Kacprzyk Ed.), Wiley & Sons.

ZEMANKOVA M., KANDEL A., (1984) *Fuzzy relational data bases – a key to expert systems*, Verlag TUV Rheinland, Köln.