Proc. 8th Int. Workshop on Database and Expert Systems Applications (Dexa '97), Tolouse, France, September 1-2 1997
IEEE Computer Society, Vol. 7/97

VQS - A Vague Query System Prototype

Josef Küng, Jürgen Palkoska
Research Institute for Applied Knowledge Processing (FAW), University of Linz, Altenberger Str. 69
A-4040 Linz, Austria, e-mail:jk@faw.uni-linz.ac.at

Abstract

Conventional relational database systems are designed to produce exact query results. If there is no record complying to the specified query conditions, the system will return an empty answer. In this case the user is forced to modify the query criteria and to try again. This process would gain much more user-friendliness if the database system would suggest alternative query results by itself, in the case of failure of a conventional query.
In this paper an extended query system called VQS will be introduced. This system obtains the query result on semantic information which is hidden behind the attribute values and not visible to the user. The consequence of VQS is to retrieve records semantically close to the query.
VQS has been designed to be an application independent attachment for any relational database. In the case of using an existing database its structure has not to be changed. The prototype explaining and evaluating the concept works with any Oracle database.

1. Introduction

The results of relational databases are records which match exactly to the conditions specified in the query. In many cases this is sufficient. If the query "The data of employee Scott" fails, obviously no data is available for this person and it makes sense when the database system gives an empty result. The data of an other person is not interesting.

But in other application domains such a precise behavior brings disadvantages. As an example in a tourism information system in which a person looks for a hotel room in Salzburg which should be cheaper than 1000 ATS (Austrian Shillings). When the system gives an empty answer, the user will formulate another query with the condition "price lower than 1200" and at least he will find a room which costs 1100 ATS. He will never find a room with 950 ATS in the little neighbor village Aigen.

In this case an information system would be much more user friendly when it answers: "For the specified query conditions is no data available in the database, but, the following results are suitable alternatives. ..." Therefore the system needs additional knowledge behind the attribute values. Unfortunately relational database systems have no concept for semantic background information.

The motivation behind the work described in this paper was the idea to build a concept for modeling such semantic background information which can be used universally. This concept should enable any (also an existing) database system to give vague query results. Main objectives are the application domain and database system independence.

2. Related research

First we want to introduce different levels of vagueness. The classification is based on Lee D.H. and Kim M.H. [1]. Unfortunately they did not consider that database systems with precise values inside and allowing only exact queries can also give vague results. Therefore the classification was extended with the criteria "result set".

1. Crisp Data, Crisp Query and Crisp Result (CDCQ-CR): A typical relational database system. Exact values are stored, only exact queries are allowed, and the result set is also precise.
2. Crisp Data, Fuzzy Query and Fuzzy Result (CDFQ-FR): The query allows also words like "big", "very expensive" and so on. As a consequence the result set is also fuzzy.
3. Fuzzy Data, Fuzzy Query and Fuzzy Result (FDFQ-FR): Here the data itself can be fuzzy.
4. Crisp Data, Crisp Query and Fuzzy Result (CDCQ-FR): In this case the system is able to produce results to an exact query which are records that match only approximately. A query in this system could be "An employee with a salary about 8000". "about" is the fuzzy element in the query. But, it is not a fuzzy value only a fuzzy operator.

2.1. Fuzzy Based Solutions

Basing on the theory of Fuzzy-Sets [2], [3] a lot of work has been done in several application areas, also in the database domain. Extensions of SQL from Bosc, Galibourg and Hamon [4], the Fuzzy Database-Query Language from Wong and Leung [5], Fuzzy Base from Gazzotti, Piancastelli, Sartori and Beneventano [6] are some examples. Most of them have to be classified into the CDFQ-FR-Level. This means that they use fuzzy data, which is not compatible to most of the existing databases. The fuzzy base approach is situated in the FDCQ-FR level. Here exact queries are allowed and in the case of an empty result the system builds a range query and executes it. Step by step it increases the ranges until the first results occur. This solution is limited to numeric values.

2.2. ARES

In 1986 Ichikawa and Hirakawa presented the Associative Information Retrieval System (ARES) [7] to get fuzzy results for an exactly formulated query. The extension of the query language was done by only one operator. It implies "approximately equal to" or "similar to" for cases in which the user expects a flexible interpretation of the query conditions from the system. ARES translates this operator into the well known operations of the relational algebra. Therefore the system can be added to an existing database very easily.

In ARES the necessary semantic background information is based on so called Similarity Relations. Figure 1 shows an example.

SR-Job	A-Element	B-Element	Similarity
	Technician	Technician	0
	Technician	Programmer	1
	Technician	Assembler	1
	Technician	Clerk	2
	Technician	Manager	2
	Technician	Seller	4
	....	...	...

Figure 1. Example of a Similarity Relation.

So it is possible to formulate conditions containing the operator "is similar to" on attributes where a Similarity Relation is defined. Then the user has to specify the degree of ambiguity and the system returns the result. In the example above a query like "All employees with a job similar to technician" will return all technicians, programmers and assemblers when the degree of ambiguity is set to 1.

Based on this idea, in ARES also the combination of conditions with the operator "similar to" and joins are possible.

2.3. VAGUE

Beside the relational algebra of ARES which is extended by the operator "similar to", VAGUE presented by A. Motro in [8] is an very interesting approach to enhance facilities of relational databases. In this case "similar to" is represented by the sign "~". This implementation bases on the idea of data metrics.

Each attribute corresponds to a domain. For each domain at least one metric is defined. There are four possible types of metrics:

1. A data metric is *computational* if it derives its distances by computation only (i.e., no retrievals are involved). An example of a computational metric on a numerical domain, such as SALARY, is the absolute value of the difference between two numbers. An example for a computational metric on a non-numerical domain, such as PERSON_NAME, is a procedure that determines the degree of similarity between two strings of characters.
2. A data metric is *tabular* if it derives its distances by retrieval only (i.e., no computations are involved). The distance between all value pairs of the domain is stored in a table, and the metric simply searches this table. An example for a Tabular Metric is the geographic distance between locations. This tables are comparable with the Similarity Relations in ARES.
3. Sometimes an attribute can be mapped to another existing relation where this attribute is the key. Therefore the distance between two values of this attribute can be interpreted as the difference between their descriptions; that is, a combination of the individual distances between the corresponding components. Such metrics are called *referential* metrics. Motro gives the following example: Consider a relation FILM with the attributes TITLE, DIRECTOR, CATEGORY and RATING and let (▫▫▫▫▫▫▫ ▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫) and (▫▫▫▫▫▫▫▫▫▫▫▫▫ ▫▫▫▫▫▫▫▫▫ ▫▫▫▫▫▫▫▫ ▫▫▫) be two tuples from this relation. The distance between ▫▫▫▫▫▫ and ▫▫▫▫▫▫▫ ▫▫▫▫▫ may be defined as a combination of the individual distances between ▫▫▫▫▫▫▫▫▫ and ▫▫▫▫▫▫▫,▫▫▫▫▫▫▫▫▫ and ▫▫▫▫▫▫, and ▫▫▫ an ▫▫▫.
4. When a domain cannot be provided with a suitable metric, the *standard* metric should be used:

$$DEFAULT(x,y) = \begin{cases} 0 \ \ if \ \ x = y \\ 1 \ \ if \ \ x \neq y \end{cases}$$

If the operator "~" occurs in a query, the query processor uses the corresponding metric for calculating the results, which are represented in an ordered set (if the user wants), containing the best matches (in the metric nearest) first. VAGUE incorporates several features that contribute to its flexibility. For example, multiple metrics for each domain are supported; it allows users to judge the relative importance of attributes of referential metrics.

3. The basic idea of VQS

3.1. NCR-Tables

With the approaches ARES and VAGUE in the background we designed our system which is able to enhance relational database systems with vague retrieval capabilities. We called it *Vague Query System* (VQS). Our idea was to use a representation of nonnumeric values as numeric coordinates in a feature space like the solutions in numeric classification. The needed semantic background information for attributes is stored in so called *NCR-Tables* (Numeric-Coordinate-Representation-Tables). Such a table consists of the attribute itself as the key (the *NCR-Key*) and the *NCR-Columns* which represent dimensions. Figure 2 shows an example for representing colors in a NCR-Table as numeric values.

Colors	Name	red	green	blue
	black	0	0	0
	blue	0	0	255
	light blue	75	75	255
	dark blue	0	0	150
	...	...	...	..
	white	255	255	255

Figure 2. Example of a NCR-Table.

We call attributes which are mapped to a NCR-Table *Fuzzy Fields*. NCR-Tables are conventional relations in a database. Still existing tables can be used as such NCR-Tables, a thing which fairly often happens. NCR-Tables replace the Similarity Relations from ARES or the Tabular Metric from VAGUE. Beside this, they have some advantages:

They are smaller. For n attribute values, n^2 entries in the Similarity Relation are necessary, but only n in the NCR-Table.

Less maintenance effort is required. In the case of adding a new attribute value n additional records are needed in Similarity Relations, only 1 is needed in NCR-Tables.

3.2. Similarity of attribute values

With the concept of the NCR-Tables the similarity of nonnumeric attribute values can be defined by the distance of the corresponding coordinates. We chose the euklidean distance. In the case of numeric attributes the distance can be calculated from the values itself. No NCR-Table is necessary.

3.3. Similarity of whole records

Normally queries have more than one criteria. For instance the query "A hotel room in Salzburg cheaper than 1000,-- ATS" has the price criteria (< 1000) and the location criteria (in Salzburg). Now the system has to find records which are nearest over all criteria. A normalization is the consequence. In VQS the minimal and maximal values of the NCR-Table are computed for each dimension. These numbers represent a cube where the diameter is the longest possible distance. The normalization is based on this value and transfer distances into the interval [0,1]. Figure 3 gives a graphical view on this normalization.

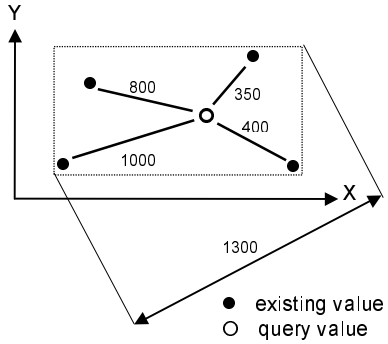

Figure 3. Graphical view on normalization

Now the distance from the query to a whole record can be defined by the average of the normalized distances based on each query criteria. This result (the *total distance*) is again in the interval [0,1] and it can be used directly as a presuming number for goodness. A total distance of 0.1 is quite good and 0.9 means "very far away".

But sometimes, the average of the normalized distances is not satisfactory. For instance, in the example above the price criteria should be more important than the location criteria. In this case it is possible to define individual weights in VQS.

4. The query language

The whole VQS-system should be an enhancement of a relational database system. The query language VQL (vague query language) is close to the SQL-syntax. Figure 4 shows a formal definition of this language.

```
VQLExpression =   "SELECT FROM" DataSource
                  "WHERE" Conditions
                  "INTO" destinationTableName
DataSource =      ([ownerName"."]rootTableName |
                  ([ownerName"."]rootViewName |
                  "("sqlSelectStatement")"
Conditions =      columnName "IS" ValueExpression
                  {"AND"
                  columnName "IS" ValueExpression}
ValueExpression = (" ' "alpanumericValue " '") |
                  numericValue
                  ["WEIGHTED BY" numericValue]
```

Figure 4. Formal description of VQL.

DataSource contains the whole functionality of SQL. Based on the DataSource the operater IS can be used for Fuzzy Fields. A condition like "Town IS 'Salzburg'" forces the system to make a ranking of all possible DataSource results. The nearest result is presented first.

At this level, only the boolean operator AND is implemented to combine more than one IS-operation. The WEIGHTED BY - option allows to weight the conditions individual. The numeric value in this clause must be an integer (>= 1). The default value is 1.

The result of a VQL-Statement is stored in a table of the relational database which is defined in the INTO-clause.

5. System Architecture

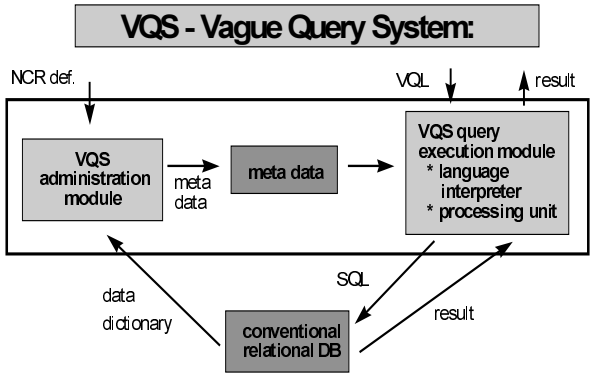

Figure 5. The system architecture of VQS

The System consists of several modules and a meta database. Both can be added to an existing relational database in an easy way. The modules are grouped by user types. The administrator needs functionality for defining the Fuzzy Fields and managing some additional parameters. Users communicate with the modules which interpret and execute the VQL-Statements.

5.1. Vague query administration module

This module represents the whole functionality needed by the administrator of VQS to manage meta data. It checks the consistence between Fuzzy Fields, NCR-Tables and the data dictionary of the conventional database.

5.2. Meta database

The information which Fuzzy Fields and NCR-Table-Columns are mapped is stored in the meta database. Due to the fact that existing tables, more exactly some columns of existing tables can be used as NCR-Tables, the concept of logical NCR-Tables was introduced. This concept represents a layer between Fuzzy Fields and physical NCR-Columns.

5.3. Vague query execution module

First the Vague Query Language Interpreter makes the syntactic check of the VQL-statement, which was submitted by the user. Then it compiles the statement into a formalized meta language which will be sent to the query processing unit.

The query processing unit is the kernel of VQS. Based on information from the meta database conventional SQL-statements are generated. Afterwards the SQL-statements are transmitted to the relational database system.

6. The Prototype

6.1. Implementation

The prototype of VQS was implemented as an add on to Oracle7 databases. We used the PL/SQL language from Oracle. Because of this we achieved a platform and operating system independent solution. Beside this, the interpretation and execution of a VQL-statement will be done on the server, not in an application on a client.

Therefore the communication between client and server is reduced to a minimum.

We realized the modules of VQS as PL/SQL-Packages. So the communication can be done by simple calls of stored procedures via every interface supported by Oracle, even ODBC.

6.2. Additional functionality

Numeric attributes need no semantic background information in the NCR-Tables. The metric is calculated and normalized over the domain itself. Therefore every numeric attribute can be used in a VQL WHERE-clause. In the case of non-numeric attributes it seems that only attributes with corresponding NCR-Table-Columns are allowed in combination with the IS operator. According to the standard metric in VAGUE we defined a *standard distance function* in VQS which returns 0 when two attributes values match exactly, otherwise it returns 1. This function allows every non-numeric attribute to be used in a VQL WHERE-clause too.

In VQL views of a relational database can be used as data sources. Views can base on other views and so on. VQS is able to look for a definition of NCR-Columns assigned to attributes along this view definitions, down until to basing table attributes are reached. The first corresponding NCR-Definition which will be found is the definition which will be used. The implementation of this feature is very sophisticated. It considers renaming of attributes and joins in the view definition.

The next step is, that the same feature can be used for SQL-Statements in the DataSource. Semantic background information will be inherited from attributes of the tables or views which are involved in this SQL-statement.

By the way of views or SQL-statements as DataSource, any combination of exact query conditions with vague conditions is possible. Exact query conditions are formulated in the SQL-statement or in the view whereas the IS-operator is situated in the VQL-statement.

7. Conclusion

We introduced a system, which can be used as an enhancement of an existing relational database. It extends the query facilities with the feature that records which do not match exactly are in the result set too, ordered by the distance to the query values.

A new language (VQL) was defined containing the operator "IS" ("similar to") and the DataSource which can be a conventional table, a view or a SQL-Statement as major elements.

NCR-Tables (numeric coordinate representations) are the base of the concept. In this NCR-space normalized distances represent how precise records match to the query condition values.

A prototype was implemented to evaluate our concept and the result was quite good.

Our future work will concentrate to following points: First we want to get more experience in larger databases. Then the system could be extended by following features:

- Additional normalizing techniques.
- Domain specific semantic background information: It should be possible to define more than one NCR-Table for an attribute. This authorizes the user to decide which NCR-Table should be used in the query.
- Vague joins

8. References

[1] D.H. Lee, M.H. Kim, "Accommodating Subjective Vagueness Through A Fuzzy Extension to the Relational Data Model", *Information Systems*, Vol. 18, No. 6, Pergamon Press, 1993, pp. 363-374

[2] L. Zadeh, "Fuzzy Sets", *Information and Control*, Vol. 8, 1965, pp. 338-353

[3] L. Zadeh, "Fuzzy Sets As a Basis for a Theory of Possibility", *Fuzzy Sets and Systems*, Vol. 1, 1978, pp. 3-28

[4] P. Bosc, M. Galibourg, G. Hamon, "Fuzzy Querying with SQL: Extension and Implementation Aspects", *Fuzzy Sets and Systems*, Vol. 28, No.3, 1988, pp. 333-349

[5] M.H. Wong, K.S. Leung, "A Fuzzy Database-Query Language", *Information Systems*, Vol. 15, No. 5, 1990, pp. 583-590

[6] D. Gazotti, L. Piancastelli, C. Sartori, D. Beneventano, "Fuzzy Base: A Fuzzy Logic Aid for Relational Database Queries", *Database and Expert Systems Applications, 6th International Conference DEXA 1995*, Ed. Norman Revell, A Min Tjoa, Springer-Verlag Berlin Heidelberg New York Barcelona Budapest Hong Kong London Milan Paris Tokyo, London 1995, pp. 385-394

[7] T. Ichikawa, M. Hirakawa, "ARES: A Relational Database with the Capability of Performing Flexible Interpretation of Queries", *IEEE Transactions on Software Engineering*, Vol. 12, No. 5, 1986, pp. 624-634.

[8] A. Motro, "VAGUE: A User Interface to Relational Databases that Permits Vague Queries", *ACM Transactions on Office Information Systems*, Vol. 6, No. 3, 1988, pp. 187-214