Implementation of an Inference Engine for Fuzzy Databases

I. Blanco iblanco@decsai.ugr.es
J. C. Cubero JC.Cubero@decsai.ugr.es
F. Cuenca fcuenca@decsai.ugr.es
O. Pons opc@decsai.ugr.es

Department of Computer Science and Artificial Intelligence
University of Granada
Avda. Andalucía, 38
18071 Granada (SPAIN)

Abstract

This work shows how a relational DBMS with deductive and imprecise capabilities can be improved by adding the capability of representing and handling imprecise values to an existing classical inference engine that works on an RDBMS. The possibility of making flexible queries on classical values stored in the database is also added.
Keywords: relational databases extension, fuzzy deduction, inference.

1 Introduction

Since the introduction of the relational model by Codd in [1], it has been used in a generalized way by many authors.

In such a model, we can store information about people's ages, and this can be recovered if we want to find out a specific person's age or even an interval containing such a value. But this type of information is not usual when a human being makes a reasoning. We usually ask for "young" or "old" people without considering that such concepts vary from one person to another, and that the edges of such concepts are not well-defined.

In order to accomplish this task, we have to distinguish between how such information is represented and handled. To deal with the problem of the representation of fuzzy information in the relational model, Medina, Pons and Vila presented GEFRED [4] but this is only one part of the problem studied. We also have to model another way of implicit reasoning which a person can use.

When a human being thinks, he obtains data from other data, i.e. he is able to deduce information that depends on itself. For instance, a person's father, mother and ancestors can be stored in separate tables, but ancestors can be obtained from the father and mother tables and queried in a fixed order. Another problem arises if we store ancestors extensionally, that of the subsequent modification of the ancestor table when a modification of the fathers or mothers table occurs.

When we obtain data from other data, we follow a set of steps or "rules" which we could represent in a logic structure and we could design an algorithm that uses this structure to access the tables to complete the dependent table.

In order to implement both fuzzy and deductive features in a classical RDBMS, Medina, Pons, Vila and Cubero introduced the FREDDI system in [3].

Consequently, we would have a representation of the flexible values and a language for querying them, a flexible query language for querying concrete values, a structure to represent logical rules, and an algorithm to deduce data from other ones.

2 Considerations of implementation

To develop the inference engine, two possibilities have been taken into account:

- *Weak coupling:* this is orientated to host languages, that is, general purpose programming languages that have been extended with database communication features. An inference engine designed using this type of language is a *built-out engine*, in the sense that it is external to the RDBMS and requires a network or some type of message protocol to communicate with the database.

- *Tight coupling:* this is orientated to database programming languages such as language PL/SQL© provided by Oracle©, that are less generic but which are incorporated into the RDBMS. A module developed with such languages is usually executed by the RDBMS. As a result, an inference engine developed with a language of this type is a *built-in inference engine*.

Once we have examined the two possibilities, we shall have to consider portability versus efficiency. Due to the amount of data handled by a system with deductive capabilities, the Built-in Inference Engine is more efficient for our purposes because database communication capability is not required and the RDBMS executes the module more efficiently than any external module. The architecture of the system is shown in figure 1. Once our choice has been made, we describe it taking into account the problems of both representation and handling associated with deductive and fuzzy data.

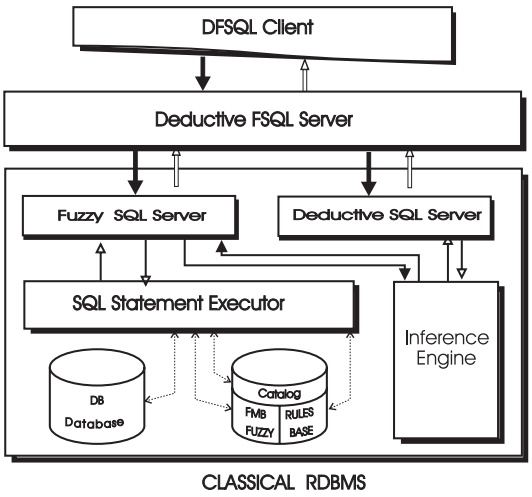

Figura 1: Architecture for Fuzzy Deductive SQL Server

3 Classical inference engine

As introduced in [3], two types of tables can be stored in a database with deductive capabilities. The first type are the so-called *extensional tables* whose content is stored explicitly in the table. The second type are the so-called *intensional tables* whose content depends on the contents of other tables.

Let us imagine a table representing a relation, for instance, a table called PARENT that

stores the parent together with each son. Now, let us suppose that we want to retrieve and store information about ancestors in a table whose content depends on the PARENT table. We could store this information explicitly but every time the PARENT table changes, we would have to manually change the ANCESTOR table. We could make ANCESTOR in some way dependent on PARENT. If we were to do this, then the process of updating it can be made automatically. The PARENT table can be seen as a predicate PARENT (X, Y) and each tuple stored in the table as a fact, for instance, PARENT ('Mike', 'Susan'). From this point of view, we could associate extensional tables to a type of predicate that we shall call "extensional" and intensional tables to "intensional" predicates. Once the tables have been represented logically, we can describe the relations established between intensional and extensional tables by using logical rules. An intensional predicate is described as a disjunction of sub-rules containing intensional or extensional predicates that are joined by a conjunction operator.

At this point, a query to the database can involve both extensional or intensional tables. The sub-query involving an extensional table is solved classically, that is, with a classical SELECT sentence. But the sub-query involving an intensional one is solved by filling in the table using the logical rules that define the condition that tuples satisfying such a predicate have to verify.

The algorithm to expand the rules that define an intensional predicate is based on the backtracking mechanism and on the resolution inference algorithm used to do this task, that is, variables instantiation. Each rule starts with a set of free variables on which predicates that are part of the rule will be applied, and some or all of the variables will be fixed with a value. When the algorithm applies a predicate to the set of variables associated to a rule, it can obtain more than one combination of values so at this point we shall have a set of several variables instead of one which is no longer valid because it has been processed. Once the algorithm has processed the current set, it can be deleted and another one becomes the active set. When the algorithm finishes, we shall have sets of several variables that are fully, partially or not instantiated. The set of fully instantiated variables has predicate results because a combination of values satisfying the rule has been found. Those that are partially or not instantiated are ignored because such a combination of values has not been found.

4 Fuzzy extension of the deductive module

Once we have studied how to improve a classical RDBMS with deductive capabilities, we accomplish the problem of incorporating the handling (representation and deduction) of fuzzy information. To achieve this, we start from the theoretical model for representing fuzzy information in the relational model presented in [4] and the concrete implementation of this introduced by J. Galindo in [2]. On the other hand, we use the theoretical model to represent deductive information presented in [3].

When we apply predicates of a rule to a variables set, we operate as follows: when the first predicate is applied, no variable is instantiated so the values combination returned corresponds to all tuples contained in the table

associated to the predicate. But from this moment, some variables are instantiated in all variables set. This is not a problem in classical deduction because, in this case, when the next predicate is applied only those tuples whose values are equal to non free variables values are important. In classical deduction, the comparator used is equality but now fuzzy comparators must be considered as a coincidence or matching measure.

But this is not the only consideration to be taken into account. Once we have decided which fuzzy comparator has to be applied, it cannot be applied directly. In classical deduction, only those tuples whose values coincided with non free variables values appeared in the result. But in fuzzy terms, all values in a domain coincide with a concrete value with a degree that can be zero. Exploring all values can increase the number of variables sets exponentially and some of them can be irrelevant. So we have to modify the model in such a way that allows us to associate a threshold to each predicate in a rule if necessary, or even to each argument contained in these predicates. Finally, DDL sentences will have to be modified in order to accept this type of fuzzy logical rules.

5 Conclusions

This work shows how the relational model can be improved and accepts the representation and management of fuzzy and/or deductive information. This part is an extension of works that separately manage fuzzy information or deductive information.

Advantages and handicaps of the two possibilities for the inference engine implementation are analyzed. Finally, a built-in solution is chosen since it offers more advantages in data transfer between the DBMS and the inference engine. If we consider the implementation assumed, the language SQL has been improved so that it can accept the new data types introduced (DDL) that can be fuzzy or intensional, and manipulate such data types (DML) (so far, only relevant sentences have been implemented).

In the future, we will add an importance degree to the rules so that tuples obtained by expansion of one or other rule do not have the same weight. In order to perform this task, we will have to unify the imprecision measures obtained with the importance value associated to the rule.

References

[1] E.F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[2] J. Galindo. *Tratamiento de la Imprecisión en Bases de Datos Relacionales: Extensión del modelo y adaptación de los SGBD actuales*. PhD thesis, Department of Computer Sciences and Artificial Intelligence, University of Granada, SPAIN, 1999.

[3] J.M. Medina, O. Pons, J.C. Cubero, and M.A. Vila. Freddi: A fuzzy relational deductive database interface. *International Journal of Intelligent Systems*, pages 597–613, 1997.

[4] J.M. Medina, O. Pons, and M.A. Vila. Gefred: A generalized model of fuzzy relational databases. *Information Sciences*, 76(1-2):87–109, 1994.