Mathware & Soft Computing 3 (1996) 415-424

Client/Server Architecture for Fuzzy Relational Databases

J.M. Medina, O. Pons, M.A. Vila, J. C. Cubero
Dept. Computer Sciences and Artificial Intelligence
E.T.S. de Ingeniería Informática.
Universidad de Granada 18071. Granada (Spain)
e-mail: medina@robinson.ugr.es

Abstract

This paper shows a FRDBMS architecture whose main characteristics are: 1) it is implemented entirely on classical RDBMS just using the resources provided by them, 2) it preserves all the operations and qualities of the host RDBMS and gives them more power adding new capabilities to deal with"fuzzy" information and 3) it provides a frame to develop applications which exploit fuzzy information.

1 Introduction

Since the appearance of the Relational Database proposed by Codd, several approaches have tried to provide a theoretical environment for the representation and handling of fuzzy information. These approaches, which are based on the use of the fuzzy sets theory as a tool of representation and manipulation, are grouped mainly into two tendencies: *Unification Models by Similarity Relations* and *Relational Models on Possibility Distributions*.

The first approach is described in Buckles & Petry's proposal [1]. The second approach comprises several proposals, the most important of which were introduced by Umano, [13], Prade and Testemale, [12] and Zemankova [14].

In[6, 7], we present a model, named GEFRED, which attempts to synthesize the most relevant aspects of the previous approaches within a common theoretical framework.

Fuzzy Relational Database Management Systems (FRDBMS) extend the capabilities of the classical RDBMS in order to treat fuzzy information. They are based on a theoretical model and on some ideas which make possible to build and operative implementation. Several proposals have arisen with this aim, defended most of them by the authors of the different theoretical models [3, 4, 10, 12, 13, 14].

In [5] we propose some notions to represent and implement fuzzy operations and data on classical RDBMS within the theoretical framework established in GEFRED.

This work presents some ideas to integrate the elements related to imprecise information handling within the field of modern architectures of database management systems in general. Exactly, it is proposed a FRDBMS architecture based on the Client/Server approach with the requirements that the modules which compose it must join to resolve the different aspects implicated in the treatment of information (fuzzy or not): access on line to data, development of applications, support for the decision, accessibility to data from other applications, etc.

Based on the architecture introduced in [9], we will describe the elements which compose it in the next section. In section 3, we will establish the minimum requirements which a RDBMS must include to be used for the implementation of Server and Client components of the FRDBMS proposed, we will present some ideas to build them and we will study methods of communication between both components into net environment. In section 4, we will show some aspects about the building of the architecture proposed on RDBMS Oracle7 and its development environment. Finally, we will analyze the advantages our approach presents and we will study future works which develop it.

2 A General Description of the Architecture

Figure 11 shows the general scheme of the proposed FRDBMS named FIRST (**F**uzzy **I**nterface for **R**elational **S**ys**T**ems) [9]. First of all, it is necessary to explain that we consider clients to the applications which exploit the data of the server and not the tools of development this server provides. The main elements of FIRST architecture are: the FSQL Server and the programming interfaces that this one provides. Both are built by the resorts the host RDBMS provides and they must be joined strongly with the SQL Server and its programming interface. Round the FIRST Server the client applications are built, which take charge of exploiting the data that this one manages. Next, we will describe the elements which compose FIRST attending to its functional character, structure and implementation:

Host RDBMS It is the system used as basis of an implementation of FIRST. It provides the management resources for relational database and a group of basic tools for programming SQL applications in the "host" language, (by an API or an embedded SQL specification). Incorporated SQL syntax is provided with a fuzzy extension and is used in the implementation of the conversion routines of FSQL to SQL sentences. In next section we will analyze some additional characteristics which RDBMS must own to make easy the implementation of an appropriate FSQL Server.

DB It stores all the permanent information, "fuzzy" or not, by the use of the relational scheme. Certain representation criteria are adopted in order to implement the "fuzzy" data in this data scheme.

Figure 1: General Scheme of FIRST

FMB Fuzzy Meta-knowledge Base is an extension of the host RDBMS catalog whose mission is to provide FIRST with the information about all the fuzzy structures, data and definitions contained in the DB. All this information is organized in accordance to the scheme of the host RDBMS catalog.

FSQL Server This module extends the capability of the host SQL Server in order that it can process fuzzy requests to the system expressed in FSQL. To carry out this job, it owns a parser for the FSQL syntax adopted and a protocol for the communication with client applications which incorporate FSQL code. This protocol can be based on an extension of the API provided by the host RDBMS or on an embedded FSQL syntax which can be provided with different precompilers. FSQL Server contains the following modules:

- **FSQL Processor** It transforms FSQL sentences into SQL sentences of the host RDBMS, executes them and sends the results to the client application.

- **FCL** (Fuzzy Call Library) is a library of functions which contains the routines that permit to carry out the interface tasks supported by the FSQL Server.

FSQL Precompilers They are a group of programs which, interacting with FSQL Server, permit to transform embedded FSQL applications into applications which are compiled and linked with the libraries provided by the FCL.

Net Protocol and DB "Drivers" FIRST can be supported by any platform in which the host RDBMS is, and it must be able to support the communication protocols at the net level and DB "drivers" it supports. In section 3 we will incise about these aspects.

FIRST clients can accede to the system using one or some of the next alternatives:

Embedded FSQL Applications written in a host language which incorporate embedded FSQL sentences. FIRST provides this syntax and precompilers for the most important host languages.

FAPI Fuzzy Application Programming Interface to incorporate the access to the FSQL Server in host applications. This interface must be joined strongly to the API of the host RDBMS.

Non Fuzzy Access Applications which require a conventional access to the RDBMS can use directly the interfaces provided by this one, (API or/and embedded SQL).

3 FIRST implementation

The specification of FIRST is established over four points: a) the theoretical model adopted, GEFRED [6], b) the representation criteria adopted for fuzzy information (data and operations) [5, 7], c) the description of a data sub-language (FSQL) [8] and d) the implementation using the resources provided by a conventional RDBMS. In this section, we will talk about some aspects related to d), and first, we will create the specifications the components of FIRST must satisfy and, then, we will establish the minimum requirements a host RDBMS must own to make easy the implementation of them.

3.1 FIRST specifications

The more important elements of the proposed architecture are: the FSQL Server, the programming interface for "fuzzy" applications and the protocols between clients and FSQL Server in net environments. We will establish the specifications of these elements and, according to this and the implementation techniques introduced in [5, 7, 8, 9], we will create the resorts which must be provided by the host RDBMS. Let's see these specifications:

- **FSQL Server**

  1. It will provide an extension of SQL syntax which will allow to process host SQL sentences combined or not with clauses for the treatment of fuzzy information.
  2. It will incorporate a processor for this FSQL syntax.
  3. This processor will translate a FSQL sentence into a group of sentences executed by the host RDBMS.
  4. It will store into the host RDBMS the fuzzy data to operate and the necessary information to do it.
  5. The FSQL processor and the routines used in its construction have to be implemented into a language provided by the host FSQL and be stored into the Server.

- **FAPI**

  1. It will be composed by one specification to build applications which can access to the FSQL Server.
  2. The specification will be a superset of API which provides host RDBMS. Additional operations incorporated will be determined by the treatment of fuzzy data in their transference from Server to client application.
  3. It will be composed by a library based on the FCL which will be incorporated to these applications. This library can be developed into all the languages the host API supports and it will incorporate clearly the access through the net.
  4. FAPI library will be developed using the host API through the library this one provides.
  5. Because of this, it is interesting that the API of host RDBMS is developed in a language which permits to use the code again. An object oriented language with a library of types will make easier the development.

- **Communication Protocol**

  1. Communication between the FSQL Server and the clients will be carried out over the standard net protocols supported by the host RDBMS.
  2. This communication must be done through "drivers" provided by the host RDBMS. These "drivers" can be "native" (they just only permit communication between database clients and servers from an only maker), or they can be based on an open communication protocol which permits the access to servers of different makers, for example ODBC (Open Data Base Connectivity).
  3. The use of the communication protocol will be clear for the user.

3.2 Host RDBMS Requirements

To satisfy these specifications the host RDBMS must join the following requirements:

• **Minimum**

1. Support of the most powerful SQL syntax.
2. Client/Server architecture.
3. To have some alternatives to develop applications in "host" languages which permit the access to the server data by SQL (API or embedded SQL).
4. Implementation of some communication protocol of data between Server and Client from different platforms.

All the present RDBMS observe these requirements which permits the implementation of the proposed architecture on a great variety of managers (in [7] can be found an example). However, if we want a perfect coupling of FIRST and the host RDBMS, this one must join most of the following requirements:

• **Additional**

1. To have a SQL procedural extension. This extension couples the data language with characteristics of Third Generation languages (C, Pascal, etc).
2. The capability of storing, in the database procedures and packets written into that procedural extension.
3. To provide a programming interface to build applications which access and execute these stored procedures.
4. This interface must support, at least, the use of a "native" protocol for the access to the data and procedures stored in the server. It would be able to accept also, an open protocol, if possible.

Present evolution of the RDBMS assures the presence of these characteristics in most of them through numerous platforms. All this guarantees the building of versions of FIRST coupled strongly to the host RDBMS for all possible combinations of machines, operative systems, RDBMS and net environments, also allowing the transference of information from one platform to another.

4 Example of implementation on a concrete RDBMS

To explain the possibilities of implementation of FIRST we are going to show the complete architecture of a solution which provides access to the FSQL Server in a remote machine from an application written over local Windows^{TM} platform. On the server side we are going to use the Oracle7^{TM} RDBMS on SunSPARC^{TM} under

Solaris^{TM} 2.x and, on the client side, we will use the Oracle Objects^{TM} for Windows. To access Oracle7 Server data through the net, we will use SQL*NetV1^{TM} for TCP/IP (the"native" connection protocol of Oracle). Figure 2 shows the architecture adopted in the example.

First of all we will describe the elements which constitute the architecture and the way to use the development tools provided by Oracle7 to build them. Finally, we will study the solving process of a generic FSQL query through the different modules.

4.1 Elements of the architecture

On line processor of FSQL This client application accepts the user requests in FSQL, transforms them into PL/SQL (the procedural extension of SQL provided by Oracle) and sends them to server component of OLE developed by Oracle to accept accesses to databases from Windows. This application can be developed in Visual Basic, Borland C++ or Visual C++, or in any combination of them. Oracle Objects for OLE provides the DLL, VBX controls and the libraries of C++ class essential for that.

Oracle Object Server It is an OLE 2.0 server developed on the OLE 2.0 server for Windows and it solves all the requests of access to an Oracle server made from an application.

Oracle Call Interface The services intercepted by the previous module are transformed by it into calls supported by this programming interface. This module takes charge of establishing the communication through the net by the use of the Oracle native drivers for the appropriate framework.

Sql*Net It is built on the net protocol and intercepts, clearly, the requests directed to the Oracle server, liberating the developer from implementing low level applications for the access to the server data through the net.

FSQL Server It takes charge of solving the requests in FSQL and translating them into SQL sentences executed by Oracle. It is developed in PL/SQL and incorporates a syntax and a semantics processor for the FSQL syntax adopted. When the sentence is translated, it sends it to the SQL executer to get the results. They are processed and sent to the client through a communication mechanism supported by the slide concept. Sometimes, FSQL server can require services of the SQL executer and, of course, services of the PL/SQL machine incorporated into Oracle Server (Figure 2 does not show it). FSQL Server is written in PL/SQL and is precompiled and stored into the Oracle Server.

In all these elements it is only necessary to implement the modules corresponding to the on line processor of FSQL and the FSQL Server. The other ones are available as elements of support and development of FIRST architecture.

Figure 2: Implementation of FIRST on Oracle7

4.2 Processing of a FSQL sentence

With the help of figure 2, we are going to process a sentence expressed in FSQL.

1. On line processor of FSQL, after receiving a statement, carries out a syntax parsing to verify that it is right and transforms the entrance into one or more processes in Oracle PL/SQL .
2. It requests its execution by Oracle Objects Server implemented in the Windows Client.
3. This one intercepts the call and changes the petition into a set of routines of the Oracle Call Interface.
4. These routines detect, automatically, the presence of the net support SQL*Net and use this driver to send, through the net protocol, the petition to the

SQL*Net available into the machine containing the Oracle7 Server.

5. This one receives the petition and sends it to the Oracle7 Server for processing.

6. This requests is composed by PL/SQL routines, which constitute the package that FSQL Server implements; so then it will be responsible of executing the petition.

7. The process called from the client processor has got the work of parsing the FSQL sentence which carries as argument and, if this one and the referenced data are right, translating it into a SQL sentence, which will be sent later to the SQL executer provided by Oracle7 Server. This process of parsing and translation uses the PL/SQL machine, the SQL executer, the database and the "fuzzy" catalog implemented into the system dictionary.

8. Execution of the sentence translated by the SQL executer can produce the recovery of some tuples which must be returned to the on line processor of FSQL for its edition and presentation to the user.

9. This process follow an inverse way of the described one and this client application is shown as a set of tuples stored by an structure used through the protocol the Oracle Objects for OLE provides.

10. The integration of this protocol with the visual characteristics of Windows allows the presentation of data in a suitable form for each necessity.

5 Conclusions

Architecture shown in this paper provides an excellent starting point for the building of FRDBMS. That is so because it makes use of the available resorts in present RDBMS and it is able to incorporate into them mechanisms for the handling of fuzzy information in a clear way for the user. Face to user, the proposed architecture provides a SQL fuzzy extension and development tools to build applications which exploit fuzzy or not information. In this way, we get FRDBMS which keep all the operations and characteristics of the present ones, (Client/Server approach, distributed management of information, integrity and safety of data,etc.) improving them with the possibility of handling imprecise information.

Future work lines will be directed to the development of prototypes on concrete platforms for the most important elements in this architecture: the FSQL Server, FSQL on line processor, a visual version to process imprecise queries and development tools of FSQL applications.

References

[1] Buckles B.P., Petry F.E. "A Fuzzy Representation of Data for Relational Databases". Fuzzy Sets and Systems, 7. 213-226. (1982)

[2] J.C. Cubero and M.A. Vila. "A new definition of fuzzy functional dependencies in fuzzy relational databases". International Journal of Intelligent Systems, 9(5):441–448, (1994).

[3] Hamon Guy. "Extension d'un language d'interrogation de Base de Donnees en vue de l'utilisation de questions imprecises", PhD. Disertation. (1986).

[4] Li D., Liu. "A Fuzzy Prolog Database System". ed. John Wiley & Sons. New York. (1990)

[5] Medina J. M., Vila M.A., Cubero J.C., Pons O. "Towards the Implementation of a Generalized Fuzzy Relational Database Model". Fuzzy Sets & Systems 75 pp 273-289 (1995)

[6] Medina J. M., Pons O., Vila M.A. "GEFRED. A Generalized Model of Fuzzy Relational Data Bases". Information Sciences,76,1-2, pp 87-109. (1994)

[7] Medina J. M. "Bases de Datos Relacionales Difusas: Modelo Teórico y Aspectos de su Implementación". PhD. Disertation. University of Granada. (1994)

[8] Medina J. M., Pons O., Vila M.A., "Un Procesador Elemental de Fuzzy SQL". 4º Congreso Español sobre Tecnologías y Lógica Fuzzy. Blanes (Gerona).(1994)

[9] Medina J. M., Pons O., Vila M.A. "FIRST. A Fuzzy Interface for Relational SysTems". Proceedings of the 6º Sixth International Fuzzy Systems Association World Congress (IFSA 95) vol II pp 409-412. (1995)

[10] Nakajima Hiroshi, Sogoh Taiji, Arao Masaki. "Fuzzy Database Language and Library - Fuzzy Extension to SQL -" Proceedings of the Second IEEE International Conference on Fuzzy Systems, pp. 477-482. (1993)

[11] O. Pons, J.C. Cubero, J.M. Medina, M.A. Vila. "Dealing with Disjunctive Missing Information in Logic Fuzzy Databases" to appear in International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems.

[12] Prade H., Testemale C. "Generalizing Database Relational Algebra for the Treatment of Incomplete/Uncertain Information and Vague Queries". Information Sciences, 34. 115-143. (1984)

[13] Umano M. "Freedom−0 : *A* Fuzzy Database System". Fuzzy Information and Decision Processes. Gupta-Sanchez edit. North-Holland Pub. Comp. (1982)

[14] Zemankova M., Kandel A. "Fuzzy Relational Data Bases - A Key to Expert Systems". Verlag TUV Rheinland. (1984)