A PROTOTYPE FOR RESEARCH ON HETEROGENEOUS DATABASE SYSTEMS

Rafael Alonso

CS-TR-102-87

June 1987

A PROTOTYPE FOR RESEARCH ON HETEROGENEOUS DATABASE SYSTEMS

Rafael Alonso

Department of Computer Science
Princeton University
Princeton, N.J. 08544
(609) 452-3869

ABSTRACT

When two organizations decide to share the information in their databases, it is not uncommon for them to find that the data stored in their respective computer systems are kept in incompatible database management systems. To allow users to query the combined database in a straightforward manner, a mechanism is needed that will mask the details of each particular systems and present a homogeneous interface to the collection of database systems. A *heterogeneous database system* enables users to transparently access the data contained in a multiplicity of differing databases. In this paper we describe some of the issues that must be addressed in the implementation of heterogeneous database systems, and in particular we focus on resolving the conflicts that arise in data integration. We also describe the design of a prototype that is currently being built to conduct research on this problem, one of whose salient features is the use of an embedded expert system to aid in data integration.

June 1987

A PROTOTYPE FOR RESEARCH ON HETEROGENEOUS DATABASE SYSTEMS

Rafael Alonso

Department of Computer Science
Princeton University
Princeton, N.J. 08544
(609) 452-3869

1. Introduction

When two organizations decide to share the information in their databases, it is not uncommon for them to find that the data stored in their respective computer systems are kept in incompatible database management systems (DBMS's). In some cases it may be possible for one of the two entities to convert their data to the other's system, although at a high cost in terms of time and effort. Even this costly solution may not be appropriate if the two organizations are simply co-operating on a temporary basis. Furthermore, there may be users that want to access the separate databases as well as those who want to query the joint one. We are currently studying this problem of logically integrating two (or more) dissimilar databases by exploring the issues involved in the design and implementation of a *heterogeneous database management system* (HDBMS).

Designing an HDBMS is a difficult task. It is at least as difficult as constructing a distributed database system (for example, such as R* [Williams82] or distributed INGRES [Stonebraker77]), since the databases to be united may potentially reside at different sites. Previous approaches to this problem have usually consisted of creating a global data structure that captures enough information about the format of the participating databases such that a query dealing with that global structure can be translated into queries that each local DBMS can understand (this point will become clearer after the discussion in the next section). For example, in Multibase [Dayal84], the designers of the HDBMS made use of a model which captures more semantics than the relational model (see [Date75] for a description of the later model). Their model incorporates the notion of *generalization* [Smith77], which essentially allows them to abstract the properties of a data item and to categorize it as part of a generic group. Thus, by grouping related items as a class, they can translate a (global) query about objects in a (globally known) class into (local) queries about each of the (locally known) components of that class.

A different approach is taken in [Motro87], which introduces the concept of *superviews*. This mechanism allows the integration of different databases, but those that are built on the same kind of DBMS (e.g., two different INGRES databases created by two separate organizations). This technique involves the creation of a global schema from which each of the local schemas can be

derived as a view.

Although the systems outlined above can be successfully used in some environments, it seems to us that suffer from two drawbacks. First, they involve very complex mechanisms; we feel that there must be simpler and more straightforward methods for capturing enough semantics about a group of databases to allow query translation. Secondly, in many situations it would be desirable to have a technique that would allow the gradual integration of the databases. For example, it seems reasonable to expect that, as users begin to share data with other organizations, they might begin by asking very simple queries dealing with foreign data, and eventually make more complex requests. A system that can function with varying degrees of semantic information (geared to the class of queries currently being asked by users) seems to us a tempting approach.

We are currently exploring the use of expert system based tools for data integration. Expert systems provide a natural framework for dealing with the semantics of the database information. Furthermore, it is the nature of such systems that they can be gradually extended by incorporating ever more detailed amounts of information. In this paper we sketch the ideas behind our approach, and briefly describe the work we have carried out to date.

In the next section, we give a brief description of what we feel are the main design issues involved in implementing an HDBMS. In Section 3, we present an overview of the system architecture of the general-purpose HDBMS which we are currently implementing; one of the main features of this system is that it will make use of an expert system to aid in query decomposition. This section also includes some examples of how we expect our HDBMS to behave in various situations. Section 4 provides some details about the actual software and hardware to be used in our prototype. We conclude by sketching our research plans for the near future.

2. Design Issues

One of the design goals of a heterogeneous database system is to provide a *uniform* way of accessing the data in the databases involved. By this we mean that the users should be shielded (as much as possible) from the fact that different databases are being queried on their behalf. Furthermore, ideally users should not be aware of the fact that the data is possibly physically distributed among different locations; that is, the user view of the data should be *location-independent.* If these two goals are met, we say that the user is provided with *transparent* access to an HDBMS.

The techniques required to provide location independence are the same as those employed in distributed homogeneous DBMS. To hide the differences among the databases (i.e., to provide uniform access), a variety of new research issues must be addressed. One of the principal questions that arise is how to present the user with a coherent view of the collective data (which may be stored in dissimilar ways). This issue may be resolved by defining a common data model and data manipulation language (DML). However, it may be difficult to design a global model that captures that idiosyncrasies of each and

every one of the databases involved, and it may be inefficient to translate queries into the common DML (especially if it is necessary to translate strictly local queries into the global DML before executing them). In some applications where there is one preferred (i.e., most commonly accessed) database, it may be preferable to provide translations from the other DBMS's to the data model and DML of the preferred one. This has the advantage of minimizing the overhead of executing the most common queries, but at the loss of generality in system design; if at some point in the future another DBMS turns out to be the most frequently accessed one, the entire system will have to be redesigned.

Some of the problems that arise in defining a common model and DML are caused by the many conflicts that may exist among the DBMS's. For example, similar information may be kept in relations† with different names, or the reverse may be true, i.e., perhaps relations with identical names have quite distinct semantics. Another type of conflict occurs if the tuples of equivalent relations use different units (such as feet in one and meters in the other). These types of conflicts are fairly easy to resolve, but other conflicts are structural and much more difficult to handle. For example, the branch of a corporation may keep an inventory of office equipment for each building at its site, while the parent company keeps such information on a departmental basis. As we will describe later, our approach for dealing with these occurrences is to encode the information required to resolve potential conflicts into an expert system. We also expect that the power of an expert system will allow us to handle situations where two or more databases have inconsistent information, as well as the cases where the data contained in all the databases does not completely specify the answer to a user's query.

Once the global scheme is defined, the issue of query processing comes into play. By this we mean that, if a user asks for a query of the combined database, that query must be broken up into a set of sub-queries that will be executed on the different DBMS's. There are two research problems here: the first is how do we break up the query, the other how do we make sure this decomposition process results in efficient query execution. Initially, we will concentrate on addressing the first issue only. It is clear that to devise a decomposition of the global query the HDBMS must have the appropriate information to make the correct decisions. This information is kept in the *auxiliary database.* Since this information is related to that required to resolve the conflicts described above, currently we are also planning to integrate the auxiliary database information into the expert system previously mentioned.

There are many other issues involved in the design of HDBMS's. We have only outlined a few in this section. For a more extensive commentary of the issues described in this section, the interested reader is referred to Chapter 15 in [Ceri84].

---

† At this point it should be noted that throughout we will assume that all the DBMS's, although dissimilar, will be relational ones.

3. Proposed System Architecture

In this section we sketch the architecture of an HDBMS which we are currently implementing. The overall structure of the system is not radically different from other available ones. Rather, it is the use of an expert system to subsume the information contained in the auxiliary database previously described which sets our work apart from earlier approaches.

In our architecture, each of the separate DBMS's will continue to handle requests that deal solely with the "local" data. (We will refer to those DBMS's as the local DBMS or LDBMS for short.) Thus, if a user does not have any need to access information spanning more than one DBMS he will not pay any overhead, and is able to use the data manipulation language (DML) of the appropriate LDBMS. (If the user is unsure whether the query requires only single site data, or is more comfortable using the global DML, he may submit that query as if it were a multiple database one; the query will then be processed in a manner analogous to that described below.)

Whenever a user executes a query that involves the joint data, that query will be passed to a global database management system (GDBMS). In the simplest of cases, the job of the GDBMS will be to cleanly decompose the global query into sub-queries (one per LDBMS), each posed in the actual data manipulation language of their respective LDBMS. When both LDBMS's reply with the answer tuples, all the data will be stored in a temporary relation, perhaps after appropriately modifying the returned tuples. Then the GDBMS will present the answer to the waiting user (perhaps after some further modification of the temporary relation).

To be more precise, consider the case where a user wants to know the salaries of the employees that work in the computer research departments of two entities. That query (in the DML of the GDBMS) is translated into two queries, each in the DML of the LDBMS's, asking for the salaries of the employees working in computer science research according to each local database. When both queries are answered, the GDBMS puts all the data together. If there are some minor differences in the tuple format, the GDBMS will correct them at this point; for example, if the employee name fields happen to be of different sizes, the GDBML will expand the size of the tuples with the smaller size, and right-fill with blanks. When the GDBMS is done, it may perform some final processing to discard duplicates (maybe some employees work for both organizations), and presents the user with the result.

Of course, the situation may be much more complex than in the simple case described above; for example, the computer science research department may go by different names in the two companies, or it may not exist as a separate entity in one of the organizations. This is one example of the class of database schema incompatibilities that we described in the previous section. Our strategy for integrating the database schemas is to make available to the GDBMS an expert system that can be used to resolve conflicts. For example, in the case where the two departments have different labels, the expert system can easily point this out, and the GDBMS can then account for this fact while forming each local sub-query, as well as while preparing the final output for user

viewing. If computer science researchers work within some larger department (say Electrical Engineering and Computer Science), the expert system can then inform the GDBMS of this, as well as of what criteria could be used to pinpoint the correct employees (for example, that employees work within the EECS department and that the phrases "computer science" and "research" appear in their job description).

To complicate the situation further, suppose that a consultant happens to work for both companies, and receives a different compensation from each. After the GDBMS collects the tuples from the local queries executed by the LDBMS's, it is faced with an inconsistency in the data it has. The expert system could then suggest a number of approaches, perhaps to average the two salaries or to list the individual twice. For more sophisticated users, a better approach might be to let them know that there is an inconsistency in the data; another possible heuristic here might be to provide the user with a partial answer (and advise him of this), and automatically notify the global database administrator that perhaps the expert system's information should be enhanced to provide help for this kind of global query.

It should be clear from the comments above that we expect quite a bit of functionality from our expert system, since it will have to embody the heuristics we deem appropriate to integrate the different data models of the LDBMS's into the data model of the GDBMS. We also expect that this expert system will be used not only for run-time global query decomposition, but also to aid database administrators in designing an integrated global data model. Whether such an expert system is possible to construct, and whether it is an efficient method for database integration are two of the most important issues our research will address.

4. Prototype Design

We expect to build an HDBMS following the architecture described above by using the facilities of the Princeton Distributed Computing Laboratory. This laboratory consists of a wide assortment of computers (Sun workstations, DEC Vaxes, IBM PC's, NCR Towers, AT&T 3B2's, etc) connected by an Ethernet. Our section of the network can be physically disconnected from the departmental network, which will ensure a quiescent network for performance measurements.

Our plans presently call for the HDBMS software to be developed on both SUN workstations and IBM PC's. We expect that the initial HDBMS will bridge an INGRES database residing on a Sun and a DBASE DBMS running on an IBM PC. The communication between the two databases will be carried out in two ways. First, by using the TCP/IP communication protocols, but also by making use of SUN Microsystems PC NFS software, which allows PC users access to files residing at SUN workstations.

The expert system will be developed using Prolog. We feel that this language will be appropriate for this task, and is an especially adequate tool for building system prototypes. Some of the members of our project have already built an object-based query language using Prolog to develop expertise with

that language in system applications.

5. Conclusions and Future Work

Although we are not aware of any such attempts in the available literature, we feel that the use of an expert system can be of great help in resolving the conflicts inherent in data integration. By actually implementing a heterogeneous database system we expect to provide our ideas with a reality check that is not possible from a purely paper design.

We plan our work to proceed along three distinct directions, the first two of which we are already exploring. The first is to consider a set of possible compatibility problems (such as those discussed above), and try to devise solutions (or, more likely, reasonable heuristics) for them, and attempt to capture the ideas behind the solutions in a simple Prolog-based expert system. The second aspect of our work will focus on choosing examples of database applications in order to guide our selection of data modeling languages (DML's). We will explore the usefulness of existing languages, and determine if they are suitable, or if extensions (such as object-oriented descriptions) need to be developed.

Finally, after the issues have been explored to our satisfaction in the two initial phases of our work, we will start on the implementation effort described in the previous section. In this phase (as well as in the previous two aspects of the work just described), we expect to collaborate closely with members of the technical staff of SRI's David Sarnoff Laboratory.

6. References

[Ceri84]
S. Ceri and G. Pelagatti, "DISTRIBUTED DATABASES Principles and Systems," McGraw-Hill, 1984.

[Date75]
C. J. Date, "An Introduction To Database Systems," Addison-Wesley, 1975.

[Dayal84]
U. Dayal and H. Y. Hwang, "View Definition and Generalization for Database Integration in a Multidatabase System," *IEEE Transactions on Software Engineering*, November 1984.

[Motro87]
A. Motro, "Superviews: Virtual Integration of Multiple Databases," *IEEE Transactions on Software Engineering*, July 1987.

[Smith77]
J. M. Smith and D. C. P. Smith, "Database Abstractions: Aggregation and Generalization," *ACM Transactions on Database Systems*, June 1977.

[Stonebraker77]
M. Stonebraker and E. Neuhold, "A Distributed Database Version of INGRES," *Proceedings 1977 Berkeley Workshop on Distributed Data Management and Computer Networks*, 1977.

[Williams82]
R. Williams et al, "R*: An Overview of the Architecture," *Proceedings of the International Conference on Databases*, Israel, June 1982.