ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

Rainer Manthey and Viacheslav Wolfengagen (Eds)

Advances in Databases and Information Systems 1997

Proceedings of the First East-European Symposium on Advances in Databases and Information Systems, (ADBIS'97), St Petersburg, 2-5 September 1997

Supporting Interoperability of Autonomous Hospital Databases: A Case Study

A. Zisman and J. Kramer

Published in collaboration with the
British Computer Society

©Copyright in this paper belongs to the author(s)

ISBN: 3-540-76227-2

Supporting Interoperability of Autonomous Hospital Databases: A Case Study

A. Zisman*

az@doc.ic.ac.uk

J. Kramer

jk@doc.ic.ac.uk

Department of Computing - Imperial College

180 Queen's Gate - London - SW7 2BZ - UK

Phone: +44-171-5948239 Fax: +44-171-5818024

Abstract

We describe an approach that supports distributed information discovery when interoperating with large number of autonomous hospital databases. The approach is presented through a case study based on the health care network. The idea is to perform complete execution of a query: data request, database location and data access, in a distributed way, avoiding the use of integrated schemas and centralised structures. The discovery process is limited to a group of databases that have either related data or knowledge about other databases that possibly holds the required data. The process is recursive, executed in parallel and avoids cycles. The approach permits dynamic evolution of the system and attempts to preserve the autonomy of the databases. A prototype has been developed for academic and research purposes. It demonstrates the feasibility of the approach as an alternative to interoperate with a large number of autonomous databases without using integrated schemas.

1 Introduction

The ability to share and exchange data between (legacy) database systems is an important problem currently faced by many large organisations. Research is necessary to achieve interoperability so as to evolve from the situation where single databases are accessed and manipulated in isolation to the case where it is possible to access simultaneously different databases (local and remote access). The difficulties associated with interoperability between large scale databases include heterogeneity and autonomy of the databases, conflict resolution and semantic representation of the data, location and identification of relevant information, access of remote data, and easy evolution of the system.

An example of the interoperability problem is found in the health care network, where each hospital, or even each department in a hospital, keeps its own database. In this environment it is very important to permit users to locate and access data from several remote databases, supporting the needs of patient care, daily operations of the hospitals and research consultations. This thereby supports the sharing and exchange of data related to clinical, administrative, managerial and research (statistical) information. Examples of these situations are illustrated as follows: (1) a gynaecologist wants to know the records (the last 7 months history) of patient X who is in her 8th month of pregnancy; (2) an administrator wants to know about the availability of operating theatres on a certain day/time to arrange a surgery; (3) a fellow student wishes to know statistics information about successful heart transplants in patients between 40 and 50 years old.

The location and identification of information that is relevant, identical, similar and related to requested data (*information discovery*) is an important issue when dealing with interoperability of autonomous databases. Existing approaches assume that the system either knows the location of the requested data, or uses centralised

*This work has been supported by Brazilian foundation CNPq, grant no. 200209/94-9.

structures with information about the available data and where it is located. In an environment composed of databases that are created and administered independently, in which each hospital or department keeps its own patient records, it is difficult and impractical to build integrated schemas. The use of a centralised structure with information about the location of data can be a performance bottleneck. Therefore, it is necessary to propose an approach that permits interoperability of these databases.

In this paper we demonstrate, through a case study, the use of the approach proposed in previous work [18, 20, 21, 22]. The approach permits distributed information discovery when interoperating with various hospital databases. Our aim is to allow “naive” users (doctors, nurses, fellows, members of the hospital staff) to access and manipulate local and remote data. The approach is an alternative to the problem of building an integrated global schema when dealing with a large number of databases. The idea is to perform the complete execution of a query: data request, database location and data access, in a distributed way, avoiding the use of integrated schemas, centralised structures and broadcast to all the databases in the system. The approach attempts to preserve the autonomy of the hospital databases and supports evolution of the system in terms of adding or removing databases.

The case study is based on data obtained from real databases of different hospitals in Britain. A prototype has been built for academic and research purposes. Although it has not been tested in real situations and does not contain a large number of databases, it is sufficient to show the feasibility of the approach. It has demonstrated the possibility of the execution of various queries and the access of data in remote databases. It also indicates that the discovery process terminates and that the approach supports evolution of the system. Experience indicates that it is easier to use this approach than to build a global schema composed of different existing schemas. It has also shown that the autonomy of the databases is preserved.

The paper addresses the setting up of the system, the location, identification and access of data, and the evolution of the system for a large number of hospital databases. The remainder of the paper is structured as follows. Section 2 contains an overview of the system, with a description of the different parts of the architecture which assist with the operation of the system. In section 3 we present how to setup the system. Section 4 contains the description of the use of the system, where we illustrate the complete execution of a request. In section 5 we present the evolution of the system. Section 6 contains some implementation experience and evaluation of the prototype. In section 7 we present an overview of related work. Finally, section 8 summarises the results and suggests future work.

2 System Overview

The case study is composed of three hospitals wishing to share and exchange data between each other. *Hospital-A* is a general hospital composed of relational multidatabases where each department of the hospital has its own database. Figure 1 presents the various departmental databases of hospital-A. *Hospital-B* is a child care hospital formed by a centralised relational database (DB_{HB}). *Hospital-C* is a general hospital composed of a centralised object-oriented (O2) database (DB_{HC}).

To support interoperability of the different databases we suggest the use of the architecture proposed in previous work [18, 21, 22]. The architecture supports a distributed information discovery process [20] and access and manipulation of local and remote data.

We organise the participating databases in the system into *federations*. A federation consists of a set of databases that wish to share and exchange data with each other. It is formed based on the shared data (*public schema*) of the databases and the databases that are allowed to access this data. This public schema is a subset of the local schema of the database. A database can participate in more than one federation (overlapping), sharing a distinct set of data in each of these federations. There is no communication between two federations. Inside a federation a set of shared data from one database is public for all the other participants in the federation. The idea of the federations is to provide a certain privacy of the shared data and to avoid using negotiation strategies between databases that are exchanging data. All the databases consulted during a discovery process belong to the same federation and can exchange its set of shared data, related to this federation, with each other.

Inside a federation we propose a second level of organisation, classifying the databases into *groups*. These

Figure 1: Various databases participating in hospital-A

Figure 2: Federation 1 with its groups

groups are used to facilitate the information discovery process by reducing the universe of search. The groups are defined based on the type of data shared by the databases, i.e. ‘context classification’. In one federation it is possible to have a database participating in more than one group (overlapping). Each federation has a *coordinator* that assists with the formation and evolution of the federation.

Examples of some federations used in the case study, with its respective groups, are presented in figures 2, 3 and 4. Federation 1 is created to permit the exchange of data related to children between hospital-A and hospital-B. Federation 2 is composed of all the departmental databases of hospital-A, allowing the share of data inside the whole structure. Federation 3 is meant to let the share of data related to transplants performed in hospital-A and hospital-B.

In order to assist with the location of the data we use an unbalanced hierarchical structure. Each database has a different hierarchical structure for each federation in which it participates. This structure is composed of the names of the groups of the federation and *specialised terms*, forming a natural hierarchy of names. Associated with these terms are references to the concerned databases. Figures 5 and 6 present the hierarchical structure for database DB_{HB} in federation 1 and federation 3, respectively. Experience indicates that it is possible to use this structure to identify the databases related to a request.

Figure 3: Federation 2 with its groups

Figure 4: Federation 3 with its groups

Figure 5: Hierarchical structure of DB_{HB} in federation 1

Defining the terms that compose the hierarchy requires human assistance. This is provided by the *coordinator* of the federation. We have built a methodology and a tool to help with the construction and evolution of the hierarchy [19]. The idea is to define the terms based on the interests of the users and applications, and information that each database is sharing with the system (schema and instances). Therefore, the different group names refer to the type of databases participating in the federation. The other levels are related to the entities, attributes and object names, and instances of the databases. The methodology and tool permit the hierarchy to be built and to evolve by incremental addition of the participating databases, where many steps are executed automatically. The coordinator is responsible to specify the type of the participating databases and to identify the part of the schemas and instances used to compose the hierarchical structures. The tool is responsible for proposing the different groups composing a federation, with its related specialisations. The coordinator validates the suggestions of the tool and propose some changes, if necessary. The hierarchical structure evolves as a consequence of system modification (addition and removal of databases) and as needed to adjust the structure to the requirements of the users and applications. This is done based on statistical information related to the use of the different branches of the hierarchy, and on recommendations from users when they identify the absence of terms matching their request.

For each group in a federation we suggest the use of a special structure called *master* (M_{Gx}). The master of a group is the hierarchical structure described above containing the most up to date information about the organisation of the related federation. The idea of using the master is to permit the system to evolve without having to stop ongoing request executions in the system. The changes in the system such as: addition and removal of databases; creation, removal and modifications of the groups; insertion and deletion of federations are first notified to the masters of related groups. Subsequently, the hierarchical structures of the related databases are gradually updated (cf. section 5). Thus, inside a federation the hierarchical structures of the participating databases can contain different information.

Each database in the system contains a special structure called *FED* with information about the different federations in which the database participates. Figure 7 contains the FED structure of database $DB_{C\&P}$ of hospital-A. The idea of using this structure is to identify the related federation where the search for a certain data should be executed. For example, it does not make sense to perform the search of data related to heart transplants executed in patients older than 35 years in federation 1.

We suggest to use a replicated relational database named *Syst-DB* with general information about the

Figure 6: Hierarchical structure of DB_{HB} in federation 3

different federations, groups and participating databases of the system. This structure is consulted to identify the correct federation to add a new database.

A major issue when dealing with interoperability of autonomous databases without using integrated schemas is related to the specification of the queries to be executed in a remote database. It is impossible for a user to execute remote queries without knowledge of the schemas of the remote databases. As a solution for this problem we suggest the use of a graphical interface that assists the user to compose the query related to a request. The user is responsible to select and complete options in a template. The options presented to the user are associated with the terms in the hierarchy that are relevant to the request of the user. This general query is translated to the specific schema/languages of the remote accessed databases. An example is presented in figure 8 where a user from database $DB_{C\&P}$ in federation 3 is interested in *successful kidney transplants performed in patients younger than 15 years*. The different options presented in the graphical interface related to a certain request are based on the schema of the different databases that possibly contain the requested data. However, we affirm that this is not a global integrated schema formed by the integration of the different schemas as proposed in some existing approaches [1, 4, 16, 17].

3 System Setup

The system setup is performed when two or more databases want to share and exchange data between each other. In the first step the DBA of the involved databases determines the set(s) of data of each database to be shared and specifies the type of data being shared. Afterwards, a coordinator is elected to assist with the setup of the system.

Federation	Information Type
1	Children
2	Hospital-A
3	Transplants

Figure 7: FED structure of database $DB_{C\&P}$

PATIENT

HISTORY

SURGERY

...

STAFF

name: ?

date: ?

.

.

address:

time: ?

.

.

age: <= 15

treatment: ?

.

.

sex:

situation: successful

.

.

nationality:

doctor: ?

.

.

blood-type:

description: ?

.

.

RH:

.

.

.

Figure 8: Part of the graphical interface for federation 3

Based on the set of shared data, on the databases allowed to access this data and on the type of information in which the users are interested, the coordinator determines the various related federations. Notice that it is only necessary to have one federation to start the system. The other federations can be added incrementally to the system during its operation. When more than one federation is specified it is essential to have one coordinator for each federation. In each identified federation it is necessary to determine the different groups with its associated terms. This is achieved using the methodology and tool explained above.

After organising the databases into the different groups the coordinator identifies the masters associated to each group and the hierarchical structures are defined. During the creation of the system all the hierarchical structures of a certain federation contain the same information.

The next step consist of determining the graphical interface. Subsequently, the general information of the system is inserted in the Syst-DB, the FED structures of the related databases are constructed and the system can start to be used.

For the case study, let us assume two situations. First, the users of the databases of hospital-A want to share data among the different departments of the hospital. Second, the users of the databases of hospital-A and hospital-B wish to access and share information related to children and transplants. Therefore, the DBA of each of these databases determine the different sets of data to be shared. The federations presented in figures 2, 3 and 4 are constructed together with their related hierarchical structures shown in figures 5, 9, and 6.

4 System Use - Discovery Process

The discovery process is executed in parallel in databases that either contain the requested information or know about other databases that possibly hold the required data. A detailed description of the process, together with the correctness proof of the algorithm and some complexities measurements, can be found in [20].

Consider the situation where a doctor from hospital-A, working at the Cardio-Pneumology department ($DB_{C\&P}$), wants to know about *successful kidney transplants performed in patients younger than 15 years*. The

Figure 9: Hierarchical structure of federation 2

first step consists of identifying the federation in which the database $DB_{C\&P}$ participates that is related to the request. In this case, the system presents to the doctor the information shown in figure 7, allowing him/her to choose the correct federation. The doctor identifies federation 3 as the related federation to perform the search.

After identifying the federation where the process will be executed, the user browses the local hierarchical structure of database $DB_{C\&P}$ related to federation 3, in order to determine the group of databases to be searched, the path of terms in the hierarchy related to the request and the respective databases that possibly contain the related data. Assume the hierarchy of $DB_{C\&P}$ is as shown in figure 6. For the current request the user identifies *Renal* group, the terms: *history* and *children* and the databases DB_{TRA} , DB_{HB} and $DB_{R\&H}$. The idea of browsing the hierarchical structure the first time is to facilitate data location. Thus, we avoid the problem of keywords matching between the user input and the terms in the hierarchy. Keywords matching normally requires the use of dictionaries and depends on the order that the terms appear in the hierarchy.

Based on the specified group and terms the system shows the associated interface and the user composes the query q' as presented in figure 8. The query q' , the address of the source component ($DB_{C\&P}$) and the related terms in the hierarchy constitute the request r' . The request is sent in parallel to databases DB_{TRA} , DB_{HB} and $DB_{R\&H}$. After receiving r' , each database is consulted to verify if it contains the requested information. This is done by translating q' to the specific language/schema of the database.

Suppose that database DB_{TRA} does not contain the data related to the request. The process is recursive and the hierarchical structure of DB_{TRA} , related to federation 3, is automatically traversed. The hierarchy is traversed based on the terms that compose the request r' . This is executed to allow the identification of other related databases not presented in the hierarchy of the source component. It is possible that the terms in the hierarchy of DB_{TRA} differ from the terms in the hierarchy of $DB_{C\&P}$. However, we guarantee that all the hierarchical structures of a federation contain the same terms related to the various group names. Assume the hierarchical structure of DB_{TRA} the same as presented in figure 6. DB_{TRA} does not contain the requested information and does not know about any other database participating in federation 3 that is related to the request and has not been visited. Therefore, we propose to send the request to the master M_{G12} of the *Renal* group.

Consider the case where DB_{HB} receives the request r' , it does not contain the related information, but it knows about a 'new' related database. That is, a database that is not presented in the associated branch of the hierarchy of $DB_{C\&P}$. Figure 10 presents part of the hierarchy of DB_{HB} , related to federation 3, with the 'new' database DB_{HC} . In this case, r' is sent to DB_{TRA} , $DB_{R\&H}$ and DB_{HC} .

Let us assume that $DB_{R\&H}$ contains data related to r' . The related data is sent to the source database to be evaluated by the user. In the event where the user is satisfied with the data, the discovery process has to be cancelled, since it is still being executed for the other databases. Suppose that the user is not satisfied with the received data and the process continues normally. The hierarchical structure of $DB_{R\&H}$ is consulted in order to identify other related databases.

When the master M_{G12} receives the request r' it tries to identify 'new' related databases recently added to the system and not yet visited. Consider part of the hierarchy of M_{G12} as shown in figure 11, where M_{G12} contains extra information about database DB' . The request is sent to DB' .

Due to the fact that the process is executed in parallel, when a database receives a request it does not know about the other databases that have received this request before. However, the discovery process tries to prevent cycles, that is, the access to a database more than once for the same request. This is executed by maintaining, in each participating database, for each discovery process, information about the associated request. Hence, whenever a database receives a request it verifies if it has received this request before. If so, it sends a *negative reply* to the last sender and ignores the request. For instance, when databases DB_{TRA} and $DB_{R\&H}$ receive the request r' from database DB_{HB} , they ignore this request and send a *negative reply* to DB_{HB} .

Consider that DB_{HC} does not have the requested information and does not know about any other database not accessed for r' . In this case, DB_{HC} sends a *not-existing message* to DB_{HB} denoting that it has finished the search process for r' . The same happens with $DB_{R\&H}$ that sends a not-existing message to $DB_{C\&P}$. When a database receives negative replies and not-existing messages from all databases for where it have sent the

Figure 10: Part of the hierarchy structure of DB_{HB}

Figure 11: Part of the hierarchy structure of M_{G12}

Figure 12: Graph of the messages sent during the execution of the discovery process of r'

request, this database sends a not-existing message to the first database that had sent it the request. This process continues until reaching the source component. As a result, the source database concludes about the non existence of the requested data in the system. Thus, after DB_{HB} receives the not-existing message from DB_{HC} and negative replies from DB_{TRA} , $DB_{R\&H}$ and M_{G12} , it sends a not-existing message to $DB_{C\&P}$.

Suppose that DB' does not contain the requested data. DB' sends a notification to M_{G12} , that sends it to DB_{TRA} . Figure 12 presents a graph related to the scenario being presented, with the different types of messages sent during the execution of r' .

5 System Evolution

The proposed approach permits dynamic evolution of the system. It allows the addition and removal processes to be executed in an interactive way, without having to stop the whole system. The idea is to first update 'strategic points', the masters of the related federation, and gradually notify the other participating databases, by updating its hierarchical structures. A database (or group) has autonomy to decide whether it wants to participate in the system. The decision is very decentralised, not requesting the approval of a central authority or of all the databases in the system. However, when a database is added into a federation the other databases in this federation have to authorise its participation. The existing databases have to agree about sharing and exchanging data with the 'new' database. In order to support the evolution of the system we use the *Syst-DB*, together with the assistance of DBAs and coordinators of the involved databases and federations.

5.1 Addition

The system supports the addition of a database, a group of databases and a federation. The addition of a 'new' database refers to the inclusion of this component into a federation. The database can already exist in the system, but participating in different federations. The addition of a single database can cause either the inclusion of a group inside the related federation, or the creation of a new federation.

Suppose the situation where the database of hospital-C (DB_{HC}) wants to participate in the system. Assume that the users of DB_{HC} are interested in sharing and exchanging information related to *transplant* and *cancer*.

Figure 13: Up to date hierarchical structure of federation 3

The first step consists of identifying the ‘correct place’ to execute the addition of DB_{HC} . Therefore, the DBA of DB_{HC} consults the Syst-DB.

For the ‘transplant’ case, the DBA identifies federation 3 as a possible federation to insert DB_{HC} . S/He consults the DBAs of the other databases participating in federation 3 (DB_{HB} , DB_{OPH} , DB_{TRA} , $DB_{C\&P}$ and $DB_{R\&H}$) to authorise the insertion of DB_{HC} . Let us assume that the databases in federation 3 agree with the insertion of DB_{HC} . The addition of DB_{HC} causes modifications in the existing hierarchical structures related to federation 3, by creating a new group and updating existing ones. The modification of the hierarchy is assisted by the coordinator, tool and methodology. Figure 13 contains the modified part of the hierarchy for federation 3. The next steps consist of setting DB_{HC} and the master of the newly created group (*Research group*) available to the system, followed by updating the existing hierarchical structures with the modifications. This is performed gradually. First the masters of the groups into which DB_{HC} was inserted (*Ophthalmology* and *Renal*) are updated. Afterwards, the other masters of federation 3 are updated. Each master is responsible for updating the hierarchical structures of the databases participating in their respective groups. A new database exists for the system (a federation) whenever the masters of its associated groups have information about it. This is possible due to the way that the discovery process is executed. That is, by consulting the master of a related group when other accessed databases do not contain the data. When a new group is created all of the masters in the related federation are updated together. This guarantees the access of the databases in this new group during the discovery process. The hierarchical structures are gradually updated in the same way as explained above. A new group exists in the system when all of the masters in the related federation have information about it. Finally, the Syst-DB and the FED structure related to DB_{HC} are updated.

In the ‘cancer’ situation the DBA concludes that there is no federation in the system suitable to perform the addition of DB_{HC} . However, some databases are identified as candidate to compose a new federation with DB_{HC} and share information related to cancer. Federation 4 is created with databases DB_{HC} , DB_{HB} , DB_{CAN} and DB_{CHI} , as presented in figure 14. The initial hierarchical structure of the participating databases and masters is shown in figure 15. Notice that the creation of a federation can also occur when a suitable existing federation is identified, but not all the participating databases in this federation agree with the addition of the new database. The construction of a new federation, either spontaneously or as a result of adding database(s) to the system is executed in a similar way to the setup of the system. That is, a coordinator is elected, the hierarchical structures are constructed, the masters of the different groups are specified, and the Syst-DB and the FED structures are updated.

Figure 14: Federation 4 with its groups

Figure 15: Hierarchical structure of federation 4

Figure 16: Modified part of the hierarchical structure of federation 1

5.2 Removal

The removal of a database does not mean its deletion from the system, but the elimination of this database from one of the federations in which it participates. Similar to the addition case, the update of the hierarchical structures about the removal of a database or a group of databases is gradually performed. Therefore, it is possible to have incorrect information in the different hierarchical structures for a certain period of time. That is, during a discovery process a ‘non-existent’ database may be identified as a candidate to contain the requested data. This is acceptable in order to reduce the number of messages in the system and to avoid stopping the whole system to execute the updates. Hence, when a request is sent to a ‘removed’ database, the requester receives a ‘special notification’ as a reply denoting the non participation of the database in that federation anymore.

The removal of databases from a federation can cause modifications in the hierarchical structures such as: removal of some groups and terms, generalisation of other terms and combination of other groups. These tasks are executed with the assistance of the coordinator and the tool. Consider the removal of database DB_{LAB} from federation 1. The modified part of the hierarchical structure related to federation 1 is presented in figure 16. The Syst-DB and the FED structure of DB_{LAB} are updated. Afterwards, the related masters are notified and the hierarchical structures of the other participating databases are updated.

The removal of a federation from the system can occur either because its participating databases do not want to share and exchange data among each other, or because of decremental deletions of its databases, resulting in a federation with only one database. The procedure consists of updating the Syst-DB and the FED structures of the related databases followed by removing the hierarchical structures of the databases associated to the federation.

6 Implementation Experience and Evaluation

An experimental prototype of our proposal is currently under development. The initial prototype is being used to demonstrate, evaluate and refine our approach. It is implemented in C++, using Darwin/Regis [9] environment to allow the distributed exchange of data and messages. The tool and methodology to assist with the creation, modification and evolution of the hierarchical structures are implemented in Java.

The prototype assists with the system setup, the concurrent execution of the discovery process for various requests, together with the dynamic evolution of the system (addition and removal of databases). We are currently developing the graphical interface that allows the user to execute the query. The prototype is small and simple. However, it demonstrates the feasibility of the approach as an alternative to the problem of interoperating with a large number of databases without using integrated schemas. It shows that it is not difficult to find and access the databases in the system, even for the components that have recently joined the system. Access to requested data is performed as part of the distributed discovery process. The prototype supports discovery and access to one type of information. It does not deal with data that is partitioned among several databases. Our work has also denoted that the discovery algorithm terminates and the approach supports evolution of the system.

Our experience indicates that it is easier to build the hierarchical structure, determine its terms and specify the different options of the graphical interface, than to build an integrated global schema. The construction of a global schema is not a simple task. It has to cope with difficulties such as: semantic and syntactic conflicts,

equivalence and ambiguity of data, and dissimilarities in modelling and viewing aspects of the real world. The use of a global schema does not guarantee the autonomy of the different databases and does not allow the dynamic evolution of the system to be executed in a simple way. That is, any modification in the system has to be reflected into the global schema causing alterations and reconsideration of many design decisions. We affirm that the definition of the different sets of data shared by a database, the idea of federations, the use of the hierarchical structures together with the terms, and the graphical interface support the preservation of the autonomy of the participating databases.

In the proposed approach the number of transmitted messages during the discovery process of a request is large. However, this number can be reduced as suggested in [20]. The large number of messages is related to the fact that the process is executed in parallel, there is no global schema and the dynamic evolution of the system is executed concurrently with system use. We need to perform some simulations to measure the average time necessary to locate data and the average number of messages sent during the process.

The proposed approach requires human assistance in the execution of some tasks, such as: setup of the system, creation and evolution of the hierarchical structures, specification of the graphical interface, definition of the federation where to perform the discovery process and to add a database, formulation and validation of the request/query. However, when using an integrated schema it is also necessary to have human help during the construction and evolution of the schema.

We decided to use a hierarchical structure to assist with the location of data, due to the fact that such a structure naturally reflects the way that a person tries to search for information, i.e. from general to specific. In addition, a hierarchy supports the organisation of grouping related objects together. A position in a hierarchy implies some context. The terms composing a path in the hierarchy have more meaning than an end single term. This fact helps to perform the queries that are sent to the remote databases, facilitating the access of the data in these components. A hierarchy supports scalability and evolution. That is, it allows repetition of information in the structure giving more flexibility to find the databases related to the request. Inside a hierarchy it is possible to have different ways of organising its terms. As a result, it is difficult to choose and to decide about the combinations of appropriate keywords. However, in our approach, the decision of the terms is assisted by the schemas and instances of the involved databases, and the interest of the users. We also allow certain level of redundancy in the hierarchy. Another drawback when using a hierarchical structure is related to the limitation of searching. The search depends on the existing terms and on the way (order) that the terms are organised. We address this problem by allowing the user to browse the hierarchical structure of the source component when s/he is executing a request. Therefore, the user can be familiar to the existing terms in the hierarchy and the order that they appear. This strategy also prevents the use of dictionaries, thesaurus and context indices to execute the keyword matching between the user input and the existing terms. In addition, experience indicates that the hierarchical structures have three levels in average, allowing a good performance when traversing the hierarchy.

Apart from the definition of the terms that compose the hierarchy, another issue in the development of the proposed approach is related to the specification of the query that allows the access of the data in remote databases. It is impossible for the user to perform specific remote queries without knowing about the schemas of the remote databases. The use of the graphical interface is a promising way to solve the problem. However, it is necessary to perform more experiments to validate the approach. One more important and complex issue that will be a topic of future work is related to the access of requested data that is partitioned among several databases.

7 Related Work

Some approaches have been proposed in the literature to address the information discovery problem for a large number of databases [2, 3, 6, 10, 11, 12, 14]. These approaches present a way of identifying the content of the different databases, but do not specify how to access the data after locating the database. In our approach we specify the complete execution of a query (data request, database location and data access). The requested data is accessed as part of the discovery process. In [3, 10, 11] the authors proposed a framework that allows dynamic

education of the users about the available information space. This is executed in a simple way by supporting interaction and identification of inter-relationships between databases. Each participating database contains an object-oriented *co-database* with information about coalitions and services that the database is involved. The inter-node relationships are based on ‘context abstraction’, defining objects using *Global Concepts*. The terms used in the hierarchical structures of our approach share some similarities with these Global Concepts. The approach entails high interaction with the user to build the co-databases and execute a search. The search mechanism proposed in [6, 14] performs the location of relevant databases with the help of a meta knowledge-base (MKB). The approach uses natural language to execute the discovery of related databases. This provides a user-friendly interface to the users, but requires some additional structures such as: context and part of speech indices, dictionary and thesaurus. In [12] the approach uses intelligent agents to assist with the discovery process. Each database has an agent that is responsible to receive high-level queries and translate them into a database specific one. The approach proposed in [2, 15] relies on the use of an external indexing scheme that does not respect the autonomy of the involved databases. The databases are interdependent and changes in one tend to affect another.

8 Conclusion and Future Work

In this paper we demonstrate the feasibility of using a distributed information discovery process that supports the sharing and exchange of data between ‘naive’ users, in a number of autonomous hospital databases. Unlike [5, 7, 8, 13], the approach aims to avoid the use of integrated schemas and centralised structures. It also permits dynamic evolution of the system. The process is assisted by dynamic hierarchical structures composed of special terms. These terms are used to identify the databases related to a query. The idea is to limit the discovery process to a group of databases that have either related data or knowledge about other databases that possibly hold the required data. Accessed databases are able to share data with the requester and among each other. The process is recursive, executed in parallel and avoids cycles.

We are currently developing the graphical interface that permits the execution of the queries and allows data access. We are also exploring the idea of expanding the approach to support the execution of queries where the data is partitioned among several databases. Before large scale experimentation and use we plan to execute more tests and compare the proposal with existing approaches.

References

- [1] R. Ahmed, P. De Smedt, W. Du, W. Kent, M.A. Ketabchi, W. A. Litwin, A. Rafii, and M. Shan. The Pegasus heterogeneous multidatabase system. *Computer*, 24(12):19–27, December 1991.
- [2] R. Alonso, D. Barbara, and S. Cohn. Data sharing in a large heterogeneous environment. In *7th International Conference on Data Engineering*, pages 305–313, Held Kobe, April 1991. IEEE.
- [3] A. Bouguettaya, S. Milliner, and R. King. Resource location in large scale heterogeneous and autonomous databases. *Journal of Intelligent Information System*, 5(2), 1995.
- [4] C. Chung. DATAPLEX: An access to heterogeneous distributed databases. *Communications of the ACM*, 33(1):70–80, January 1990.
- [5] D. Fang, J. Hammer, and D. McLeod. A mechanism and experimental system for function-based sharing in federated databases. In E.J. Neuhold D.K. Hsiao and R. Sacks Davis, editors, *Interoperable Database Systems (DS-5)*, pages 239–253. Elsevier Science Publisher B.V., 1993.
- [6] R. Giladi and P. Shoval. An architecture of an intelligent system for routing user requests in a network of heterogeneous databases. *Journal of Intelligent Information Systems*, 3:205–219, 1994.

- [7] J. Hammer and D. McLeod. An approach to resolving semantic heterogeneity in a federation of autonomous, heterogeneous database systems. *International Journal of Intelligent and Cooperative Information Systems*, 2(1):51–83, 1993.
- [8] D. Heimbigner and D. McLeod. A federated architecture for information management. *ACM Transaction on Office Information Systems*, 3(3):253–278, July 1985.
- [9] J. Magee, N. Dulay, and J. Kramer. Regis: A constructive development environment for distributed programs. *Distributed Systems Engineering Journal*, 1(5):304–302, 1994.
- [10] S. Milliner, A. Bouguettaya, and M. Papazoglou. A scalable architecture for autonomous heterogeneous database interactions. In *Proceedings of the 21st International Conference on Very Large Data Bases*, pages 515–526, Zurich, Switzerland, 1995.
- [11] S. Milliner and M. Papazoglou. Reassessing the roles of negotiation and contracting for interoperable databases. *Int'l Workshop on Advances in Databases and Information Systems, Russian ACM SIGMOD*, May 1994.
- [12] V. Ramesh, K. Canfield, S. Quirologico, and M. Silva. An intelligent agent-based architecture for interoperability among heterogeneous medical databases. In *Americas Conference on Information Systems*, Association for Information Systems, pages 549–551, Phoenix, Arizona, August 1996.
- [13] A. P. Sheth. Semantic issues in multidatabase systems. *SIGMOD RECORD*, 20(4):5–9, December 1991.
- [14] P. Shoval, P. Ein-Dor, R. Giladi, and I. Spiegler. A meta knowledge base and a search mechanism for distribute, heterogeneous databases. In *Americas Conference on Information Systems*, Association for Information Systems, pages 617–619, Phoenix, Arizona, August 1996.
- [15] P. Simpson and R. Alonso. A model for information exchange among autonomous databases. Technical report, Princeton University, May 1989.
- [16] J.M. Smith, P.A. Bernstein, U. Dayal, N. Goodman, T. Landers, K.W.T. Lin, and E. Wong. Multibase - integrating heterogeneous distributed database systems. In *National Computer Conference*, volume 50 of *AFIPS Conference Proceedings*, pages 487–499, 1981.
- [17] G. Thomas, G.R. Thompson, C. Chung, E. Barkmeyer, F. Carter, M. Templeton, S. Fox, and B. Hartman. Heterogeneous distributed database systems for production use. *ACM Computing Surveys*, 22(3):237–266, 1990.
- [18] A. Zisman. Towards interoperability in heterogeneous database systems. Technical Report 11, Department of Computing, Imperial College of Science, Technology and Medicine, 1995.
- [19] A. Zisman. A methodology to assist with a distributed information discovery process for autonomous databases. Technical Report 3, Department of Computing, Imperial College of Science, Technology and Medicine, 1997.
- [20] A. Zisman and J. Kramer. Distributed information discovery for autonomous databases. Submitted for Publication.
- [21] A. Zisman and J. Kramer. An architecture to support interoperability of autonomous database systems. In *2nd International Baltic Workshop on DB and IS*, Estonia-Tallin, June 1996.
- [22] A. Zisman and J. Kramer. An information discovery process for interoperable heterogeneous databases. In *Americas Conference on Information Systems*, Association for Information Systems, pages 617–619, Phoenix, Arizona, August 1996.