

An Architecture to Support Interoperability of Autonomous Database Systems

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

Abstract

We propose an architecture designed to support interoperability between autonomous heterogeneous databases. The architecture supports a distributed information discovery process and avoids the use of integrated schemas and centralised structures containing information about the shared data. Users of a database can access its local data as well as data from other databases in the system. Such access can be supported without violating the privacy and confidentiality of the data, and preserving the autonomy of the databases. With the proposed architecture it is possible to support change by adding and removing databases from the system.

1 Introduction

The development of database management systems has increased the utility of such systems, but has not solved the problem of having a great number of separate databases in a large company or community. Users would like to access and manipulate data from several databases, and applications may require data from a wide variety of independent databases. These databases are generally created and administered independently, differing physically and logically. Therefore, we are faced with the challenge of moving from the situation where single databases are accessed and manipulated in isolation, to the case where these databases support interoperation, permitting applications to also access remote information. The interoperability should be supported without modifying the databases or losing their autonomy, and in a way that is relatively transparent to users and applications.

Various architectures and methodologies have been proposed in the literature addressing the problem of manipulating and accessing heterogeneous databases. Some of these architectures use a *global schema* integrating the databases [1, 6, 16, 17]. Other architectures avoid the use of a global schema [3, 5, 7, 8, 10, 11, 12, 13, 15] making the databases *interoperable*.

Based on the existing architectures we propose an approach to the problem of interoperating with a large number of autonomous heterogeneous databases. The architecture aims to permit users of different databases to access local data and

the primary goal of this study is to design, and evaluate, a methodology. This approach tries to support a distributed information discovery process. It attempts to avoid the use of integrated schemas and of centralised structures, containing information about the shared data. It also aims to be evaluable, supporting the addition and removal of databases, without stopping the whole system. We are not considering information sources of non-database types.

The rest of this paper is organised as follows. Section 2 contains a survey of related work, comparing some of the existing architectures and methodologies. In section 3 we present the proposed architecture, describing all of its different components, a brief description of the information discovery process and how to perform the evaluable aspects of the approach. Section 4 contains an example. Section 5 summarises the results and suggests further work.

2 Related Work

Various architectures have been proposed to support interoperability of different heterogeneous database systems. However, there is little work related to the distributed information discovery process. As outlined by Sheth [14], in the future, the primary issue will not be to efficiently process the data that is known to be relevant, but to determine which data is relevant and where it is located.

Some of the approaches proposed in the literature use a centralised structure like dictionaries and/or repositories, containing information about the objects that are exported by the components [5, 7, 8, 10]. However, when dealing with a large number of databases the use of dictionaries and repositories is not suitable. They violate databases autonomy and do not guarantee privacy and confidentiality of the components. The databases have to notify all data that they export into it. A centralised structure is also not realistic for a system where components often join, leave and change their specifications. Other approaches permit the interoperation of databases without using a centralised structure [3, 11, 12, 13, 15]. We present and compare below some of these proposed architectures.

The *federated* [9, 10] and *multidatabase* [11] *architectures* were suggested to produce a better alternative to the use of global schema. They guarantee autonomy and cooperation in interdatabase sharing. In the federated architecture the sharing of data is performed by *export* and *import* schemas. Each federation has a *federal dictionary* describing available data items and services in the federation. The information discovery is executed in two steps. First, the requiring component consults the federal dictionary for available schemas. Second, it imports the correct schemas and browses them for a relevant information. A mechanism of negotiation is used to coordinate the sharing of information between two components. We affirm that in a large unstructured network of dictionaries it is difficult to locate the right information. In the multidatabase architecture databases are accessed by a multidatabase language or through an *external view*. It is assumed that users know what information is of interest. However, with a large number of components this idea is not feasible.

¹During the text we use the terms *database* and *component* interchangeably.

individuals, allowing for data sharing (*local schema*, *cooperated schema*, *federated schema*, and *external schema*). However, it is necessary to execute (partial) integration in the federated schema, which is not a simple task. It has to deal with the dissimilarities of the different schemas (semantic and syntactic conflicts), causes difficulties in the addition and removal of databases, and does not preserve the autonomy of the databases.

In [2] it is possible to find some issues of information discovery in the context of data sharing among a large number of autonomous database systems. The approach relies on an external indexing scheme, where each node of the index contains a network address with a set of descriptions called *skeletons*. The authors do not specify how the information is summarised and how the actual node selection is performed. The databases autonomy is not strictly respected, since databases have to provide both the content and the structure to other databases. It is also not specified how the external indexes are updated when components are added or removed into/from the system. In [2] the authors assumed, for simplicity, that all the data is public, ignoring security aspects.

Milliner and Papazoglou [12] proposed an approach to inter-node relationship discovery for a network with a large number of databases. It is based on a client/server-oriented multidatabase (CSOMD) architecture. To maintain the organisational autonomy of the nodes, there is a high level “context abstraction” that defines objects using *Global Concepts* (GCs). These GCs divide the Universe of Discourse (UoD) into dynamic clusters of database nodes, based on areas of interest, thereby reducing the search space. The databases are related to the GCs by *link weights*. The system may allow dynamic merging and splitting of the GCs, and updating of the link weights. In [12] the authors did not specify how to organise the GCs and how to add and remove components of the system. They assumed, for simplicity, that all nodes in the system are registered during initialisation.

In [3, 4], a framework and system called FINDIT was proposed to address the problem of information discovery for a large number of databases. The idea is to dynamically educate the users about the available information space, assisted by an interoperable language named TASSILI. FINDIT is a two-level approach based on *coalition* (first level) and *service* (second level). A coalition is a group of databases that share some common interest in an information type. A service is provided as a means to share minimal information description. Each participating database contains an object-oriented *co-database* storing information about coalition and services in which it is involved. The resolution of a query is performed with a high interaction of the user, allowing a database to know what other databases contain. To guarantee protection of data the system uses checking mechanisms to identify if a data can be accessed by a specific user. In FINDIT, it is very difficult to manipulate queries which contain more than one type of information.

3 The Proposed Architecture

Based on the existing approaches, we propose an architecture which shares some features with previous work. The proposed architecture tries to support a distributed information discovery process for a large number of databases. It does not use the

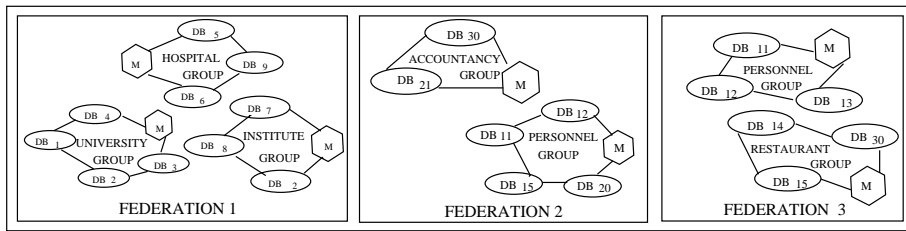


Figure 1: Federations of heterogeneous databases

obvious solutions of either having a global centralised structure, or broadcasting a query to all the participants in order to find an information. We also aim to avoid the use of integrated schemas like federated [15] and import [9, 10] schemas. The goal is to permit heterogeneous autonomous databases, independently created and administered, to interoperate with each other, in as transparently as possible, without compromising their autonomy. That is, the architecture intends to permit the situation where a certain query q' is performed by a user of a database in the system (source component), and the answer for this query, or part of the answer, is presented in a different database (target component). The approach aims to preserve the privacy of data and to be evaluable, allowing the addition and removal of components to be performed in an easy way.

In the proposed architecture the component databases are arranged into *federations* as illustrated in figure 1. A federation consists of a set of databases willing to share data with each other. The criteria used to form a federation are based on the shared data of the components and the components that are allowed to access this shared data. Inside a federation, the shared data of a component is public for all the other components in this federation. However, there is no communication between two federations. A database outside a federation cannot access the shared data of the components in this federation. It is possible to have a database participating in different federations (overlapping), sharing a distinct set of data in each of these federations (DB_{11} , DB_{12} , DB_{15} and DB_{30} participate in federations 2 and 3 at the same time). The idea of using federations is to provide a natural way of guaranteeing privacy and security of the shared data. It also avoids the necessity of executing *negotiations* [2, 3, 10, 12] between two components that are exchanging data, after identifying a required data (access rights).

As proposed by Milliner and Papazoglou [12], inside a federation, we organise the participating databases into *groups*, to make the universe of search smaller, facilitating the information discovery process. A group of databases is formed based on the type of data shared by these components. However, the existence of these groups are only related to classification aspects. For instance, a federation can contain a *university group*, a *hospital group* and an *institute group*, as presented in federation 1 of figure 1. When a certain type of data is related to two or more different areas, it is possible to have a database that participates in more than one group in the same federation (DB_2 is related to university and institute groups). It is also likely to have the same group name in different groups of distinct federations.

As presented in figure 1 each group contains a special component, named *master*

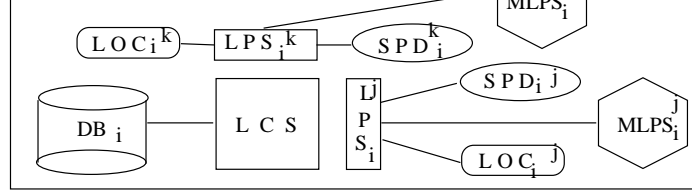


Figure 2: A component database with its extra structures

(M). The master records the location of the different components participating in a federation. It assists in the process of adding and removing databases, since the notification about these processes are first executed in the master. It is responsible to notify these updates to the other components in the system. The master is also important in the discovery process, since it contains the more up to date information about the components of a certain group. Thereby, during a discovery process, whenever a requested data is not found in any of the components of a group, the master of this group is consulted. The procedure tries to identify in the master a not yet visited database which possibly contains this data. Thus, when the masters do not know about components that possibly contain the requested data no other component in the system knows about this. The type of information that the master holds and the way that it is structured is the same as the LOC structure specified below in item 4.

Each component in the system has a set of additional structures named LPS, MLPS, SPD and LOC, for each federation that it participates in. These additional structures allow the interoperability of the heterogeneous databases without modifying their original structure and preserving their autonomy. Figure 2 shows a database that participates in two different federations, j and k , with its sets of additional structures. In the following paragraphs, we explain the need for these structures and their functionality.

1. **LCS/LPS:** As can be seen in figure 2, the non-public data (private) of each component is represented in the *local component schema* (LCS), in the native data model of the component. The information that a component shares with the other participants of its federation is called *public data* and is registered in a special schema called *local public schema* (LPS). The LPS is also specified in the native data model of its component. Thus, for each federation that a component participates in, it can potentially share a different set of data, represented in a distinct LPS.
2. **MLPS:** Associated with each LPS there is a *mirror local public schema* (MLPS). The MLPS contains the public data of the component expressed in a common (canonical) data model for all the federation. Two federations in the system can use different canonical data models to express their MLPSs, depending on the native data models of their components. Unlike the existing approaches, the architecture uses the representation of the shared data in the local and canonical data models. The purpose of representing public data in two different data models are to avoid unnecessary translation work, when exchanging

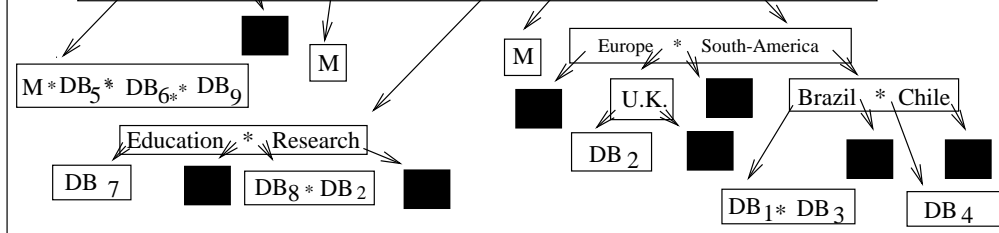


Figure 3: Hierarchical structure - LOC and M

data between components that have the same data models; and to maintain a linear number of translators, when performing interoperability between databases containing different data models.

3. **SPD:** We propose the use of a structure called *semantic public data* (SPD) to facilitate the task of identifying similarities, equivalences and conflicts when exchanging data. The SPD expresses and describes semantic information of the public data in a different way from the LPS and MLPS (not as a data model). The semantic information in the SPD is particular for each component and not for all data in the system.
4. **LOC (and Master):** The information discovery process is assisted by the use of a hierarchical structure named *locate* (LOC)². This structure contains the names of the groups of the related federation and *specialised terms*, forming a natural hierarchy of names (from general to specific terms). The terms are related to the type of information that the components are sharing. Associated with these terms there are references to the concerned databases.

Figure 3 presents the LOC structure for federation 1 in the example of figure 1. The first node in the hierarchy (root node) contains the names of the groups. The internal nodes contain specialised terms. A node in one level contains more specific names than the terms inside its parent node.

Inside a node the terms are strings of characters, separated by a special mark (*) and organised in alphabetical order. Associated with each term there are left and right pointers. The left pointer either points to a leaf node or is undefined (nil), represented as black square boxes in the figure. The right pointer either indicates an internal node holding specialised terms or is undefined (nil). The left leaf node related to a term in the root node can contain: (a) the address of the master of the group associated with this term (represented as M), when the term is specialised; (b) the address of the master together with the addresses of different components in the group (represented as $DB_i, 1 \leq i \leq n$, where n is the number of different components in the system, and separated by (*)), when the term is not specialised. The left leaf node related to a term inside an internal node: (a) can be undefined, when the term has specialisations; (b) can contain address(es) of component(s) related

²The master and the LOC are the same structures. They differ only in the fact that the master can contain more up to date information. In this part of the text we use only the term LOC.

the projects in execution at Brazilian Universities. Thus, DB_1 and DB_3 are identified as the components that possibly contain the requested data.

The proposed structure is unbalanced. It depends on the level of specialisations that can be achieved for each group of the federation. The use of a balanced tree limits the natural hierarchy of the terms.

Each federation contains a human *manager* to help in the organisation of it. The manager is responsible to decide about the different groups of the federation, the names and terms associated to these groups, the common data model used in the MLPS, and so on. Alternatively, each database has an *administrator* (DBA). The DBA decides about the different data that the component can share, the federations that it participates in, and the semantics associated with the shared data.

3.1 The Information Discovery Process

Because of limitation of space we do not present here the details of the distributed information discovery process, providing only a brief explanation. A detailed description of this process can be found in [18].

Consider the situation where a user of a database DB' (source) performs a query q' and DB' does not contain the requested data d' . Suppose that DB' participates in more than one federation. The first step consists in identifying which is the federation that DB' participates in that is related to q' . After this, the LOC structure of DB' associated to this federation is traversed. Thereby, a specific group of components having the sets of shared data related to d' is identified. These components are called *possible target components*. The identification is executed based on the terms used to classify the components. In the next step, q' is sent in parallel to these “possible target components” to verify the existence of d' . If a possible target component does not contain d' , then its LOC structure is consulted, in order to identify “new” possible target components. The process is recursive and it is guaranteed to finish either when d' is found (success), or d' is not found (failure). In the latter case, all components of the related group, including its master, are visited. The master of a group is visited to identify other possible target components. This is performed whenever the requested data is not found in the possible target components already known and visited.

The information discovery process for a query q' consults only the components of the groups in one federation related to the query. It depends on the way in which the components are grouped and in the correct choice of the terms to express the organisation.

3.2 Evolution

The proposed architecture is designed to permit dynamic evolution of the system. It allows a component or a group of components to join or leave a federation without having to stop the whole system, thereby preserving the autonomy of the databases. In the approach, a component (or group) has autonomy to decide whenever it wants to participate in the system. We divide the evolution process into three cases: (a)

processes we suggest the use of a special structure named *Syst-DB*, together with the assistance of DBAs and managers of the involved databases and federations, respectively. The *Syst-DB* is a structure that contains enough information to help on identifying the correct federation/group to insert a new component (“correct place”) and can be implemented as a replicated object-oriented database.

The addition of a “new” component refers to the participation of this component in a different (new) federation (this component can exist in the system participating in other federations). Similarly, the deletion is the removal of a component from one of the federations that it participates in.

In any of the addition cases the *Syst-DB* is initially consulted. The addition of a single component can cause the creation of either a new group in a specific federation, or a new federation. When an existing federation is identified as the “correct place” to insert a “new” component, its participating databases have to agree on sharing and exchanging data with this “new” component (authorisation). The authorisation is executed by consulting the DBAs of the involved databases. Afterwards, a group has to be identified or created, together with the specification of new terms, to classify and allocate the “new” component. The manager of the federation is responsible to perform these tasks. The next step consists in updating the master of the group related to that component. We propose the idea of first updating strategic points (the masters) and then, whenever possible updating the LOC structures of the other components. The creation of a new federation requires choosing a manager for it. This manager is responsible for specifying all the aspects of the federation. After executing an addition case it is necessary to notify the *Syst-DB* about the modifications in the system. This is performed by the manager of the involved federation.

For the removal cases we allow the LOC structures of the components to hold incorrect information for a certain period of time. Therefore, a discovery process for a query q' can try to access a non-existent component. The idea is to allocate a “special notification” into the position of a removed component. For the system, a component does not exist in a federation, from the instant that the “special notification” is introduced into its position. Afterwards, the *Syst-DB* is updated, and the masters and the LOC structures of the other participating databases are gradually notified about the modifications.

The modification case arises in situations where a component wants to modify any of its set of shared data (LPSs). Changes in one of the LPSs of a component requires the review of the maintenance of this component in the respective federation. The manager of the federation, together with the DBAs of the participating databases, analyse the situation. A detailed description of the addition, removal and modification cases can be found in [18].

4 Example

We present below a simple example to illustrate the proposal. Consider the following scenario involving *book-stores*, *chemists*, *hospitals*, *universities* and *credit-card* companies. Suppose that a company named “Company-Shops” owns two different



Figure 4: Federation of the databases in Company-Shops

book-stores and two chemists in different parts of the world. Each of these shops has its own database independently created and administered. To facilitate the control of the company, the owner of the company wants to have an overall view of the accounts without creating a single central database with that information. Alternatively, Company-Shops and other book-stores and chemists decide to build a “pool of shopping” (network), to improve the sales in the shops. The aim is to permit people from different universities and hospitals to buy the products in the pool via the network. Payment is made either by credit-card or by direct deduction from the payroll of the staff.

The databases are organised in two different federations in order to guarantee privacy and confidentiality. Figure 4 presents federation 1 formed by the databases of the book-stores and chemists of the Company-Shops. These databases interoperate to control the accounts, profits and expenses of the company. Federation 2, in figure 5, contains the “pool of shopping” with the databases of the book-stores, chemists, hospitals, universities and credit-card companies.

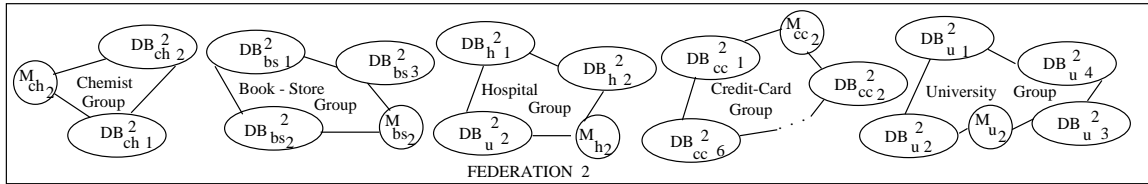


Figure 5: Federation of the databases in the pool of shopping

In figures 4 and 5 the databases are represented as $DB_{x_i}^j$, where: $x \in \{bs, cc, ch, h, u\}$, with bs, cc, ch, h, u corresponding to book-stores, credit-cards, chemists, hospitals and universities, respectively; i is an integer expressing a database in the system; and $j = \{1, 2\}$, represents the two federations of the example.

We select components DB_{bs_1} and DB_{bs_2} to demonstrate some structures of the architecture. Suppose that federation 1 and federation 2 use the relational and object-oriented data models as the canonical data model, respectively. Consider DB_{bs_1} representing its conceptual schema in the E-R model, with part of its LCS presented in figure 6 (a). Alternatively, suppose DB_{bs_2} a relational data model, with part of its LCS shown in figure 6 (b). Both of these databases participate in federation 1 and federation 2. Thus, each one contains two groups of additional structures ($LPS_i^j, MLPS_i^j, SPD_i^j, LOC_i^j, i, j = \{1, 2\}$).

Assume that in federation 1 LPS_1^1 and LPS_2^1 are equal to the local conceptual schemas of DB_{bs_1} and DB_{bs_2} (figures 6 (a) and (b), respectively). For federation 2, LPS_1^2 and LPS_2^2 are presented in figures 7 (a) and (b), respectively.

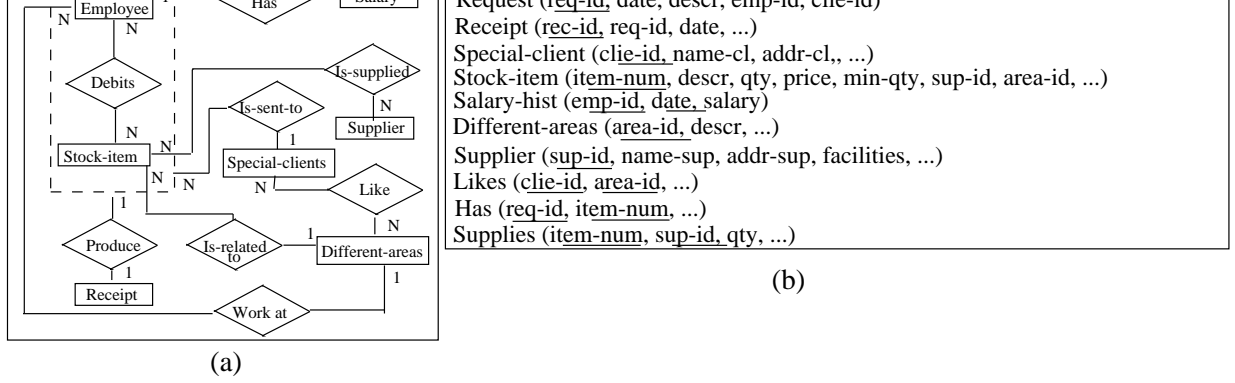


Figure 6: Part of the conceptual schema of (a) $DB_{b_{s_1}}$ and (b) $DB_{b_{s_2}}$

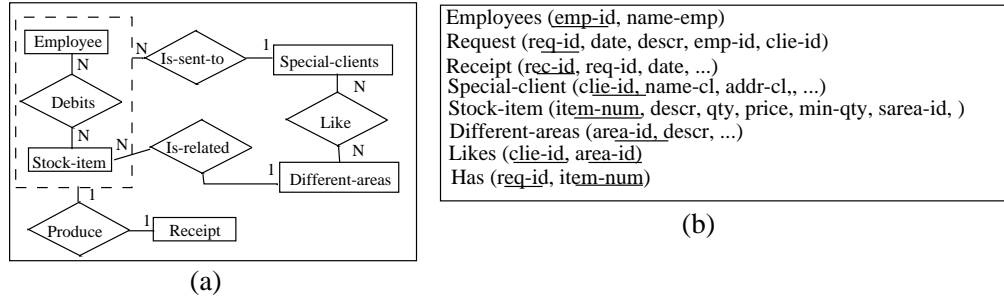


Figure 7: Part of the local public schema of (a) $DB_{b_{s_1}}$ and (b) $DB_{b_{s_2}}$ in fed. 2

5 Conclusion and Further Work

In this paper we have proposed an approach to the problem of interoperating with large numbers of autonomous heterogeneous databases. Our goal is to move from an environment where single databases manipulated in isolation, can interoperate without the need to re-implement the databases or modify them. Therefore, we wish to allow users of different databases to perform queries requesting data from other databases.

The different databases are organised into federations, depending on the data to be shared and on the other components with which they want to share this data. The federations are designed to guarantee privacy and confidentiality of the shared data, and to avoid negotiations between the databases. Inside a federation the databases are organised into groups and classified by special terms, depending on the type of shared data. This is performed to reduce the universe of search during the discovery process. Unlike FINDIT [3] and federated architecture [10], our idea of federations avoids the need to negotiate after identifying the requested data.

When interoperating with autonomous heterogeneous databases it is very important to determine which is the relevant data and where it is located. Thus, the approach attempts to support a distributed information discovery process. In contrast to the five-level [15] and federated [9, 10] architectures, our approach aims to

with the help of a hierarchical structure. The correctness of this process is directly related to the way that the groups are formed in a federation and the terms are chosen.

We believe that the approach is evaluable, supporting the addition and removal of databases without stopping the whole system. These are executed with the assistance of master structures, managers of the federations, and DBAs of the participating databases. Note that in [2] the authors do not specify how the external indexes are updated when additions and removals occur. In [12] it is assumed that all nodes in the system are registered during initialisation.

We need to gain experience in order to evaluate the proposal. In particular, we are investigating aspects concerning: (a) implementation of the information discovery process; (b) specification of how to perform the queries and solve the semantics problems; (c) proposal of an approach to identify the correct federation of a source component to be searched; (d) control of the participation of a component in many federations; (e) problem of having a large number of federations in the system; and (f) evaluation of the approach through modelling.

Acknowledgements

We gratefully acknowledge Dr. Julie McCann for her constructive comments. Thanks are also due to Prof. John J. Florentin and the members of Distributed Software Engineering Section at Imperial College. In the case of the first author, this work was supported by Brazilian foundation CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) grant No. 200209/94-9. For the second author, this work was partially funded by Information System Interoperability (ISI) project, supported by EU on grant No. ECAUS003.

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. *A Dynamic Framework for Interoperability in Large Multidatabases*. PhD thesis, Faculty of Graduate School of the University of Colorado, 1992.
- [4] A. Bouguettaya, S. Milliner, and R. King. Resource location in large scale heterogeneous and autonomous databases. *Journal of Intelligent Information System*, 5(2), 1995.
- [5] M. W. Bright, A. R. Hurson, and S. Pakzad. Automated resolution of semantic heterogeneity in multidatabases. *ACM Transaction on Database Systems*, 19(12):212–253, June 1994.

- [7] A. K. Elmagarmid, J. Chen, W. Du, O. Bukhres, and R. Pezzoli. InterBase: An execution environment for global applications over distributed, autonomous, and heterogeneous software systems. Technical Report TR112P, Purdue University, 1994.
- [8] 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.
- [9] M. Hammer and D. McLeod. On database management system architecture. *MIT Lab for Comp. Sc.*, (MIT/LCS/TM-141), October 1979.
- [10] D. Heimbigner and D. McLeod. A federated architecture for information management. *ACM Transaction on Office Information Systems*, 3(3):253–278, July 1985.
- [11] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22(3):267–293, September 1990.
- [12] 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.
- [13] M. K. Papazoglou, S. C. Laufmann, and T. K. Sellis. An organizational framework for cooperating intelligent information systems. *International Journal of Intelligent and Cooperative Information Systems*, 1(1):169–202, 1992.
- [14] A. P. Sheth. Semantic issues in multidatabase systems. *SIGMOD RECORD*, 20(4):5–9, December 1991.
- [15] A.P. Sheth and J.A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.
- [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, September 1990.
- [18] A. Zisman. Towards interoperability in heterogeneous database systems. Technical Report 11, Department of Computing, Imperial College of Science, Technology and Medicine, 1995.