

LARGE MULTIDATABASES: BEYOND FEDERATION AND GLOBAL SCHEMA INTEGRATION

Athman Bouguettaya

School of Information Systems
Queensland University of Technology, Gardens Point Campus
2, George Street, GPO Box 2434
Brisbane 4001, Queensland, Australia
athman@fitmail.fit.qut.edu.au

ABSTRACT

Advances in wide area networking is making the goal of achieving a world-wide interoperable database system closer than ever. Because of the large environment, new problems have surfaced that need to be addressed. We distinguish several reasons for a new design approach. Traditional approaches like global integration and federated approaches are better suited to small and medium scale heterogeneous databases. In a large network of interoperable autonomous heterogeneous databases, the architecture should be flexible enough to allow negotiation to take place to establish grouping of databases. In such a large environment, it is important that databases be made aware of other participating databases in an incremental fashion. Further, users should be educated about the space of information in an incremental and dynamic fashion. These goals are to be achieved under the major assumptions that participating databases keep their autonomy intact as much as possible and that the overhead in bridging heterogeneity be proportional to the level of sharing. In this paper, we describe a two-level approach that addresses these issues.

1. Introduction and Background

In this information age, the need to share data across autonomous heterogeneous databases has become a necessity and not a luxury. Organizations all over the world rely on a wide variety of databases to conduct their everyday business. In many instances, databases are designed from scratch if none is found to meet the organization requirements. This has led to a proliferation of databases obeying different set of requirements and modeling. Intuitively, it is a lot cheaper to reuse an existing piece of information than to create one. In many instances, and because of a lack of any organized conglomeration of databases, users create their own pieces of information that may exist in current databases. Sometimes, although it is known that a certain piece of information is stored in a database, finding it and the database where it is stored can be prohibitive in terms of cost and effort. Figure 1 summarizes the problems introduced by large scale (Bouguettaya, 1993).

We believe that one of the fundamental problems building a large scale network of interoperable databases lies in the fact that up to now researchers have solely considered full integration as a means to achieve interoperability. In this respect, one of the fundamental problems is understanding how semantically apart databases are (Sheth, 1993)(Krishnamurthy, 1991)(Saltor, 1993). Although progress has been made, we are far away from any satisfactory solution (Sheth, 1993). This is particularly true for a small number of heterogeneous databases. If the environment is large, the problem becomes almost impossible (Bouguettaya, 1993). We proposed a new architecture that provides a flexible framework for databases to interoperate (Bouguettaya, 1992)(Bouguettaya, 1991)(Bouguettaya, 1993)(Bouguettaya, 1994). This system is called *FINDIT*.

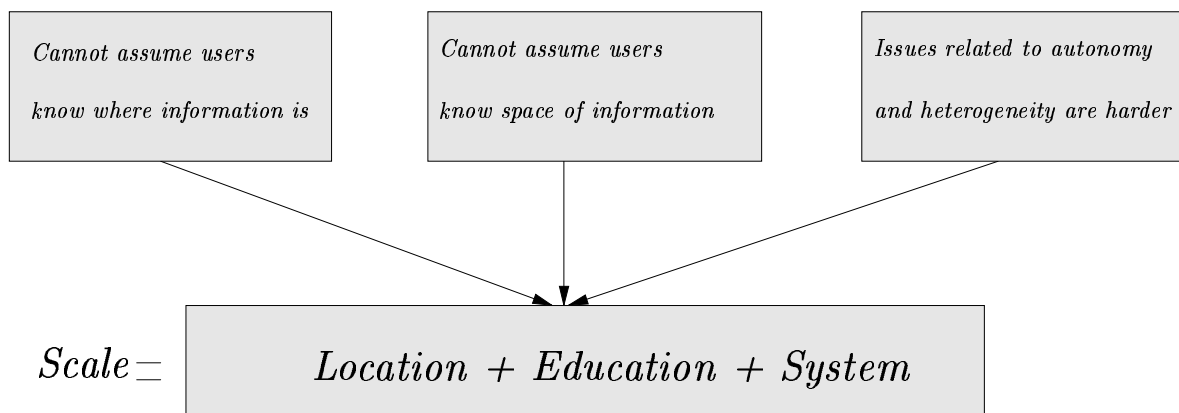


Figure 1: Problems Introduced by Scale

To enable users to find the right information and to provide a tool for participating databases to share information, there is a need to provide an interoperable language. The problems we are addressing in this research are new in the area of heterogeneous databases. So, none of the existing languages (Litwin, 1987)(Elmagarmid, 1989)(Krishnamurthy, 1991)(Elmagarmid, 1990) were built to address the issues related to the large scale. An adequate query language must provide constructs to locate information and information sources along with incrementally educating users about the available space of information.

The problem is that there are few languages for multidatabase systems to begin with. Those existing languages were designed to operate under specific environments. In (Litwin, 1987), users are presumed to know the target schemas and these are assumed to be organized under the relational data model. (Krishnamurthy, 1991) is meant to build views assuming users know the information and where it is located. (Elmagarmid, 1990) describes a parallel logic language that is mainly used for specifying transactions in a multidatabase system. (Elmagarmid, 1989) uses the *Distributed Operation Language (DOL)* to address heterogeneity. A service directory provides location and distribution transparency. The constant across these three approaches is the assumption that users know exactly the target schemas and what they are looking for. Further, there is, in many instances, a schema integration or transformation necessary to present users with a homogeneous view of the participating databases. These assumptions are legitimate because the number of databases considered is always small. They are also legitimate because the underlying architecture assumed, is either based on schema integration (Batini, 1986) (Templeton, 1986) (Landers, 1982) (Smith, 1981) (Litwin, 1982) (Stocker, 1984) (Kim, 1991) (Motro, 1981) or federated databases (Heimbigner, 1985) (Sheth, 1990). In a large network of databases, the above assumptions cannot be met.

The paper is divided as follows. In Section 2, we present an overview of FINDIT framework that includes new concepts needed to address the new challenges. Section 3 discusses the language support for large environments. In Section 4, we present an example. In Section 5, conclude with some remarks about the research presented in this paper. Finally, in Section 6, we list the references.

2. FINDIT Framework

In FINDIT, databases are grouped into *coalitions* that may be related to each other by *service links*. Each of the participating databases has a *co-database*¹ attached to it that stores information about the coalitions it is a member of as well as service links it is part of. A coalition stores information about a grouping of databases that share some common interest about a certain type of information. Databases that are members of a coalition are said to be strongly connected. A service link is a means for databases to share information and be loosely connected. Coalitions and service links provide a flexible mechanism through which databases can exchange information on their own terms.

¹The use of the word "co-database" is similar to the use of the prefix "co" in other terms. In our case, we choose to name this database a co-database because it is an accessory to the actual database. It is only used for specific purposes.

2.1. *Coalitions*

A real life parallel with our concept of coalition is how political systems work in western democracies. In many countries, political parties obeying different ideologies agree on a minimum set of objectives for a limited period of time to form a coalition. Therefore, coalitions are based on short term interest. In political coalitions, members keep their autonomy intact while being committed to a set of rules agreed upon. Our concept of coalitions is therefore very close to the concept of political coalitions. On the other hand, the concept of federation, as defined in (Heimbigner, 1985), is more like the federation of states where the set of rules obey long term interests and where states enjoy a reasonable, though limited, amount of autonomy. It is important to keep in mind that the federated approach has introduced dramatic changes over previous approaches to sharing information.

In FINDIT, every component database has an object-oriented co-database attached to it. This co-database is used by the component database to describe what information other component databases in the coalition contain. It is also used by the component database to let users know what information they can have access to, along with the information about the providing databases. Instead of storing information about foreign data in the component database and thus making it handle queries about information stored elsewhere, the idea is to separate the storage of the data depending on its locality. This organization has the major advantages that it respect component database security, autonomy and heterogeneity.

A database may belong to more than one coalition. In this case, its co-database will contain information about all coalitions it belongs to. Two databases can belong to the same coalition and still have their co-database different. This is true because these databases might belong to other different coalitions. This is one reason it is desirable that each database have one co-database attached to it instead of having one single co-database for each coalition. Database autonomy and high information availability are other reasons why it is not desirable to physically centralize the co-database. In this case, it is better to replicate coalition schemas in several co-databases.

2.2. *Services*

Services are of three types. The first type involves a service link between two coalitions to exchange information. The second type involves a service link between two databases. The third service link involves a service link between a coalition and a database. A service link between two coalitions involves providing a general description of the information that is to be shared. Likewise, a service link between two databases also involves providing a general description of information that databases would like to share. The third alternative is a service link between a coalition and a database. In this case, the database (coalition)

provides a general description of the information it is willing to share with the coalition (database). The difference between these three alternatives lies in the way queries are resolved. In the first and third alternative (when the information provider is a coalition), the providing coalition takes over to further resolve the query. In the second case, however, the user is responsible for contacting the providing database to know more about the information.

Whenever databases are reluctant to show too many details of the type of information they contain, they have a choice to join in a service link with other databases or coalitions. In essence, service links are a means for databases to be loosely connected to other databases (coalitions). This provides a framework where databases exchange a minimum amount of data about information they would like to share. Only general information about actual information to be shared and information about the servicing databases are exchanged. It should be stressed that as in any service link, there is service to be provided by one of the servicers. This means that a service link between a database **A** and a database (coalition) **B** does not necessarily mean that each database has to provide the other with types of information. Instead, a service link may involve one database giving information to another without entailing the other doing the same thing.

A database might simultaneously be a member of one or more coalitions and be at the same time a service link. This usually happens when a database is willing to show extensive information about part of its data to some set of databases while giving little information about other parts to another set of databases. This mechanism enables different degrees of data sharing depending on the type of data to be shared.

The other type of service link is between two coalitions or between a coalition and a database. In this case, one coalition describes the type of information it deals with to the other coalition (database). Along with this description, points of entries are provided to the servicing coalition. It should be clear that service links may be cancelled (if there is an explicit provision) upon short notice and without any approval.

2.3. *Co-databases*

Two criteria determine how databases organize themselves in an IS-A schema: information interest and (optionally) geographical proximity. While we already talked about the first criterion, we need to give some motivation about the second one. First, let us emphasize that it is there optionally. It is up to the participating databases to actually choose to uphold it. Common sense and efficiency are the main reasons why this criterion ought to be followed whenever possible. It generally makes sense to form an association based on proximity due to certain social, cultural, and political motives. It may also be more efficient and less costly to associate with those databases that belong to a common geographic

location. This is not to say that this criterion should be taken as a rule. It may be that in certain cases, there are good reasons not to use geographic location as a criterion.

figure fig.ps height 3.5i width 3i level 100

Figure 2: A Skeleton of a Typical Co-database Schema

A typical co-database schema contains subschemas that represent coalitions that deal with specific types of information. Each of these sub-schemas consists of a lattice of classes where each class represents a set of databases that can answer queries about a specialized type of information. A database may be member of more than one coalition.

A co-database also contains another subschema. This schema consists on one hand of a subschema of service links the coalitions (it is a member of) has with other databases and coalitions; and the other hand of a subschema of service links the database has with other databases and coalitions. Each of these subschemas consists in turn of two subclasses that respectively describe service links with databases on one hand and service links with other coalitions on the other hand. Figure 2 shows a skeleton of a typical co-database schema. Description of the main classes as shown in Figure 2 follows.

Description about coalition service links includes information about points of entries and contact with those coalitions. Other descriptions will provide information to local databases choose the best point of contact if there is more than one point of entry in the target coalition. It should be noted that the subschema representing the set of coalition servicers will be the same for all databases that are members of the servicing coalition.

A set of databases containing a certain type of information is represented by a class in the schema. In particular, every class contains a description about the participating databases and a description about the type of information they contain. Some attributes describe a type of information while the remaining attributes describe the databases that contain this type of information. Description of the databases will include information about the data model, operating system, query language, etc. Description of the information type will include its general structure and behavior (if applicable). Since databases may have different views on the same type of information, only the common parts of the view will be represented in the class.

Class Codatabase

Attributes are

/* General information */

Information_type : {<synonyms>}

Information-name : {name}

/* Information about the schema */

subclasses : {class_name : number_of_instances}

Instances : {OID}

Methods are

/* Connection */

Connect(database)

Send_query(database, query)

/* Misc */

Send_mail(database, text)

Display_attribute

Display_subclass

Display_object

Display_hierarchy

Class Coalition

Inherit From Class Codatabase

Attributes are

/* Information about coalition */

dbms : <dbms_name>

operating_system : <os_name>

query_language : <query_language_name>

known_information : {<information_name, service|coalition,
fee internet_address, membership>}

coalition_members : {<coalition_name, internet_address>}

Methods are

/* Connection */

Connect(database | coalition | service)

Send_query(coalition | service)

/* Misc */

Send_mail(database)

Display_attribute(class)

Display_subclass(class)

Display_object(class)

Display_coalition(coalition)

Class Coalition_1

Inherit From Class Coalition

Attributes are

/* General Information about Actual Information */

/* Additional Information about Coalition */

database_members	:	{<name, administrator, internet_address>}
service_with	:	{<database coalition, administrator, internet_address, information_type>}

Methods are

/* Documented Behavior */

Display_behavior(Object)

Class Service

Inherit From Class Codatabase

Attributes are

/* Information about service link */

findit_member	:	boolean
fee	:	real
usage_cost	:	real
known_information	:	{<information_name, service coalition, internet_address, membership>}
term_of_service	:	<Text>

Class Coalition_Service

Inherit From Class Service

Attributes are

/* Information about servicing coalitions */

service_type	:	<coalition database>
--------------	---	----------------------

Class Coalition_Coalition_Service

Inherit From Class Coalition_Service

Attributes are

/* Information about servicing coalitions */

Point_of_entry	:	{<coalition_name, database_name, internet_address, postmaster>}
----------------	---	--

```

Class Coalition_Database_Service
Inherit From Class Coalition_Service
Attributes are
/* Information about servicing coalitions */
database_service      :    <database_name, internet_address, postmaster>
dbms                  :    <dbms_name>
operating_system      :    <os_name>
query_language        :    <query_language_name>
opening_times         :    <Integer_range>
findit_membership     :    boolean

```

```

Class Database_Service
Inherit from Class Service
Attributes are
/* Information about database service link */
service_type          :    <coalition|database>

```

```

Class Database_Coalition_Service
Inherit from Class Database_Service
Attributes are
/* Information about coalition service link */
point_of_entry       :    {<coalition_name, database_name,
                           internet_address, postmaster>

```

```

Class Database_Database_Service
Inherit from Class Database_Service
Attributes are
/* Information about database service link */
database_service     :    <database_name, internet_address, postmaster>
dbms                 :    <dbms_name>
operating_system     :    <os_name>
query_language       :    <query_language_name>
opening_times        :    <Integer_range>
findit_membership    :    boolean

```

3. Language Support

Query languages are the interface between databases and users. They provide an abstraction through which users manipulate and define entities. It is therefore an indispensable tool in a database environment. Very few languages have been developed to support locating information and information sources along with user education about information in a large network of heterogeneous databases. There are two main reasons: Either, there was no need for one or it was just not the subject of elaborate research. In the first case, no query language that had features for querying heterogeneous databases was needed because

users were presented with a single schema that was the integrated schema (Batini, 1986). Therefore users were presented with a virtual homogeneous database. In the second case, the number of multidatabase systems that were developed was small in itself (Sheth, 1990). Most of the systems developed were experimental in nature with the exception of a few that became operational. For those experimental systems, the emphasis was more on the lower level architecture and dealt mostly with transaction processing management (Litwin, 1990).

The Tassili² query language differs from traditional query languages in that it operates in a large and highly dynamic network of heterogeneous database. Since the unit of information sharing is the type, this query language is able to query higher order information either about actual information or the providing databases. This query language is also used to create, modify, and delete entities that are part of the FINDIT framework.

Tassili is designed to address issues related to the use, design and evolution of FINDIT. It is in no way meant to be a general-purpose query language. Instead, it is mainly intended to be a specific-purpose language. The specificity comes from the environment and purpose of FINDIT. The reason being that the environment provided in FINDIT is fundamentally different from the way databases have been so far thought of. For instance, databases have been used to actually store data that is of direct concern. All existing query languages have been designed to address this environment. In FINDIT, the information dealt with is of higher order (information types) and therefore needs some environment-specific manipulation and definition features.

Tassili, like many modern query languages, consists of both data definition and manipulation constructs (Bouguettaya, 1992). It is used to define the different schemas and their intrinsic relationships. It is also used to manipulate the schema states. It is obvious that this query language should have the features of an object-oriented programming language for the simple reason that it operates on object-oriented databases (co-databases).

As for the manipulation features, Tassili does not manipulate actual instances of information types. Instead, users use it to query the structure and behavior of information types. Users also use this query language to query information about participating databases. In essence, Tassili is not a query language as commonly understood in the database literature. Standard query languages in most cases, manipulate instances of objects. In contrast, the manipulation in Tassili is on both data and meta data. Tassili is therefore a data definition and data manipulation language.

²Tassili is a region in the southern tip of Algeria. It contains remains of a very ancient civilization, the Tuareg civilization.

As for the data definition features, Tassili provides constructs to define objects and classes and their corresponding relationships. In conventional object-oriented databases, the behavior of a class is the same for all its instances. In FINDIT, we use *documentation* that may be different from one instance of a class to another instance of the same class. Tassili also provides mechanisms for defining constraints and firing triggers. This feature is mainly used to evolve schemas as well as propagate changes to related schemas. This is an important feature. It has been mentioned before that some queries are of data definition type and hence are only accessible to a selected number of users (administrators). The formation of a schema is achieved through a negotiation process. In that respect, Tassili provides some language features for administrators to form a schema. In what follows, we describe the manipulation operations Tassili provides to meet the above requirements.

- (1) Search for an information type.
- (2) Search for an information type while providing its structure.
- (3) Search for an information type while providing its structure and/or information about the host databases.

In what follows, we describe what data definition operations Tassili provides to meet our requirements.

- (1) Define classes and objects.
- (2) Define the structure of classes of objects.
- (3) Define the behavior of objects through the use of methods.
- (4) Define operations for schema evolution.
- (5) Define operation for negotiation for schema creation and instantiation.
- (6) Privileged operations for database administrators.
- (7) Define functions for schema creation.

Tassili can be used in an interactive or batch mode. Graphical queries are also accepted and are essentially scripts of queries that are repeatedly made and about the same information.

4. Example

Assume there is an import/export company that specializes in trade between Pakistan and the US. It has a database that contains information about goods exchanged between the two countries. It also contains past and current services and information related to these transactions.

We assume that this company's database is part of FINDIT. The president of the company assigns some employees to look for the best deals possible in acquiring these items. The idea is to minimize the overhead incurred in searching and looking for the right item. Every user is in charge of finding and eventually

acquiring the information needed. We assume the relevant world that may be accessible to our export/import company is as presented in Figure 3.

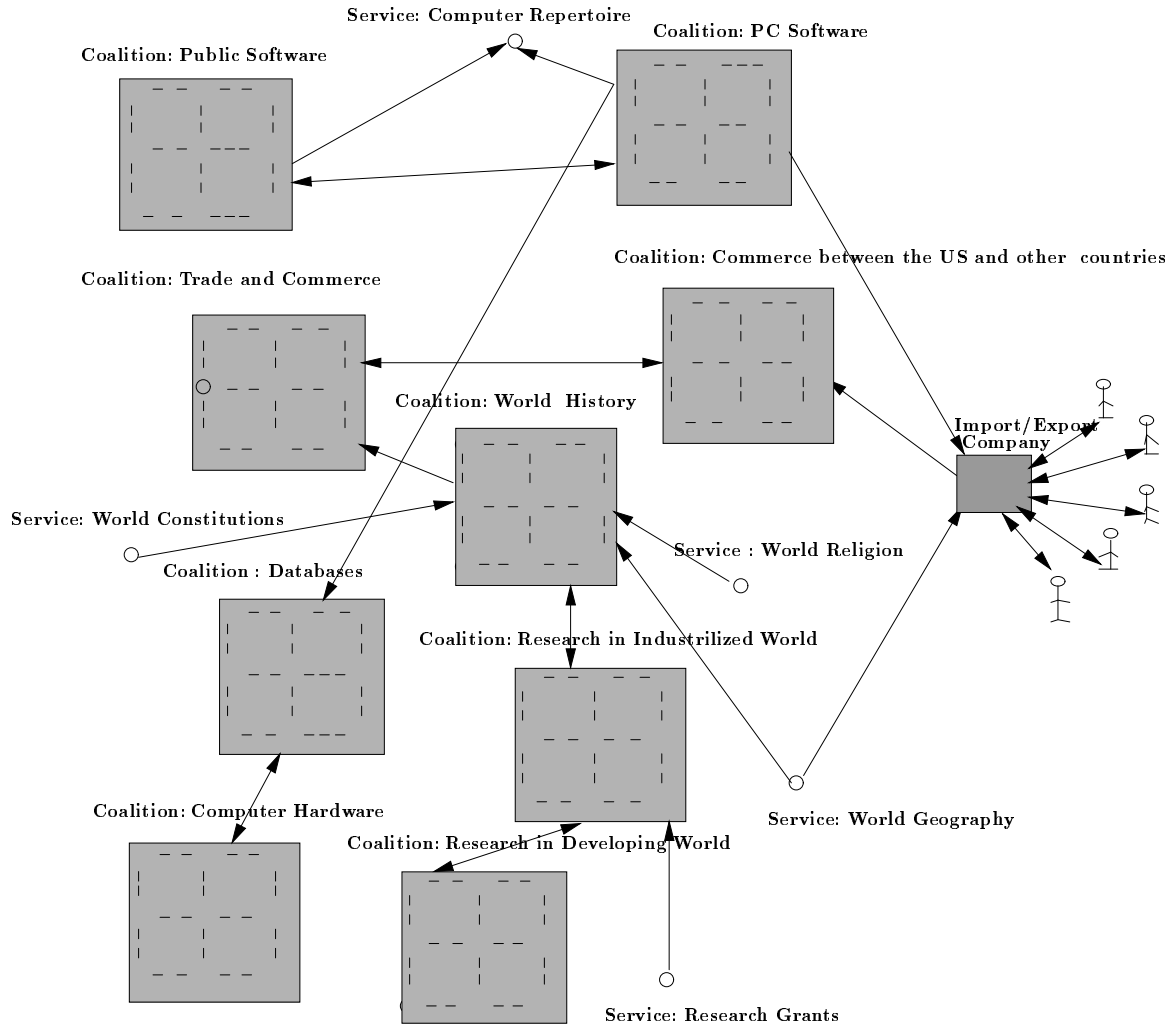


Figure 3: A Partial View of a FINDIT Network of Coalitions and Services

The example above shows nine (9) coalitions related to each other with

services³. This example also shows five database services. A bold line indicates the membership of the export/import company. In this example, this company is member of the coalition *commerce between the US and other countries*. It also provides a service to a database (*World geography*). The arrows point to the direction of services that are provided. Therefore the co-database created by the import/export company will contain information about the following coalitions and services. It keeps track of the coalition *Commerce between the US and other countries*. This co-database also contains information about other coalitions and databases that have a service with the coalition our import/export company is a member of. For instance, the company co-database stores information about the service *Trade and Commerce*. This co-database also contains information about the databases or coalitions that have a service with the company. In particular, the co-database contains information about the services *World Geography* and *PC Software*. A service between two coalitions includes an exchange (either unidirectional or bidirectional) of information type descriptions. This information is also stored in the co-database.

Increasing the company's chances to play a central role in business between the two countries requires an expertise about everything that relates to their trading. Therefore, the company is interested in information related to the social, political, and economic history of both countries. This information is to be used by both US and Pakistan business operators to inquire about prospective commercial transactions and contracts. In particular, the company is interested in acquiring the following information:

- (1) Economic overviews about Pakistan and the US.
- (2) Publications and reports written about the general situation of these two countries.
- (3) Historical perspective about exports and imports between the two countries.
- (4) Studies about the laws and regulations in the two countries.
- (5) Religious and cultural aspects in both countries.
- (6) Studies about past relations between these two countries.
- (7) Archives about the main scientific and technological research and achievements in the two countries.

The executive committee of the company would also like to install an automated office information system to be able to provide the best service possible. To achieve this goal, they need the following items:

³The little circles represent databases that either participate in services or coalitions. The number of databases in each coalition is not necessarily the same.

- (1) Local Area Network software that could connect different hardware and operating systems platforms.
- (2) Public domain text editor that could be installed on PCs and Macs.
- (3) A public domain mail software for PCs and Macs.
- (4) A spreadsheet for both PCs and Macs.
- (5) A public domain relational database.

We are now set to let users query FINDIT to find the different items they need. In what follows, we will go through the process of looking for some needed items. One of the employees queries FINDIT for a spreadsheet. FINDIT will first check whether the coalitions the company is member of has the information. For instance, a Tassili query (FINDIT query language) would look like:

Find Coalitions With Information spreadsheet

If there is no match with any name in the list of names, the system will pick a coalition and then ask the user if this corresponds to the information type. This process is repeated until either all coalitions are exhausted and no match was found or a coalition is recognized. As can be seen from the example, the coalition *Commerce between the US and other countries* does not deal with this type of information. However, service *PC Software* appears to deal with this type of information. A point of entry is provided for this coalition. To establish a connection with a coalition or a database, a user would use the following Tassili query:

Connect To Coalition Service At Entry_Point pc_software

The user query is then submitted to that coalition. It is refined until a specific information type is found. A Tassili query used for refinement would be:

Display SubClasses By Attribute of Class spreadsheet

The user is then faced with a choice of many spreadsheets contained in many databases. The user then queries specific databases to find out about the providing databases as well as the environment in which the software is supposed to perform. A determining factor will be a demonstration of the software at a user's site and how it performs. This test of the spreadsheet software will provide an idea about its performance, complexity and ease of use. Tassili provides a construct for displaying the documentation of this information. An example of this query would be:

Display Document of Instance lotus of *Information* spreadsheet

Submitting this query may have the effect of triggering an interactive session between users and the database. In this case, the database may ask users to specify their environment so as to provide an adequate documentation. This tool, therefore, has the important feature that it allows the understanding of foreign information with little effort from users. Assuming the user finds a database that contains the requested information, functions are provided to directly access the

database for an instance of this information type.

Another example is a user who would like to find an appropriate relational database. It is obvious that the coalition our company is member of does not deal with this information type. The employee would use the following Tassili query to find a service that deals with this information:

Find Coalition Service With Information database

Fortunately, the coalition *PC Software* is here again to help our user. The query is submitted to that coalition for further resolution. The information type is in the domain of a service with the coalition *Databases*. The query is again submitted to that coalition. The query is refined until an appropriate class is found. The user will then use the documentation to test the relational database. The documentation includes tests about the multiuser aspect as well as results on certain benchmarks and performance measurements. Again, the documentation enables users to have on-line experience and understanding with foreign information.

Yet another example of a query is an employee querying FINDIT for the laws and constitution of Pakistan. The system checks whether any of the coalitions it is a member of has this information. No coalition deals with this information type. The system checks whether any services with databases or coalitions deal with this information type. At this point, the service with coalition *Trade and Commerce* is a candidate. The user is provided with a point of entry and other useful information about this coalition. Querying this coalition, the user finds out that another actual coalition candidate is the service with the coalition *World History*. Again, the query is resubmitted to that coalition. The user then finds out that the information in question is provided by a service with the database *World Constitution*. Assuming this information corresponds to the user's predefined requirements, the user will either have to directly contact the database for querying an instance of the information or use functions to do so. This will depend on the terms of the service.

In the process of looking for the above information, the employee finds out that coalition *World History* has a service link with another coalition (*Research in Industrialized World*) that looks to be useful for the company's goals. Therefore, the employee decides to query this coalition looking for possibly relevant information. Browsing through this coalition, the employee finds out that there is a class that deals with economic and social polls. Understanding existing patterns of social and economic behavior will greatly improve the company's success and efficiency. The employee decides to investigate this thread of information. This browsing may begin with the following queries:

Display SubClasses By Attribute of Class research

Find Information With Attribute date : "1980", publication : "social"

After some refinements, a database is selected. The employee would like to know how these polls are actually conducted and represented. Therefore a demonstration is requested. The employee finds out that the documentation provides these polls on their local hardware and software platform. Furthermore, it also runs on their graphical user interface. The information it provides are self-explanatory and provide a wealth of information. This motivates the employee to use a local function to directly access the providing database to get the actual data.

5. Conclusion

In this paper, we described how interoperability among a large network of autonomous heterogeneous databases is supported by special constructs. The support depends on the architecture. The approach taken by FINDIT provides a mechanism that enables a database to know what other databases contain without violating the autonomy and heterogeneity of the database. The search for information and the databases that contain it, uses a combination of type names, structure, and a demonstration about the information behavior or structure. Future work will investigate the integration of local query language with FINDIT. We plan on using the functional model as a paradigm to build an interface between co-databases and component databases.

References

Bouguettaya, 1993.

Athman Bouguettaya and Roger King, "Large Multidatabases: Issues and Directions," pp. Elsevier Publishers in *IFIP DS-5 Semantics of Interoperable Database Systems (Editors: D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis)*, , Lorne, Victoria, Australia (1993).

Sheth, 1993.

Amit Sheth and Vipul Kashyap, "So Far (Schematically) yet So Near (Semantically)," pp. Elsevier Publishers in *IFIP DS-5 Semantics of Interoperable Database Systems (Editors: D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis)*, , Lorne, Victoria, Australia (1993).

Krishnamurthy, 1991.

Ravi Krishnamurthy, Witold Litwin, and William Kent, "Language Features for Interoperability of Databases with Schematic Discrepancies," *SIGMOD*, ACM, (May 1991).

Saltor, 1993.

F. Saltor, M. G. Castellanos, and M. Garcia-Solaco, "Overcoming Schematic Discrepancies in Interoperable Databases," pp. Elsevier Publishers in *IFIP DS-5 Semantics of Interoperable Database Systems (Editors: D. K. Hsiao,*

E. J. Neuhold, and R. Sacks-Davis), , Lorne, Victoria, Australia (1993).

Bouguettaya, 1992.

Athman Bouguettaya, "A Dynamic Framework for Interoperability in Large Multidatabases," *PhD Thesis in Computer Science*, (May 1992).

Bouguettaya, 1991.

Athman Bouguettaya, Roger King, and Kequn Zhao, "FINDIT: A Server Based Approach to Finding Information in Large Scale Heterogeneous Databases," *First International Workshop on Interoperability in Multidatabase Systems*, pp. 191-194 (April 7-9, 1991).

Bouguettaya, 1993.

Athman Bouguettaya, Roger King, Douglas Galligan, and Jon Simmons, "Implementation of Interoperability in Large Multidatabases," *Third International Workshop on Research Issues on Data Engineering: Interoperability in Multidatabase Systems*, (April 18-20, 1993).

Bouguettaya, 1994.

Athman Bouguettaya, Roger King, and Mike Papazoglou, "On Building a Hyperdistributed Database," *Submitted for Journal Publication*, (1994).

Litwin, 1987.

W. Litwin and A. Abdellatif, "An Overview of the Multi-Database Manipulation Language MDSL," *Proceedings of the IEEE* **75(5)** pp. 621-632 (1987).

Elmagarmid, 1989.

A. K. Elmagarmid, Marek Rusinkiewicz, and Shawn Ostermann, "The Distributed Operation Language for Specifying Multi-System Applications," *Technical Report 52-P, Software Engineering Research Center, Purdue University*, (September 1989).

Elmagarmid, 1990.

A. K. Elmagarmid, Nouredine Boudriga, Eva Kuhn, and Yungho Leu, "A Parallel Logic Language for Transaction Specification in Multidatabase Systems," *Interbase TR-020, Purdue University*, (October 1990).

Batini, 1986.

C. Batini, M. Lenzerini, and S. B. Navathe, "A Comparative Analysis of Methodologies for Database Schema Integration," *ACM Computing Surveys* **18(4)** pp. 324-364 (Dec. 1986).

Templeton, 1986.

M. Templeton, D. Brill, A. Chen, S. Dao, and E. Lund, "Mermaid - Experiences with Network Operation," *Proc. 2nd Data Engineering Conference* **292-300** pp. 292-300 (Feb. 1986).

Landers, 1982.

T. Landers and R. L. Rosenberg, "An Overview of Multibase," *Distributed Databases*, pp. 153-184 North-Holland Publishing Company, (1982).

Smith, 1981.

John Miles Smith, Phillip A. Bernstein, Umeshwar Dayal, Nathan Goodman, Terry Landers, Ken W. T. Lin, and Eugene Wong, "Multibase- Integrating Heterogeneous Distributed Database Systems," *AFIP, National Computer Conference*, pp. 487-499 (1981).

Litwin, 1982.

W. Litwin, J. Boudenat, C. Esculier, A. Ferrier, A. M. Glorieux, J. La Chimia, K. Kabbaj, C. Moulinoux, P. Rolin, and C. Stangret, "SIRIUS System for Distributed Data Management," *Distributed Databases*, pp. 311-343 North-Holland Publishing Company, (1982).

Stocker, 1984.

P. M. Stocker, M. P. Atkinson, P. M. D. Gray, W. A. Gray, E. A. Oxborrow, M. R. Shave, and R. G. Jonhson, "Proteus: A Heterogeneous Distributed Database Project," *Databases- Role and Structure*, pp. 125-150 Cambridge University Press, (1984).

Kim, 1991.

Won Kim and Jungyun Seo, "Classifying Schematic and Data Heterogeneity in Multidatabase Systems," *IEEE Computer* **24(12)** pp. 12-18 IEEE, (December 1991).

Motro, 1981.

Amihai Motro and Peter Buneman, "Constructing Superviews," *ACM SIGMOD*, pp. 56-64 (1981).

Heimbigner, 1985.

D. Heimbigner and D. McLeod, "A Federated Architecture for Information Systems," *ACM Trans. Office Information Syst.* **3(3)** pp. 253-278 (July 1985).

Sheth, 1990.

Amit P. Sheth and James A. Larson, "Federated Database Systems and Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys* **22(3)** pp. 183-226 ACM, (September 1990).

Litwin, 1990.

Witold Litwin, Leo Mark, and Nick Roussopoulos, "Interoperability of Multiple Autonomous Databases," *ACM Computing Surveys* **22(3)** pp. 267-293 ACM Press, (September 1990).