

Using Java and CORBA for Implementing Internet Databases

Athman Bouguettaya^{1*}, Boualem Benatallah^{2†}, Mourad Ouzzani*, Lily Hendra*

* Queensland University of Technology - GPO Box 2334 Brisbane, QLD 4001 Australia

† James Cook University - GPO BOX 6811 Cairns, QLD 4870 Australia

{athman,ouzzani,lhendra}@icis.qut.edu.au boualem@cs.jcu.edu.au

Abstract

We describe an *architecture called WebFINDIT that allows dynamic couplings of Web accessible databases based on their content and interest. We propose an implementation using WWW, Java, JDBC, and CORBA's ORBs that communicate via the CORBA's IIOP protocol. The combination of these technologies offers a compelling middleware infrastructure to implement wide-area enterprise applications. In addition to a discussion of WebFINDIT's core concepts and implementation architecture, we also discuss an experience of using WebFINDIT in a healthcare application.*

1. Introduction

The growth of the Internet and the Web increased dramatically the need for data sharing. The Web has brought a wave of new users and service providers to the Internet. It contains a huge quantity of heterogeneous information and services (home pages, online digital libraries, product catalogs, etc.) [3]. The result is that the Web is now accepted as the *de facto* support in all domains of life activities: finance, education, travel, business, science, healthcare, art, etc.

The data provided in the Web is not only semi-structured (HTML documents, mail messages, etc.) or unstructured (text files, images, etc.) but also structured (databases). It is widely recognized that in many emerging applications such as electronic commerce, best Web sites are database-based Web sites. In addition, "useful" and "sensitive" data (e.g., corporate data) is almost inevitably stored in databases. Organizations and individuals all over the world rely on a large number of heterogeneous information sources (databases, ftp files, Web search engines, and so on) to conduct their everyday business. These information sources are deployed in a wide area network-based environment. In such a highly dynamic environment, there is no formal control over the changes in the information

space, and several types of applications are developed without knowledge of the availability of particular information sources.

One of the most frequently encountered issues in a large cooperative environment, such as database-based Web applications, is how users can query efficiently the large and highly intricate amount of available heterogeneous information sources. Web users are in general novice. They are not expected to have experience dealing with databases or be knowledgeable about available database query languages. In this respect, requiring users to keep track of information such as locations, formats (or structures), content, and query languages of the growing number of dynamic sources is unreasonable. The challenge is to provide across the board transparency to allow users to access and manipulate data irrespective of platforms, locations, systems, etc. We distinguish the following key issues when using Web-resident data:

- Locating appropriate information sources. In Web applications, the information space is very large and dynamic. On top of that, existing Web tools give very little support for the logical organization of data. Thus, the effective use of data in the *anarchic* Web has become enormously complex.
- Understanding the meaning, content, terminology and patterns of use of the available information sources. Users have a need to be *educated* about the information of interest and dynamically know what other databases contain and eventually establish a link to those databases that contain information of interest.
- *Querying* these sources for relevant information items. Once relevant information sources have been found, users have a need to access and integrate data from these information sources.

The WebFINDIT architecture has been developed to address problems of scalability and language support for Internet/Web-based databases. The information space is organized in a topic-based (ontological) clustering of information sources. These clusters are related to each other by topic proximity relationships. Clustering of information sources are established through the sharing of high

¹This research has been partly supported by an Australian Research Council (ARC:) Large Grant number 95-7-191650010.

²This work was done when this author was a postdoctoral fellow at Queensland University of Technology.

level meta-information where individual sites join and leave these clusters at their own discretion.

The WebFINDIT prototype is implemented using the latest in distributed object and Web technologies, including CORBA as a distributed computing platform, Java, and connectivity gateways to access native information sources. This paper describes the architecture, design and implementation concepts of WebFINDIT. It also describes a practical experience using WebFINDIT. Special emphasis will be given to the use of CORBA and Java technologies in the implementation of Internet databases. Section 2 overviews briefly the WebFINDIT's approach and query language. The key middleware technologies (Java and CORBA) used in the WebFINDIT implementation are described in more details in Section 3. The major components of the WebFINDIT architecture are presented in Section 4. We overview our current implementation in Section 5. An experience of using WebFINDIT in healthcare domain is reported in Section 6. Related work is discussed in Section 7. We provide some concluding remarks in Section 8¹.

2. Design of WebFINDIT

In our approach, a user finds information sources in the following manner. Initially, the user specifies the query in terms of relevant information using a special-purpose language. The query is sent to a local metadata repository that holds relevant meta-information about the local database (clusters of the database, relationships, etc.) We assume that a user of our system is already a user of a participating database. If there are several clusters of information sources that advertise the required information, the system prompts the user to select the most interesting leads. If the local metadata repository fails to resolve the user query, using the information on clusters' relationships, the local repository sends the query to one or more remote metadata repositories. During this interactive process the system provides support for educating the user about available information space. Assuming that there are many information sources that are offering the required information, the user will use the meta-data describing information sources to further decrease their number. To assist in the decision making process, WebFINDIT provides support for understanding the information context and semantics. As a last step, the user queries the selected information sources. In what follows, we will briefly describe the WebFINDIT's approach. More details can be found in [4,13].

2.1. WebFINDIT Fundamental Concepts

WebFINDIT is centered around a two-level approach to bridge heterogeneity, accommodate database autonomy, and allow systems consisting of large number of databases

¹In the rest of the paper, we will use Internet databases and Web databases interchangeably.

to scale up. The two-level approach corresponds to coalitions (clusters) and service **links** (relationships). Coalitions are a means for databases to be strongly coupled whereas service links are a means for them to be loosely connected. Users are incrementally and dynamically educated about the available information space. The proposed approach is enabled by the introduction of a layer of meta-data (called co-database) that surrounds each local DBMS. This layer contains information about local DBMSs and their relationships. Finally, a special language called *WebTassili* provides constructs for information space definition, maintenance, and exploration.

Coalitions

A coalition is specialized to a single common topic. It provides domain specific information and terms for interacting within the coalition and its underlying databases, i.e., providing an abstraction of a specific domain. This abstraction is intended to be used by users and other coalitions as a description of the specific domain. Coalitions dynamically clump databases together based on common areas of interest into a single atomic unit. For example, the databases participating in the coalition Medical Workers Union share descriptions of the information type Medical Workers Union (see Figure 1). As database node "interests" change over time, new coalitions may form, old coalitions may be dissolved, and components of existing coalitions change. WebFINDIT takes advantage of the fact that databases are developed with a specific "purpose", and uses this as an implicit organizing principle.

Services links

Service links are a simplified way to share information. They allow sharing with low overhead. The amount of sharing in a service link involves a minimum of information exchange. In this respect, service links are low overhead alternatives to information sharing using coalitions. For example, the service link between Medical and Medical Insurance is an agreement between these two coalitions (see Figure 1). In this case, the coalition Medical provides minimal description of information type Medical to coalition Medical Insurance.

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

As an example of coalitions and service links, we consider databases from the **Medical information domain**.

Figure 1 shows fourteen databases. These database are grouped in five coalitions (Research, Medical, Medical Insurance, etc.) and nine service links (Ambulance to Medical, AT0 to Medical, etc.)

2.2. Co-databases

Locating a set of databases that fit user queries requires detailed information about the contents of each database in the system. In our approach, each participating database has a **co-database** attached to it. A co-database is an object-oriented database that stores information about its associated database, coalitions, and service links. A set of databases exporting a certain type of information is represented by a class in the object-oriented co-database schema. This also means that a coalition is represented by a class. In particular, every class contains a description about the participating databases and the type of information they contain. Some attributes describe a type of information while others provide details about the databases that contain this type of information. Descriptions of the databases will include information about the data model, operating system, query language, etc. Descriptions of the information type will include its general structure and behavior. 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.

For example, the co-database attached to the Royal Brisbane Hospital contains information about all related coalitions and service links. As the Royal Brisbane Hospital is member of two coalitions Research and Medical, it stores information about these two coalitions. This co-database contains also information about other coalitions and databases that have a service link with these two coalitions and the database itself. The co-database stores information about the service links State Government Funding and Medical Insurance. It stores also access information of the Royal Brisbane Hospital database. This includes the exported interface and the Internet address. The interface of a database consists of a set of types containing the exported operations and a textual description of these types. The Royal Brisbane Hospital represents an Oracle database that contains the following relations:

Patient(Patient Id, Name, Date of Birth, Gender, Address)
Beds(Bed Id, Location, Default Patient **Type**)
Occupancy(Bed Id, Patient Id, Date From, Date To)
History(Patient Id, Date Recorded, Description, Description Notes, Doctor Id)
Doctors(Employee Id, Qualification, Position)
ResearchProjects(Project Id, Title, Keywords, Supervising **Doctor**, **Begin Date**, **Completed Date**, **Funding**)
MedicalStudent(Student Id, Name, Course, Year)
ResearchProjectAttendants(Project Id, Student Id, Task, Date Started, Date Completed, Results)

If the database administrator decides to make public some of these relations, they should be advertised through the co-database by specifying the information type, the

documentation (a file containing multimedia data or a program that plays a product demonstration), and the access information which includes the location, the wrapper (program allowing access data in the database), and the set of exported types. Assume that the administrator decides to advertise relations related to Research and Medical. The Royal Brisbane Hospital will be advertised in WebFINDIT as follows:

```
Information Source Royal Brisbane Hospital {
  Information Type "Research and Medical"
  Documentation "http://www.medicine.uq.edu.au/RBH"
  Location "dba.icis.qut.edu.au"
  Wrapper "dba.icis.qut.edu.au/WebTassiliOracle"
  Interface ResearchProjects, PatientHistory
}
```

The URL <http://www.medicine.uq.edu.au/RBH> contains the documentation about Royal Brisbane Hospital database. It contains any type of presentation accessible through the Web (a Java applet that plays a video clip). WebTassiliOracle is the wrapper needed to access data in the Oracle database using a WebTassili query. The exported interface contains two types about research and patients. For example, the PatientHistory type is defined as follows:

```
Type PatientHistory {
  attribute string Patient.Name;
  attribute int History.DateRecorded;
  function string Description(string Patient.Name,
  int Date History.DateRecorded)
```

The function Description0 denotes the access routine that returns the description of a patient sickness at a given date. This routine is written in Oracle's C interface. In the case of an object-oriented database, an attribute denotes a class attribute and a function denotes either a class method or an access routine. Using WebFINDIT, users can locate the database, then investigate its exported interface and fetch useful attributes and functions to access the database.

2.3. WebTassili Language

WebTassili is designed to query (meta)data over the Web organized using WebFINDIT and to manage the evolution of this architecture. The syntax specifications of this language provide constructs to educate users about the available space of information, finding the target databases that are most likely to hold the required type of information, and connecting to databases and performing remote queries. The information metatype name, structure, and behavior are used as a handle for identifying the appropriate information sources. WebTassili provides the following manipulation operations:

- Search for an information type,
- Search for an information type while providing its structure,

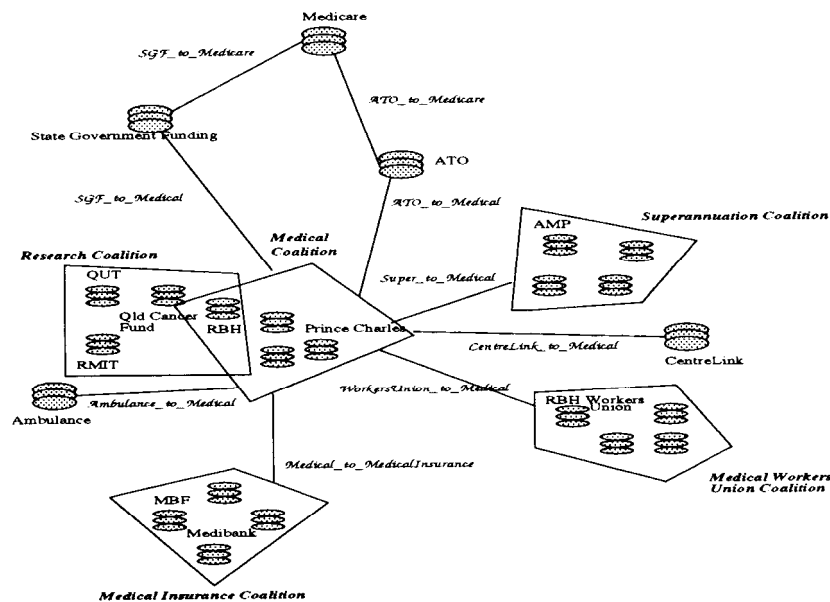


Figure 1. Coalitions and Service Links in the Medical World

- Search for an information type while providing its structure and/or information about the host databases, and
- Query remote databases.

In order to show the salient features of WebTassili, we consider the following example from the Medical information domain represented in Figure 1. Assume that researchers in the QUT research centre are interested in gathering some information about health in Queensland in order to prepare a survey. They are interested in information on hospitals, treatment costs, insurance, etc. One of the researchers at QUT research queries WebFINDIT for medical research conducted in hospitals. For this purpose, the researcher can start the investigation by submitting the following WebTassili query:

```
Find Coalitions With Information Medical Research;
```

In order to resolve this query, WebFINDIT starts from the coalitions the QUT research is member of and checks if they hold the information. The system found that the local coalition Research deals with this type of information. A point of entry is provided for this coalition using the following query:

```
Connect To Coalition Research;
```

As the user is interested in more specific information i.e., research conducted in hospitals, a refinement query is submitted as follows:

```
Display SubClasses Of Class Research;
```

The class Research shows that it contains the subclasses: Medical Research in Hospitals and Medical Research in Universities. Assume that the user is interested in the first subclass, the user then issues a WebTassili query to display instances of this subclass. The user

decides to query the Royal Brisbane Hospital which is an instance of the class Research. Before that, the user can become more knowledgeable about this database using a WebTassili construct that displays the documentation of this information. The query to achieve this goal is:

```
Display Document of Instance Royal Brisbane Hospital  
Of Class Research;
```

At this point, the user is interested in querying this database. The user uses then the following WebTassili query to display the interface exported by the database:

```
Display Access Information of Instance Royal Brisbane Hospital;
```

The database Royal Brisbane Hospital is located at "dba.icis.qut.edu.au" and exports the following type:

```
Type ResearchProjects {  
  attribute String ResearchProjects.Title;  
  attribute string ResearchProjects.keywords;  
  attribute int Date ResearchProjects.BeginDate;  
  function real Funding(ResearchProjects.Title x,  
    Predicate(x));  
}
```

The function Funding() returns the budget of a given research project. For instance, if we are interested in the budget of the research project AIDS and drugs, we use the function Funding(ResearchProjects.Title, (ResearchProjects.Title = "AIDS and drugs")). This function is translated to the following SQL query:

```
Select a.Funding  
From ResearchProjects a  
Where a.Title = "AIDS and drugs"
```

Assume that another researcher is interested in querying the system about medical insurance. The following query is submitted to the system.

WebFINDIT first checks the coalitions the QUT research is member of. The coalition Research fails to answer the query. As there are no other coalitions or service links related to the local database, WebFINDIT checks whether other databases from the local coalition are aware of a coalition or service link that deal with this information type. The system found that the database Royal Brisbane Hospital (which is member of the local coalition) is member of a coalition Medical that has a service link with another coalition Insurance that appears to deal with the requested information type. Therefore, the user decides to investigate this coalition looking for relevant information.

3. Middleware Technologies in WebFINDIT

This section presents the overall architecture which supports the WebFINDIT framework. This architecture adopts a client-server approach to provide services for interconnecting a large number of distributed, autonomous and heterogeneous databases. It is based on *CORBA and Java* technologies. We briefly overview the use of these technologies in the implementation of distributed applications such as Internet databases.

3.1. CORBA

Large scale networked systems need support for data access and communication. Recently, standardization efforts in the architecture of heterogeneous distributed systems have produced maturing infrastructure technologies that address some of these issues. One of the most important standards that have emerged is the OMG's *CORBA* [14]. The *CORBA*'s infrastructure provides mechanisms to deal with platform heterogeneity, transparent location and implementation of objects, interoperability and communication between software components of a distributed object environment. The *IDL* (Interface Definition Language) language is used for the separation between the implementation and the interface of a *CORBA* service. The *CORBA IDL* describes the operations and associated attributes of an object's interface in a way understood by the rest of the system. In essence, each *CORBA* object has an interface defined in *IDL*. In this fashion, it is possible to map functions of various resources such as network devices, databases and other applications into object-oriented interface specifications. *CORBA* allows clients to discover and use new types of objects added to the system without any change to the system.

The issue of interoperability across multi-vendor *CORBA* ORBs is addressed in *CORBA 2.0*. *CORBA 2.0* [14] specifies the General *Inter-ORB Protocol (GIOP)* which provides a set of message formats and data representations for communications between ORBs. Specifically, it defines: (1) the Common *Data Representation (CDR)* as data exchange format between an *OMG IDL* data type and

a flat networked message representation, and (2) the *Interoperable Object References (IORs)* to allow the creation of common object references from ORB specific object references. *GIOP* is designed to work on top of any communication protocol. The *Internet Inter-ORB Protocol (IIOP)* is the specification of *GIOP* over *TCP/IP* network. Hence, the use of *IIOP* allows objects distributed over the Internet, on different ORBs, to communicate. Any *CORBA 2.0* compliant ORB must support *IIOP* or provide a gateway to it.

3.2. Java

One language that is gaining popularity for writing Internet applications is Java. Java is a platform independent object-oriented programming language.

One main feature that makes Java ubiquitous is the notion of applet. The user interface can be provided using Java applets. Users can download the user interface from a Web server using a Java-enabled browser. Thus the user interface is enabled to be available on distributed heterogeneous platforms without any overhead in coding and administration (write once, run everywhere). With regard to distributed applications, several Java technologies (*Java*, *Java Remote Method Invocation - Java RMI*, *JavaBeans*, *Java Database Connectivity - JDBC*, *Java Native Interface - JNI*, and so on) are relevant. These technologies provide object and database access services (*Java-to-Java*, *Java-to-CORBA*, and *Java-to-databases* communication). In particular there are two main reasons that make Java a technology of choice in the context of data sharing. First, *JDBC* is an *ODBC*-like API (a set of Java classes) that provides a generic interface to *SQL* interfaced relational databases (*DB2*, *Sybase*, *SQL Server*, *Paradox*, *Progress*, *Oracle*, *mSQL*, etc.) from Java applications. Second, most *DBMS* vendors provide Java interfaces.

3.3. Binding Java and CORBA

Java and *CORBA* technologies seem to be converging to provide complementary types of services. *CORBA* is concerned with the communication between objects implemented in different languages, using different platforms across the network. Java can be used for the implementation of these objects. In addition Java *RMI* objects can only talk to other Java *RMI* objects since *RMI* is not designed to interoperate with other ORBs and languages. However, tools to bridge Java and *CORBA* are provided in several ways. Thus, Java applications (*JavaBeans*) can access components written in different languages and hosted on a variety of computing platforms. As pointed out before, several *CORBA* ORBs, such as *VisiBroker* for Java and *OrbixWeb*, allow Java applications to access *CORBA* objects. *JavaIDL*, which is a part of *JDK (1.2 beta)*, is another *IIOP* compliant ORB. Java applets can be downloaded onto the user machine and used to communicate with system components, i.e., *CORBA* objects. Java offers a portable infrastructure where an applet may be loaded

and executed from any machine in the Internet to communicate with CORBA services without any runtime configuration. In addition JDBC can be used to access SQL relational databases from Java applications. It is clear that using these two technologies together provides greater flexibility and power than either technology does alone. A robust distributed architecture can be created using Java as programming language, Java applets to provide user interfaces, and CORBA as an integration technology.

Other technologies such as HTTP/CGI approach and ActiveX/DCOM [14] are also used for developing intranet- and Internet-based applications. It is recognized that the HTTP/CGI approach may be adequate when there is no need for sophisticated remote server capabilities and no data sharing among databases is required. Otherwise, Java/CORBA approach offers several advantages over HTTP/CGI. We note also that the CORBA's IIOP and HTTP can run on the same networks as both of them uses the Internet as the backbone. Also, the interoperability between CORBA and ActiveX/DCOM is already a reality with the beta-version of Orbix COMet Desktop.

4. WebFINDIT Architecture

The WebFINDIT components are grouped in four layers that interact among themselves to query Internet databases using a Web-based interface (see Figure 2). In this section we introduce the different layers and their interaction. The basic components of WebFINDIT are the query layer, the communication layer, the *meta-data* layer, and the *data* layer.

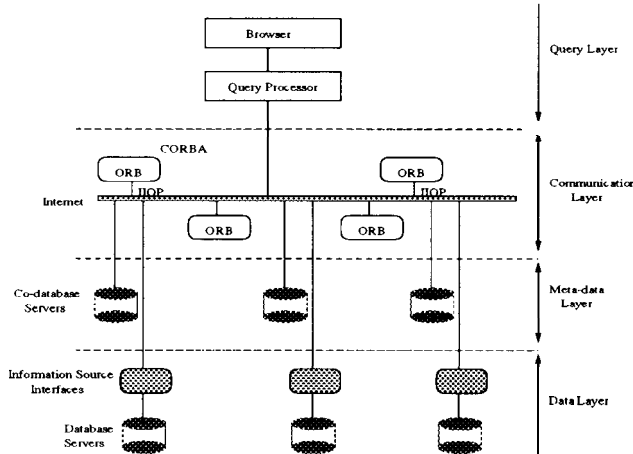


Figure 2. WebFINDIT Layers

4.1. Query Layer

The query layer provides users access to WebFINDIT services. It has two components: The browser and the query processor.

Browser: is the user interface to WebFINDIT. It uses the meta-data stored in the co-databases to educate users about the available information space, locate

the information source servers, send queries to remote databases and display their results. The browser allows users to browse the information using graphical and text queries. It allows users to browse classes or instances. Users are allowed to follow links between objects, to focus on the details of particular objects, and to specify filters which restrict the browsing to particular sets of objects.

Query Processor: receives queries from the browser, coordinates their execution and returns their results to the browser. WebTassili queries submitted by the browser are preprocessed to check the syntax. The query processor follows an algorithm that determines the steps for the query resolution. Each time a query from the browser is received, a new instantiation of the execution plan is initialized. The query is decomposed if needed. The query processor interacts with the communication layer (next layer) which dispatches WebTassili queries to the co-databases (meta-data layer) and databases (data layer).

4.2. Communication Layer

The communication layer manages the interaction between WebFINDIT components. It mediates requests between the query processor and co-database/database servers. The query processor interacts with the servers without knowing where the servers are on the network or how they accomplish their tasks. The communication layer locates the set of servers that can perform the tasks. It responds to the query processor's tasks with information about these servers and information about data stored in these servers.

4.3. Meta-data Layer

The meta-data layer consists of a set of co-database servers that stores meta-data about the associated databases (information type, location, coalitions, service links, and so on). Co-databases are designed to respond to queries regarding available information space and locating sources of an information-type. A typical co-database schema contains subschemas that represent coalitions and service links that deal with specific types of information. The first sub-schema consists of a lattice of classes where each class represents a set of databases that can answer queries about a specialized type of information. This subschema represents coalitions. The co-database also contains another type of subschema. This subschema consists on one hand, of a subschema of service links the coalitions (it is a member of) has with other databases and coalitions; and on 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 and service links with other coalitions.

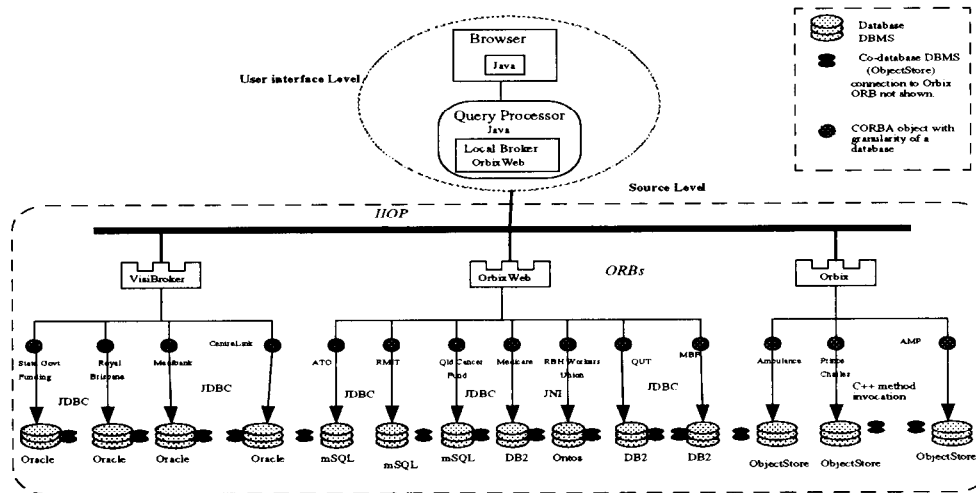


Figure 3. WebFINDIT Implementation.

4.4. Data Layer

The data layer has two components: databases and Information Source Interfaces (ISIs). The current version of WebFINDIT supports relational (mSQL, Oracle, Sybase, DB2) and object oriented databases (ObjectStore and Ontos). An information source interface provides access to a specific database server. This involves delivering requests from the communication layer and retrieving results from this database. We provide the possibility to have an information source interface located at a different site from the database. In this case, an information source interface relies on another gateway protocol (e.g., JDBC) and can be associated to several information sources.

5. WebFINDIT Implementation

We have completed the implementation of a scalable and portable architecture of WebFINDIT. This architecture has been implemented using the latest in object and Web technologies, including CORBA, Java, and database connectivity gateways to access native databases. The prototype that we developed uses three different CORBA ORBs that are IIOP compliant, namely Orbix, OrbixWeb, and VisiBroker for Java. These ORBs connect 28 databases (databases and their co-databases). Each database is encapsulated in a CORBA server object (a proxy). These databases are implemented using five different DBMSs: Oracle, mSQL, DB2, ObjectStore, and Ontos. The JDBC bridge is used to connect relational databases to their CORBA server objects. In this case, the CORBA objects are implemented in Java (OrbisWeb or VisiBroker for Java server objects). The object-oriented databases communicate with the CORBA server objects that are implemented in C++ (Orbis server objects) using C++ method invocation. The object-oriented databases communicate with the CORBA server objects that are implemented in Java (OrbisWeb server objects) using JNI. The user interface is implemented as Java applets that communicate with CORBA

objects.

The current implementation of our system is based on Solaris(2.6), JDK (1.1.5) which includes JDBC (2.0) (used to access the relational databases), three CORBA products, namely Orbix (2), OrbixWeb (3), and VisiBroker (3.2) for Java. ObjectStore databases are connected to Orbix (see Figure 3). The Ontos database is connected to OrbixWeb. Relational databases (stored in Oracle, mSQL, and DB2) are connected to a Java-interfaced CORBA. Oracle databases are connected to VisiBroker, whereas mSQL and DB2 are connected to OrbixWeb. CORBA server objects use JDBC to communicate with relational databases, C++ method invocation to communicate with C++ interfaced object-oriented databases from C++ CORBA servers (both Orbix and ObjectStore support C++ interface), JNI to communicate with C++ interfaced object-oriented databases from Java CORBA servers (from OrbixWeb to Ontos). CORBA objects communicate via IIOP.

6. Using a Healthcare Application

In order to illustrate the viability of this architecture and show how to query global information system using WebFINDIT, we have used a Healthcare application. Healthcare applications provide a very relevant context where tools such as WebFINDIT can be used. The application supports queries about healthcare related services and enable a large number of heterogeneous and autonomous healthcare providers to communicate with each other. In this application, fourteen databases are used: State Government Funding, RBH - Royal Brisbane Hospital, RBH Workers Union, Centre Link, Medibank, MBF, RMIT Medical Research, Queensland Cancer Fund, Australian Taxation Office, Medicare, QUT Research, Ambulance, AMP, Prince Charles Hospital (see Figure 1). Each database is accompanied with its own co-database. The 28 databases (databases and

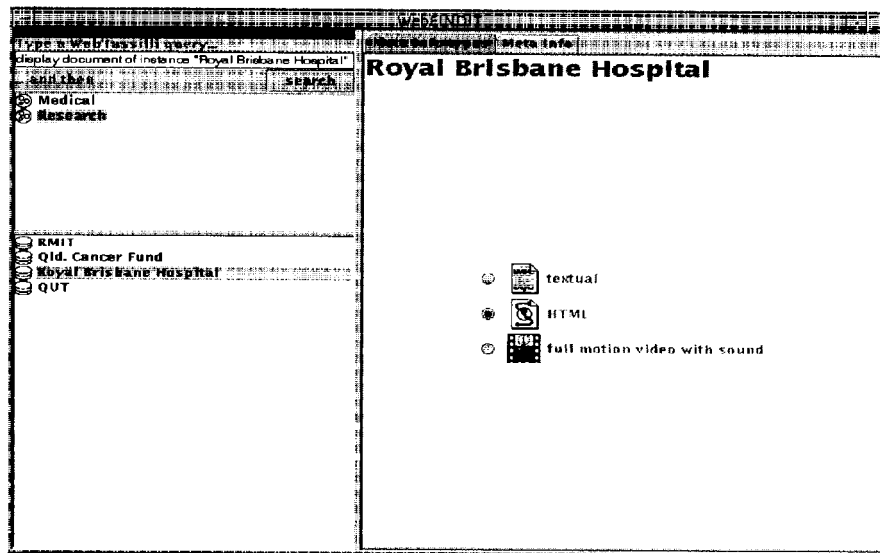


Figure 4. Display Document on RBH Co-Database

their co-databases) are implemented using four different database management systems, namely Oracle, mSQL, DB2, ObjectStore, and Ontos.

As pointed out before, users in WebFINDIT query the system at two levels: meta-data level (explore the available information, display meta information about a particular database, and so on) and data level (query actual information stored in databases). Typically, a user of this application starts by submitting queries about specific area in the healthcare domain. As an example, the following WebTassili query “*Display Coalitions With Information Medical Research*” is submitted. The system finds that both coalitions Medical and Research provide information about Medical and Research. The user can then browse these coalitions.

Suppose that the user wishes to display all members (databases) of the Research coalition. This can be done either by clicking on the Research coalition or by submitting the WebTassili query “*Display Instances of Class Research*”. The user can view the result in the lower half of the left hand side window of the Figure 4.

To know more on a particular database, the user can click on this database. For example, when the user clicks on the Royal Brisbane Hospital database, the available formats of documentation is displayed (e.g., text, HTML) in the right hand side window of the Figure 4. Note that the user can type the WebTassili query “*Display Documentation of Instance Royal Brisbane Hospital of Class Research*” to display the same result. If the user decides to read the documentation using HTML, he/she clicks on the HTML button. Figure 5 displays the content of the HTML file containing the documentation of Royal Brisbane Hospital database.

So far, the user has only used the system to query meta-data. Assume that after locating and understanding the content of the Royal Brisbane Hospital database, the user decides to query some actual data in this database. Querying actual data is performed with WebTassili queries (if supported by the target database); or directly by using

the database’s native query language. In the first case, the WebTassili query is mapped to an equivalent query in the underlying database. Assume that the user wants to know about the Medical Students who are doing internships in the hospital Royal Brisbane Hospital). As the underlying database support SQL, the user can use SQL statement “*select * from medical-students*” to get the required information. Once the definition of the query is accomplished, the query is submitted for execution by clicking on the Fetch button. Figure 6 shows the result of the query.

7. Related Work

There is a large body of relevant literature on information extraction, access, and integration. We consider those that are most closely related to our work, namely, *multidatabases* [6], *WWW information retrieval systems* [7], and information *brokering* systems [8].

7.1. Multidatabases

Multidatabases have traditionally investigated static approaches to sharing data among small numbers of component databases [6]. This has involved finding solutions to data heterogeneity and facets of autonomy. These solutions usually rely on centralized database administrators to document database semantics or to develop translators that hide differences in query languages and database structures. Tightly-coupled approaches offer better solutions for the heterogeneity problem by using a global schema [3]. However, this scheme does not provide site autonomy nor does it scale-up given the complexity when constructing the global schema for a large number of heterogeneous systems. The MIND project investigated the use of CORBA to implement a tightly-coupled interoperability approach [15]. Loosely-coupled approaches offer better solutions for autonomy but they expect users to know the semantics and locations of the available systems [6]. This assumption is not reasonable in Web-based environments.

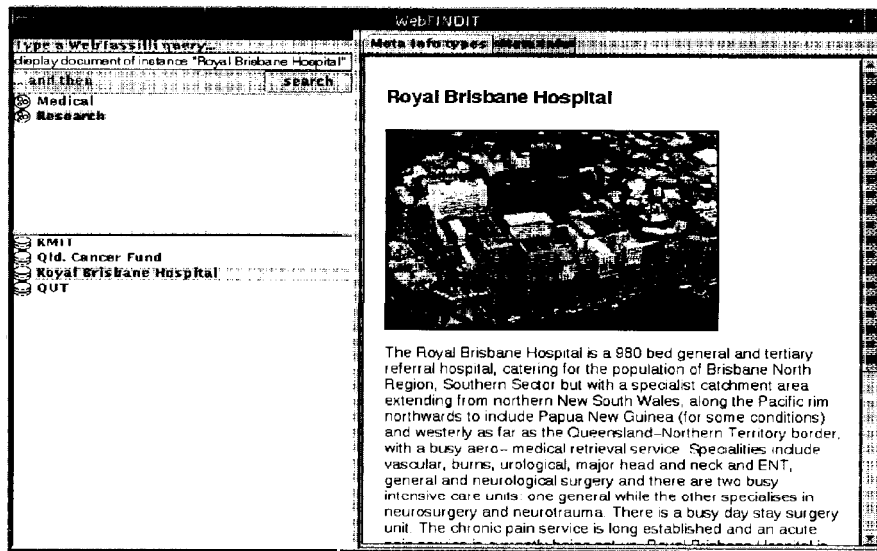


Figure 5. RBH HTML document displayed

7.2. WWW Information Retrieval

In most information retrieval systems, the emphasis is usually on how to build an indexing scheme to efficiently access information given some hints about the resource [7]. Issues like the information space organization, terminological problems, and semantic support for users requests are not addressed. For instance, *Harvest* [5] is an information gathering that presents an interesting model for finding resources in a network of computer systems. As the research is conducted from a system's point of view, databases issues are simplified. Another approach that addressed the issue of information discovery on the Web is proposed by the database community. The idea is to provide a uniform and declarative interface for data sharing on the Web. Several proposals of database-like languages for the WWW have recently emerged (e.g., *W3QL* [9], *WebSQL* [11], and *ARANEUS* [1]). These languages tend to abstract the unstructured collection of Web documents using a graphical organization (Web pages are represented as nodes in a graph with a fixed set of attributes, using one single type). Some combinations of textual retrieval with structure and topology-based queries are supported. The proposed techniques are mainly based on information retrieval systems and as a result they cannot focus on database related topics.

7.3. WWW Information Brokering

Information brokering-based systems investigated solutions for data sharing in the context of a large and dynamic information space. Here a component can be a structured (e.g., relational database), semi-structured (e.g., HTML documents), or unstructured (e.g., text files) source. Existing systems focused mostly on unstructured and semi-structured data. They propose interesting capabilities in mediations and translations. However, these systems lack facilities for information organization, user education and information source location. In the remainder of this section, we briefly overview some of the most important projects.

The TSIMMIS project [16] proposed a new data model, called the *Object Exchange Model (OEM)*, for integration of heterogeneous information sources that may include both unstructured and semistructured data. TSIMMIS primarily focused on the semi-automatic generation of wrappers and mediators that allow the integration and access to underlying information sources when processing OEM-based queries. However, the issues of information discovery, information space organization, and terminological problems are not tackled.

The DISCO project [17] uses an extension of *ODMC-93* and *OQL* as common data model and query language. It provides support for unavailable information sources and transparent addition of new information sources. As in TSIMMIS, the issues of information discovery, information space organization, and terminological problems are not tackled in DISCO. The WebSemantics project [12] extends DISCO by providing a protocol and architecture for locating data sources and translators.

Information Manifold (IM) [10] is a system that provides uniform access to collections of heterogeneous information sources on the WWW. It provides a high-level query system that describes the content and capabilities of various information sources. The domain model is the common global knowledge base that describes the browsable information space including the vocabulary of a domain, the contents of information sources and the capability of querying. We argue that it is difficult to create and maintain such common ontology because of the variety and characteristics of the underlying Web repositories.

The InfoSleuth project [2] presents an approach for information retrieval and processing in a dynamic Web-based environment. It integrates agent technology, domain ontologies, and information brokering to handle the interoperation of data and services over information networks. Although this system provides an architecture that deals with scalable information networks, it does not provide facilities for user education and information space organization. InfoSleuth supports the use of several domain on-

The screenshot shows a web browser window with the address bar displaying 'jdbc:corba:thin:@ba1.cis.qui.edu.au:1521:RBH'. The main content area contains a text input field with the SQL query 'SELECT * FROM medical_students'. Below the input field is a table with the following data:

STUDENT_ID	NAME	COURSE	YEAR	EXPECTED_YEAR	COMPL.
21276458	Pamela McCombe	Bachelor of Medical Studies	4	1999	
34791642	Kim Nyugen	Bach. of Medicine and Surgery	3	2000	
54341324	M.P. Pender	Bachelor of Medical Science	2	2002	

Below the table, there is a checkbox labeled 'To remember the connection information.' and an 'Update' button. On the right side of the interface, there are buttons for 'Configure...', 'Fetch', and 'Update'.

Figure 6. Query Result on RBH Database

tologies, however, the inter-ontology relationships are not considered.

8. Conclusion

Our experience with WebFINDIT project has shown that the combination of CORBA, Java, and JDBC offers a very useful middleware infrastructure to implement Web-resident data sharing architectures. CORBA combined with meta-data repositories (co-databases) provides support for dynamic location and integration of information sources while maintaining their autonomy. Java allows our system to be deployed dynamically over the Web and provides users with sophisticated interfaces to use it. JDBC is a simple API that can be used to access relational databases from Java applications. Most database vendors have recently announced their own Java clients. In addition, most of relational database products provide JDBC and ODBC drivers.

References

- [1] P. Atzeni, G. Mecca, and P. Meriardo. To weave the Web. In 23th VLDB'97, Athens, 1997.
- [2] R. Bayardo, B. Bohrer, R. Brice, A. Cichocki, G. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, Unruh, and D. Woelk. InfoSleuth: Semantic integration of information in open and dynamic environments. In Proceedings of the ACM international Conference on Management of Data (SIGMOD), 1997.
- [3] A. Bouguettaya, B. Benatallah, and A. Elmagarmid. Interconnecting Heterogeneous Information Systems. Kluwer Academic Publishers (ISBN 0-7923-8216-1), 1998.
- [4] A. Bouguettaya, M. Papazoglou, and R. King. On building a hyperdistributed database. *Information Systems, an International Journal*, 20(7):557-577, 1995.
- [5] C. Bowman, P. Danzig, U. M. M. Schwartz, D. Hardy, and D. Wessels. Harvest: A scalable, customizable discovery and access system. Technical report, University of Colorado, Boulder, 1995.
- [6] O. Bukhres and A. K. Elmagarmid, editors. *Object-Oriented Multidatabase Systems: A Solution for Advanced Applications*. Prentice Hall, Englewood Cliffs, New Jersey, 1996.
- [7] V. Gudivada, V. Raghavan, W. Grosky, and R. Kananagottu. Information Retrieval on the World Wide Web. *IEEE Internet Computing*, 1(5):58-68, September 1997.
- [8] V. Kashyap. *Information Brokering over Heterogeneous Digital Data: A metadata-based approach*. PhD thesis, New Brunswick, The State University of New Jersey, October 1997.
- [9] D. Konopnicki and O. Shmueli. W3QS: A query system for the World Wide Web. In *Proceedings of the 21th VLDB*, pages 54-65, 1995.
- [10] A. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of 22nd Int. VLDB Conference*, Bombay, 1996.
- [11] A. O. Mendelzon, G. A. Mihaila, and T. Milo. Querying the World Wide Web. In *Proceedings of the Parallel and Distributed Information Systems (PDIS)*, pages 80-91, 1996.
- [12] G. A. Mihaila, L. Rashid, and A. Tomasic. Equal Time for Data on the Internet with Websemantics. In *Proceedings of the International Conference on Extending Database Technology (EDBT'98)*, Valencia, Spain, 1998.
- [13] 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 (VLDB)*, Zurich, Switzerland, September 1995.
- [14] R. Orfali and D. Harkey. *Client/Server Programming with JAVA and CORBA*. John Wiley & Sons, Inc., 1997.
- [15] F. Ozcan, S. Nural, P. Koskal, C. Evrendilek, and A. Dogac. Dynamic Query Optimization in Multidatabases. *Data Engineering*, 20(3):38-45, September 1997.
- [16] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *Proceedings of the International Conference on Data Engineering*, 1995.
- [17] A. Tomasic, L. Raschid, and P. Valduriez. Scaling heterogeneous databases and the design of DISCO. In *Proceedings of the Int. Conference on Distributed Computer Systems*, 1996.