World Wide Database - Integrating the Web, CORBA and Databases

Athman Bouguettaya and Boualem Benatallah and Lily Hendra
and James Beard and Kevin Smith and Mourad Ouzzani
Queensland University of Technology
School of Information Systems GPO Box 2434 Brisbane, QLD, 4001 Australia
{athman,boualem,lhendra,james,ksmith,ouzzani}@icis.qut.edu.au

1 Introduction

While most of the data published on the Web is either semi-structured (e.g., HTML documents) or unstructured (e.g., text files, images), the Web also offers "hooks" to access non-Web centric structured data (e.g., relational databases). CGI scripts are usually used to access back-end databases. The Web has so far been *incongruous* with databases. The reason that the Web is *database unfriendly* is that it has been developed for *open* data sources. Databases are *closed* in nature in that communication with them is through a rigid protocol (DBMS). One needs to know the schema of a database to access or modify its state. This is fundamentally different from the openness and freeform type of Web data. Web protocols and search engines have been developed for this kind of requirements and environments. Therefore, it is important to note that information retrieval and search techniques could not be applied because of the different nature of, and the fundamental assumptions about the data. However, we could not discount the fact that the Web has made it now possible to have one single interface to potentially access all Internet accessible databases. The challenge now is to make the Web *database friendly*. This essentially means that we need to build an adequate infrastructure on top of the Web that will provide for a *World Wide Database* (WWD).

In order to address problems of scalability and language support for Web accessible databases, the WebFINDIT prototype has been developed [3, 1]. WebFINDIT is a system for describing, locating and accessing data in Internet-accessible databases. We present an architecture and supporting tools that enable users to build complex and emerging Web applications in a simple and flexible way. The main idea is to incorporate simple access to Web-accessible databases as well as support for scalability and extensibility into a flexible interoperable architecture for large information spaces. Most existing techniques focused extensively on either data sharing among small number of heterogeneous databases or on information discovery and brokering in the context of unstructured or semi-structured Web-resident data [1]. In our research, we consider issues to achieve the database equivalent of the WWW. We call it the *World Wide Database* (WWD). The WWD will be fundamentally different from the WWW in that databases are relatively closed as compared to traditional Web-resident data. Therefore, we need to devise an infrastructure that will recognize the specificities of accessing databases on the Web. Any approach to support the WWD must support both metadata and data queries.

We would like to note that the issues in achieving a WWD are very much in line with those enumerated by a committee of prominent and leading database researchers gathered to assess the current state of the database technology and where it should be by the year 2010 [2].

2 Design Overview

Our approach is mainly motivated by the fact that in a dynamic and constantly changing network of databases (e.g., Web accessible databases), there is a need for a meaningful organization and segmentation of the information space. Key criteria that have guided our approach are: scalability, design simplicity, and easy to use structuring mechanisms based on object-orientation. WebFINDIT aims to achieve scalability through the incremental formation and discovery of inter-relationships between Web accessible databases. Clusters (groupings) of databases are established through the sharing of high level meta-information, and individual databases join and leave these clusters at their own discretion. Users are incrementally and dynamically educated about the available information space rather than being presented with all

available information at once. We take advantage of this architecture by using the information meta-type as a unit of data sharing. As a result, databases need only show and map the information types that need to be shared.

A Two-level Approach for Information Space Organization. In order to reduce the overhead of locating information in large networks of databases, the information space is organized as information type groups. Each group forms a *coalition* to represent the domain of interest (some portion of the information space) of the related information sources. It also provides the terminology for formulating queries involving a specific area of interest. Coalitions dynamically clump databases together based on common areas of interest into a single atomic unit.

Coalitions (first level) are related to each other by service links (second level). A *service link* contains only the portions of information that are directly relevant to information exchange between coalitions. They constitute the resources that are available to a coalition to answer requests when they cannot be handled locally. The amount of sharing in a service link will typically involve a minimum amount of information exchange. In this respect, service links are low overhead alternatives to information sharing using coalitions.

Documentation is provided to document the behavior of the information types being advertised. Actual databases are responsible for generating and storing the *documentations* of the information they are advertising. A documentation consists of a set of *demonstrations* about the advertised item.

Metadata Repositories. Locating a set of databases that fits 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 (metadata repository) is an object-oriented database that stores information about its associated database, coalitions, and service links of this database. A set of databases exporting a certain type of information is represented by a class in the co-database 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 other attributes describe the databases that contain this type of information. We should also mention that the documentation (demo) associated with each information instance is stored in actual databases. This is done for two reasons: (1) database autonomy is maintained and, (2) documentations can be modified with little or no overhead on the associated co-databases.

Queries over a Web of Databases. The WebTassili language is designed to query the system at two levels: metadata level (explore the available information, display meta information about a particular database, and so on) and data level (query actual information stored in databases). Beside the standard notions of a query language, 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, as well as for connecting databases and performing remote queries. The information meta-type name, structure, and behavior are used as a handle for identifying the appropriate information sources. WebTassili differs from traditional query languages in that it operates in a large and highly dynamic network of heterogeneous databases. Since the unit of information sharing is the type, this query language is able to query higher order information in addition to querying actual information.

3 Implementation

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. CORBA provides a robust object infrastructure for implementing distributed applications including multidatabase systems. These applications are constructed seamlessly from their components (e.g., legacy systems or newly developed systems) that are hosted at different locations on the network and developed using different programming languages and operating systems. Java allows user interfaces to be deployed dynamically over the Web. Java applets can be downloaded onto the client machine and used to communicate with WebFINDIT components (e.g., CORBA objects).

The WebFINDIT components are grouped into four layers which interact among themselves to query a large number of heterogeneous and distributed databases using a Web-based interface. The basic components of WebFINDIT are the *query layer*, the *communication layer*, the *metadata layer*, and the *data layer*.

The *query layer* provides users with access to WebFINDIT services. It has two components: The browser and the query processor. The browser is the user's interface to WebFINDIT. It is implemented using Java applets. The query processor receives queries from the browser, coordinates their execution and returns their results to the browser. The query processor is written in Java.

The *communication layer* manages the interaction between WebFINDIT components. This component is implemented using a network of CORBA ORBs that communicate via IIOP. The query processor

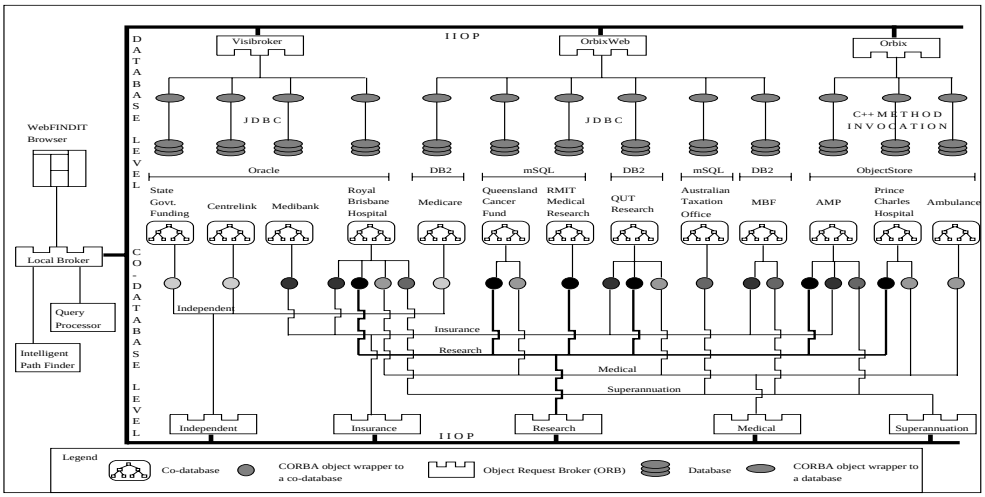

Figure 1: WebFINDIT Architecture

communicates with CORBA ORBs either directly when the ORB is a client/server Java ORB or via another Java ORB.

The *meta-data layer* consists of a set of co-database servers that stores meta-data about the associated databases. All co-databases are implemented in ObjectStore (C++ interface).

The *data layer* has two components: databases and Information Source Interfaces (ISIs). An information source interface (wrapper) provides access to a specific database server.

Software Environment. The prototype that we developed uses three IIOP compliant CORBA ORBs, namely Orbix, OrbixWeb, and VisiBroker for Java. These ORBs interconnect 13 databases (and their co-databases). Each database (and co-database) is encapsulated in a CORBA server object. These databases are implemented using four different DBMSs: Oracle, mSQL, DB2, and ObjectStore. The JDBC bridge is used to connect the CORBA server objects to their relational databases. In this case, the CORBA objects are implemented in Java (OrbixWeb or VisiBroker for Java). The object-oriented databases communicate with the CORBA server objects that are implemented in C++ (Orbix). The user interface is implemented as a Java applet that communicates with the CORBA objects. The current implementation of our system is based on the Java Development Kit version 1.1.5 (which includes JDBC version 2.0), three CORBA products (Orbix version 2.3c, OrbixWeb version 3.0, and VisiBroker version 3.2 for Java). ObjectStore databases are connected to Orbix ORBs. Oracle databases are connected to VisiBroker, whereas mSQL and DB2 are connected to OrbixWeb (see Figure 1).

Using a Healthcare Scenario. A Healthcare scenario has been developed to demonstrate the viability of the proposed WebFINDIT architecture. 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 one another.

4 Conclusion

Our experience with the WebFINDIT project has been 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 an API that can be used to access relational databases from Java applications.

Acknowledgment

The first author would like to acknowledge the support of the Australian Research Council (ARC) through a Large Grant number 95-7-191650010.

References

[1] A. Bouguettaya, B. Benatallah, and A. Elmagarmid. *Interconnecting Heterogeneous Information Systems*. Kluwer Academic Publishers (ISBN 0-7923-8216-1), 1998.

[2] P. Bernstein *et al.* The asilomar report on database research. *ACM SIGMOD Record*, 27(3), December 1998.

[3] S. Milliner, A. Bouguettaya, and M. Papazoglou. A scalable architecture for autonomous heterogeneous database interactions. In *Proceeedings of the VLDB Conference*, Zurich, Switzerland, September 1995.