An Evaluation of Distributed Database Tools

Begumhan Turgut, Ramez Elmasri, Nevin Aydin, and Damla Turgut
Department of Computer Science and Engineering
The University of Texas at Arlington
P.O. Box 19015
Arlington, TX 76019-0015
Email: {bturgut, elmasri, aydin, turgut}@cse.uta.edu

ABSTRACT

As the sharing and exchange of resources increase rapidly, the reliable way of accessing and maintaining these data have become extremely important. Because of the diversity of the types of information stored, and the fact that location of the users can vary greatly, the need for distributed database systems that can support these requirements have been in the highlights. Users from various parts of the world with different platforms or databases that are possibly working on the same or similar areas should be able to share the information among each other without being concerned about either the location or the format of the resource. There are various commercial tools that help bring a solution to this area to some degree depending on the different requirements they have been based on. Here, the author tries to come up with requirements that are essential to every distributed database tool and based on these presents an evaluation of three of the Distributed Database System (DDS) tools namely Codebase 6, Attunity, and Mariposa according the requirements that we have proposed. According to our evaluation, we have concluded that Codebase 6 accommodates most of our requirements.

1. INTRODUCTION

The recent growth of the Internet as a major source of resource storage and retrieval has led to storage of information in a reliable database. The main problem is the number of transactions that can be supported by a database system i.e., the number of hits it can handle. This has led to the use of many sources of data that can be accessed. In a Distributed Database System (DDS), the data is stored in regions that are distributed geographically and can be accessed either locally or remotely.

Distributed database system can be defined as a collection of multiple, logically interrelated databases distributed over a computer network [7]. The DDS is essentially a collection of co-operating nodes. An authorized user can access any one of the nodes in the system. The concept of DDS introduced some major advances in the field of multimedia development. It also influenced the scalability issues in the system. The security issues that drive many systems were found to be suitable in a DDS. The main problem when data is distributed among different systems is the read requests. Since it is difficult to maintain uniformity, the DDS supported the access of systems that were stored in different environments. It provides other efficient features like location transparency, partition transparency, query optimizing, portability, etc. The reason for the popularity of a DDS is due to the high level of availability, the data stored can be accessed from remote nodes. Diverse users can share the data. It is possible to keep track of the configuration information such as location of files, etc.

A Distributed Database Management System (DDMS) is used to develop applications that can create and maintain a DDS [11, 14]. A Tool to develop, maintain and manage a DDS must support the features such as concurrency control, distribution of the load among the systems, and maintenance of a secure environment by allowing only authorized access to the system. A tool to analyze a system that provides DDS capabilities would have to address the different problems that a DDS encounters and has to provide means to solve them successfully. There is an acute need for tools that aid in the development of systems that can help maintain DDS.

One of the alternative solutions is to use a database server with connection to the central server from different remote sites. The other options are to use network computers at the remote sites to access high-speed servers or to use a client/server architecture approach. The client/server approach involves each server sharing information, which is the most reasonable one to use as it addresses the scalability, availability and performance at the remote sites without a need to upgrade the main server to a high-end server.

It is also important to differentiate the difference between replicated vs. distributed databases. In a *pure* i.e., non-replicated distributed database, the system manages a single copy of all data and supports the database objects. Typically, distributed database applications use distributed transactions to access both local and remote data and modify the global database in real-time. The term *replication* refers to the operation of copying and maintaining database objects in multiple databases belonging to a distributed system. While replication relies on distributed database technology, database replication offers applications benefits that are not possible within a pure distributed database environment. Most commonly, replication is used to improve local database performance and protect the availability of applications, because alternate data access options exist. For example, an application may normally access a local database rather than a remote database server to minimize network traffic and achieve maximum performance. Furthermore, the application can continue to function if the local server experiences a failure, but other servers with replicated data still remain accessible.

The organization of the paper is as follows. The concept of Distributed Database Systems is explained in detail in section 2. This is followed by the Distributed Database Management Systems in section 3. Section 4 discusses the evaluation of the DDS Tools and finally section 5 presents the conclusions.

2. DISTRIBUTED DATABASE SYSTEMS (DDS)

In this section, we explain the concepts of a DDS. We present an overview of the architecture of a DDS along with the different types: homogeneous and heterogeneous systems. We discuss the client/server architecture, database links, and the requirements for distributed database systems.

2.1 Architecture of Distributed Database Systems

A distributed database system allows applications to access data from local and remote databases. In a *homogenous* distributed system, all the databases are of the same type i.e., ORACLE or SYBASE. In a *heterogeneous* distributed system [14], the system can contain more than one type of databases. Distributed databases use a *client/server* architecture to process information requests.

Homogeneous Distributed Database System

A *homogenous distributed database system* is a network of two or more databases of the same type. The Databases reside on one or more machines. Figure 1 illustrates a distributed system that connects three databases: HQ, MFG, and SALES. An application can simultaneously access or modify the data in several databases in a single distributed environment [6]. For a client application, the location and platform of the databases are transparent. The synonyms can be created for remote objects in the distributed system so that users can access them with the same syntax as local objects. In this way, a distributed system gives the appearance of native data access. Users on MFG do not have to know that the data they access resides on remote databases. If different versions of the same type of databases are used, they are considered compatible.

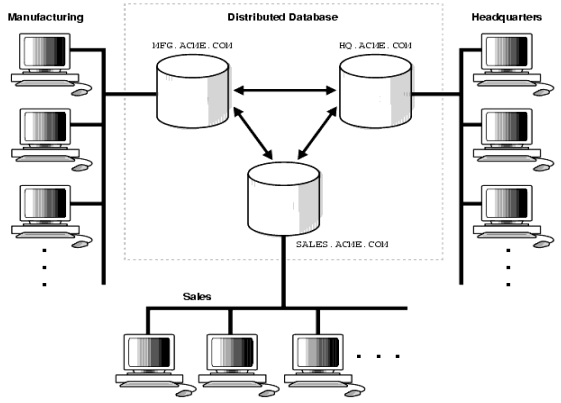

Figure 1. Homogeneous Distributed Database System [6]

Heterogeneous Distributed Database Systems

In a *heterogeneous distributed database system*, more than one database type is used i.e., ORACLE and SYBASE. To the application, the heterogeneous distributed database system appears as a single, local database. The local database server hides the distribution and heterogeneity of the data. A database server can access a remote database system using provided heterogeneous services [12, 13]. To access a remote database of a different type, the system needs to obtain a transparent gateway agent to communicate. For instance, if you access the non-Oracle data store using an Oracle Transparent Gateway, then the agent is a system-specific application. If you include a Sybase database in an Oracle distributed system, then you need to obtain a Sybase-specific transparent gateway so that the Oracle databases in the system can communicate with it [6].

Client/Server Database Architecture

A database server is the software managing a database, and a client is an application that requests information from a server. Each computer in a network is a node that can host one or more databases. Each node in a distributed database system can act as a client, a server, or both, depending on the situation. In Figure 2, the host for the HQ database is acting as a database server when a statement is issued against its local data for instance, the second statement in each transaction issues a statement against the local DEPT table. However, it is acting as a client when it issues a statement against remote data for instance, the first statement in each transaction is issued against the remote table EMP in the SALES database. A client can connect directly or indirectly to a database server. A direct connection occurs when a client connects to a server and accesses information from a database contained on that server. In contrast, an indirect connection occurs when a client connects to a server and then accesses information contained in a database on a different server [6].

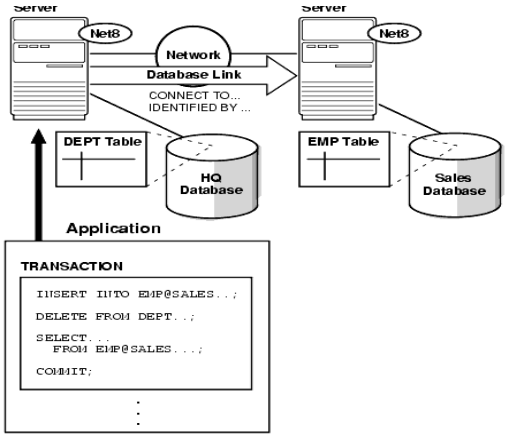

Figure 2. Client/Server Database Architecture [6]

Database Links

Database *links* are one of the most essential topics within Distributed Database System in which is a connection between two physical database servers that allow a client to access them as one logical database. A database link is a pointer that defines a one-way communication path from one-database server to another database server. The link pointer is actually defined as an entry in a data dictionary table.

To access the link, the local database to which the user has been connected must have an entry in the data dictionary. A database link connection is one-way in the sense that a client connected to local database 1 can use a link stored in database 1 to access information in remote database 2, but users connected to database 2 cannot use the same link to access data in database 1. If local users on database 2 want to access data on database 1, then they must define a link that is stored in the data dictionary of database 2. A database link connection allows local users to access data on a remote database. For this connection to occur, each database in the distributed system must have a unique *global database name* in the network domain. The global database name uniquely identifies a database server in a distributed system. Database links are either *private* or *public*. If they are private, then only the user who created the link has access; if they are public, then all database users have access. One principal difference among database links is the way that connections to a remote database occur. Figure 3 shows the database link.

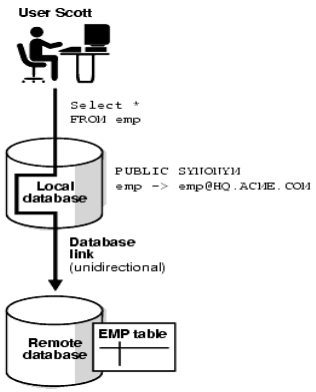

Figure 3. Database Link [6]

Users can access a remote database through one of the following different links:

- *Connected user link* connects as them, which means that they must have an account on the remote database with the same username as their account on the local database.
- *Fixed user link* connects using the username and password referenced in the link.
- *Current user link* connects as a global user. A local user can connect as a global user in the context of a stored procedure without storing the global user's password in a link definition.

Some of the advantages of using a Database Link are as follows:

- They allow users to access another user's objects in a remote database so that they are bounded by the privilege set of the object's owner. In other words, a local user can access a link to a remote database without having to be a user on the remote database.
- They allow the user to grant limited access on remote databases to local users. By using current user links, the user can create centrally managed global users whose password information is hidden from both administrators and non-administrators.
- By using fixed user links, the user can create non-global users whose password information is stored in unencrypted form in the data dictionary table. Fixed user links are easy to create and require low overhead because there are no SSL or directory requirements, but a security risk results from the storage of password information in the data dictionary.

2.2 Requirements for Distributed Database Systems

These requirements are completely based on the domain pertinent to our class of applications. These requirements are the basis for the evaluation of the tools that helped develop DDS.

1. Data should be made available to authorized users locally or remotely. This property indicates that the data stored in a system must be made available locally and remotely.The system must be capable of handling multiple environments, i.e., the system must be capable of accessing data from a UNIX environment as well as a windows environment.Access to data must be consistent. This is a major issue when we distribute a database by replication.The data stored must be accessible only by authorized users. This addresses one of the security concerns in such a system.
5. The system must be reliable i.e. if one of the servers' fail then the other servers must share the load.
6. The system must have a fast response time when handling read or write requests.The access to data must be logged to detect unauthorized intrusion and detect tampering. This is again another security feature. This feature is present to detect tampering.The system must handle large volumes of data. Usually most databases are very large and contain millions of records, so the system must handle large volume of data.
9. The system must be capable of presenting different views of the tables to various users. The system must also provide location transparency.
10. The system must have tools for the management and control of the different servers. This is an important requirement as a DDS is very complex.
11. The system must be easy to install and have good online help. These are some of requirements that would help the user.The data should be made

accessible from the Internet. This is a feature to access the DDS from the Internet.The system must be portable, i.e., we must be able to move it between different environments.

3. DISTRIBUTED DATABASE MANAGEMENT SYSTEMS

Some of the key terms used are briefly described below:

- **DDB:** is a distributed object-relational database management system. We will use it as an example of implementing the distributed database for multimedia applications.
- **DDB ORDB:** is an object-relational database management system running on various platforms. Because of its object-oriented extension, it is capable of storing various multimedia data structures such as pictures, video, audio, etc., together with methods (procedures) that operate on them. There are a number of predefined classes (for C++ and Smalltalk) that can handle multimedia data. This enables developers to focus on building application rather than thinking about the internal structure of multimedia data and converting those data into a format suitable for database storage.
- **DDB Hub:** This is another component of the DDB that enables building the distributed multidatabase management system. Besides supporting the native DDB ORDB, it provides drivers to access existing database systems, including prerelational, relational and object-oriented DBMSs.

The system is available on various platforms, giving the user freedom to choose the appropriate platform for each location. ANSI SQL database language is fully supported ensuring the immediate knowledge of data definition and manipulation. Database language has been extended with object-oriented data definition, enabling the database to store both the object's properties and methods' definition [3]. Database system is accessed through a number of interfaces ensuring the versatility of application development tools used for building the multimedia applications. The scheme of components is as shown in Figure 4. Connecting the individual local databases into a single distributed database becomes increasingly important because users demand a unified view of the data. With the advent of distributed database management systems, these demands can now be satisfied.

One of the most important capability of distributed database systems, with the respect to multimedia applications, is the ability to run the database software on different platforms, because not all the hardware and software platforms are suitable for multimedia applications. We believe that the ability to run distributed database system on different platforms, as well as the ability to compose the distributed database with different database systems, is of greatest interest for those already having local multimedia databases. This enables the preservation of previous investment in hardware, software, people and knowledge.

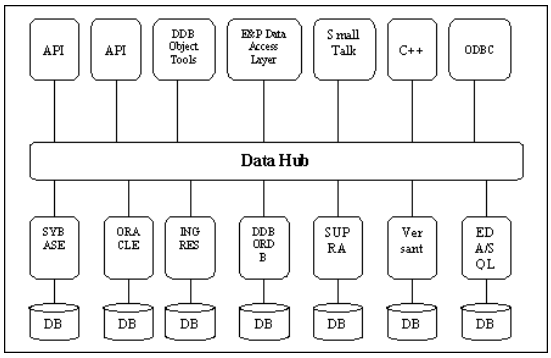

Figure 4. Scheme of Components in DDB [3]

Another very important capability is location transparency that enables users to perceive the distributed database as any local database, while holding the multimedia data (requiring huge storage space) at the location where it is needed most of the time and being accessible from all other locations. At the same time, multimedia data is no longer isolated from other data, ensuring the data integrity between multimedia and conventional data.

Object-oriented extension is included in some of the distributed database management systems, which enables simplified handling of the complex multimedia data while providing the additional reason to finally integrate both conventional and multimedia databases into one logical database [9, 10].

4. EVALUATION OF DDS TOOLS

We have evaluated the three distributed database tools: Codebase 6, Attunity, and Mariposa according to the requirements that we specified in section 2.2. Table 1 presents the capabilities of each of these tools. Therefore, we concluded that CodeBase6 was the tool best suited to meet our needs.

Other methods such as formal methods have been proposed to help evaluate software tools to develop DDS. Some of the Formal Evaluation Techniques mentioned in [5] are as follows.

- Performance Modeling of Distributed and Replicated Databases
- Analytical Models
- Simulations
- Parameter Values
- Default Homogeneity Assumption
- Database Site Models

Summary of the highlighted features of each tool are briefly mentioned in the following.

Features available in CODEBASE 6 [2]:

- Extremely fast response time
- Unlimited database size
- Low memory requirements
- Source code is available, to make changes
- Available in many international languages

- Additional tools like coding reporter, coding base, service administrator and Database fix can be incorporated
- Companies that use CODEBASE 6 include Lockheed, BOEING, AT&T, INTEL, IBM, and Microsoft

Features available in Attunity Connect [1]:

- Supports multi-threading
- Provides database management tools for integration, query processing and managing access
- Wizard to connect
- Concept of virtual database
- All user profiles and metadata are stored in Object stores
- Provides file pool caching

Features available in MARIPOSA [4]:

- Query processing is fragmented; the data itself is stored as fragments.
- Site manager to supervise the back end
- One or more backend servers
- A client front-end application

5. CONCLUSIONS

The recent trend in information processing on performing tasks to create, modify, exchange and such on given sets of data has made distributed databases very popular, because of the service they provide to maintain these resources across various platforms, regardless of the location they are residing in. There are many tools to help accomplish this, but from the user's point of view, it is very essential to know how to evaluate these tools in order to find the one best meets the given requirements. The authors give a set of requirements with different importance levels and that evaluate three different distributed database tools in order to show the process of choosing the right tool. It is extremely important to know what needs to be done as well as how to find such tool given multiple options. When studying the different systems some of the common flaws that we encountered were the ones due to portability. We found that most of the systems could be deployed in only one environment and if an application was developed in one environment it was hard to move it to another.

Another issue that needs to be addressed was the use of GUI and hiding the information irrelevant to the user. Most of the systems that we looked at were missing such GUI implementation. At last but not the least, most of the systems nowadays need to be web enabled and access to data from the Internet is of utmost importance. The systems should provide tools that automatically connect to the data sources, query them and generate results that are properly documented and can be displayed on the web. Based on the assessments made and rakings, we found Codebase 6 to be the best fit to the requirements proposed and compared to the other two systems that went under the evaluation.

REFERENCES

[1] Attunity Connect – User Manual, web page at URL: http://www.attunity.com

[2] CodeBase6 Documents available at URL: http://sequiter.com/products/cbase6.htm

[3] G. Mirkovia, "Benefits of Distributed Database Technology in Multimedia Applications", *LAIR – MIPRO 95 Proceedings of Multimedia and HyperMedia Systems*, 1995.

[4] Mariposa, UC Berkeley User document. Available at URL: http://epoch.cs.berkeley.edu:8000/mariposa/about.html

[5] Matthias Nicola and Matthias Jarke, "Performance Modeling of Distributed and Replicated Databases", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 4, July-August 2000.

[6] Oracle Document available at URL:http://oradoc.photo.net/ora816/server.816/a76960/ds_conce.htm#12207

[7] M.T. Ozsu and P. Valduriez, *Principles of Distributed Database Systems*, 2nd Ed., Prentice Hall, 1999.

[8] Project Summary. Available at URL: http://www.uga.edu/netinfo/gigapop/vbns/

[9] J.R. Nicol, C.T. Wilkes, and F. Manola, "Object Orientation in Heterogeneous Distributed Computing Systems", *IEEE Computer*, Vol. 26, No. 6, pp. 57-67, June 1993.

[10] M.T. Ozsu, U. Dayal and P. Valduriez (Eds): *Distributed Object Management Papers from the International Workshop on Distributed Object Management (IWDOM)*, Edmonton, Alberta, Canada, August 19-21, 1992, Morgan Kauffman, 1994.

[11] A.P. Sheth and J.A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", *ACM Computing Surveys*, Vol. 22, No. 3, pp. 183-236, 1990.

[12] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom, "Integrating and Accessing Heterogeneous Information Sources in TSIMMIS", *NGITS-Next Generation Information Technologies and Systems*, Naharia, Israel, June 27-29, 1995.

[13] M.L. Barja, T. Bratvold, J. Myllymaki, and G. Sonnenberger, "Informia: A Mediator for Integrated Access to Heterogeneous Information Sources", *Conference on Information and Knowledge Management (CIKM)*, Bethesda, Maryland, pp. 234-241, 1998.

[14] A. Elmagarmid, M. Rusinkiewicz, A. Sheth (Editors): *Management of Heterogeneous and Autonomous Database Systems Morgan*, Kaufmann Publishers, 1998.

FEATURE	CODEBASE 6	ATTUNITY	MARIPOSA
Developer	Sequiter Software	COMPAQ	UC Berkley
Database Type	Heterogeneous	Homogeneous	Homogeneous
File Supported	FoxPro, dBase, Clipper, Excel, Access	XML, Flat files, DB2, Informix, IMS, etc	PostGies
Database Size	Up to 1 Million Terabytes, can be extended	Not known, supports variable length records	Not known
Memory	Memory efficient, typically client/server requires around 1 MB	32 MB RAM and 24 MB hard disk space	8 MB RAM and 50 MB hard disk space
Platforms Supported	Windows 2000, 95, 98, NT, all versions of UNIX, LINUX, Solaris and Mac	IBM AS/400, UNIX, Open VMS, Windows 95, 98, 2000	DEC OSF/1 3.2 running on Digital Equipment Alpha
Scalability	Can be stand-alone or client/server, supports up to 10 simultaneous connections	Is Scalable	Can scale to a couple of sites on a WAN
GUI	Yes	Yes	Command Line
Querying Speed	Very fast access up to 1 million records in 0.65 secs	Not as fast as CODEBASE 6	Not known
Languages Supported	C, C++, VB, Delphi, Java, VBScript, ASP	Java based, support for multi-threading	C, C++
Logging	All changes get logged into the log file	Not known	No logging capability
Backup	Incremental backups	Not known	Manual backups
Stability	Highly stable since only server implements changes	Not known	Unstable and no mechanism to prevent data corruption in case of system crash
Web Accessible	Uses JDBC, ODBC drivers to construct from the web	Uses JDBC, ODBC drivers to construct from the web	Not accessible from web
Security	Secured via file access privileges and user accounts	Saves user profiles as object stores on the server	Very basic security features
Transaction Processing	Complete rollback and commit capability	Distribution transaction manager implements 2-phase commit protocol	Not known
Reliability	Reliable recovery from crashes and logging	Not known	Very insecure since the system manager can corrupt the data if it fails in the middle of a transaction
Query Processing	Not known	Query processor and optimizer are available	Query processing involves generating a query plan and integrating fragmented data
Installation	Simple, good documentation provided	Complex even with the presence of wizards	Very complex, manual setup
Help Files	Good help provided	Good help provided	Adequate release notes and user manual provided
Additional Tools	Management tools, report generation tools, debug tools and code controllers	Not known	Not supported
Cost	Approximately between $500 to $3000 (including add on tools)	Not known	Free, research software

Table 1. Evaluation of Distributed Database Tools