Development of a Message-Oriented Middleware for a Heterogeneous Distributed Database Systems

Lamia H. Khalid* and Manal F. Younis**
* Computer Science Department, College of Science, University of Baghdad.
** Computer Department, College of Engineering, University of Baghdad.

Abstract

Middlewares are enabling technologies for the development, deployment, execution and interaction of applications. These software layers are standing between the operating systems and applications. They have evolved from simple beginnings hiding network details from applications into sophisticated systems that handle much important functionality for distributed applications providing support for distribution, heterogeneity and mobility.

This paper concerns with the development of a Message_Oreinted Middleware (MOM) for a distributed database system. Middleware is a distributed software layer, or 'platform' which abstracts over the complexity and heterogeneity of the underlying distributed environment with its multitude of network technologies, machine architectures, operating systems and programming languages. The role of this middleware is to ease the task of designing, programming and managing distributed database applications by providing a simple, consistent and integrated distributed programming environment. It provides integrity and security to these databases.

Keyword**:** Distributed Database Systems (DDS)**,** Middleware, Client-Server, Message Oreinted Middleware (MOM).

Introduction

Distributed Database Systems have recently become an important area of information processing. Distributed Databases eliminate many of the shortcomings of centralized databases and fit more naturally in the decentralized structure of many organizations.

The main advantages of distributed systems that it can provide a simple, fault-tolerant and scalable architecture as well as high performance, but there are multiple points of failure in a distributed system, system components need to communicate with each other through a network, which complicates communication and opens the door for security attacks, middleware has been devised in order to conceal these difficulties from application engineers as much as possible; and it is increasingly used in this capacity [1].

As shown in Fig.(1), middleware is a software layer on top of an operating system that makes developing applications easier, by providing tools, libraries, and services. Recently, many researchers have been specially studying about middleware architecture of client-server system. The idea of using middleware to build a distributed system is comparable to using a distributed database management system when building an information system [2].

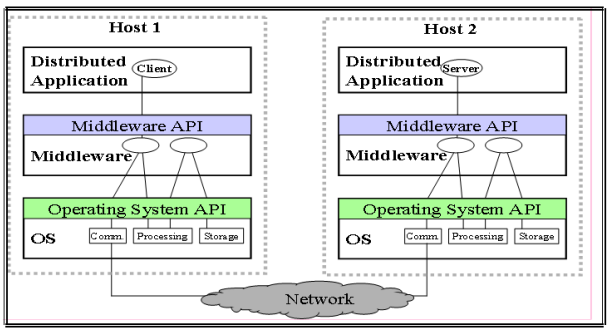

Fig. (1) Middleware in Distributed System Construction.

Several models have been used for Distributed Database and Middleware. The following are the most related works:

- Song X., Zhang R., [3].

This article describes middleware by adopting a horizontal and a vertical layer views. Middleware are enabling technologies for application development and execution in ubiquitous environments. In the horizontal view, it finds most types of middleware developed so far, such as Message Oriented Middleware (MOM), Object Request Broker

(ORB) middleware, databases middleware and more recently service-oriented architectures (SOA). Two new concepts emerged in this category: the "middleware of sensors" and the "middleware of middlewares".

- Choi J., Baek K., [4].

This paper presents a distributed event driven middleware architecture for situational awareness and intelligent decision making for command and control of geographically distributed networked battlefield agents. They tackle the important and challenging issues of distributed agents scheduling, synchronization, load balancing, and terrain database distribution/management/allocation in a distributed virtual battlefield environment.

- Abood A.H., [5].

This thesis tries to design a Distributed Database Management System that provides fast retrieval to any database tuple without taking in account the size of that database.

- Murtadhaa S.K., [6].

The objective of this research is to design and implement a distributed database with new layer of middleware which provides Security, Data integrity, Concurrency, Transparency, Error handling and recovery.

- Carvalho R. L., [7].

This paper proposes an approach to improve the level of quality of experience (QoE) that distributed database systems provide. Quality of experience is a measure of user's satisfaction when using a certain service or application. Therefore, the main objective of this paper is to provide mechanisms to increase user's satisfaction when accessing distributed database systems.

The remainder of this paper is organized as follows. Section 2 explains Communications Middleware. Section 3 reviews the test and results. Finally, Section 4 provides conclusions.

Communications Middleware:

The main component in distributed computing is the communication middleware that is used to connect the system's heterogeneous components and to manage the interactions between these components.

Middleware is connectivity software that consists of a set of enabling services that allow multiple processes running on one or more machines to interact across a network. Middleware is essential to migrate mainframe applications to client/server applications and to provide for communication across heterogeneous platforms. Middleware services are sets of distributed software that exist between the application and the operating system and network services on a system node in the network see Fig.(2) [9].

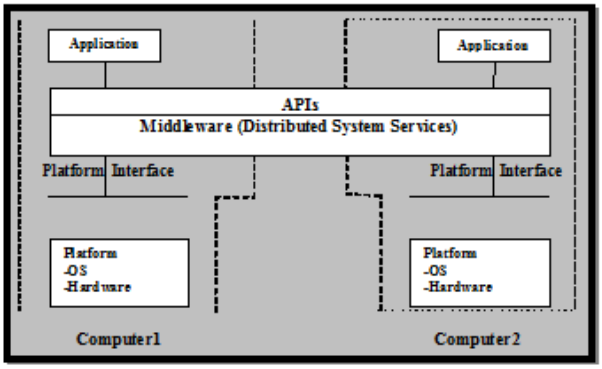

Fig. (2) Middleware for distributed systems.

Middleware services provide a more functional set of application program interfaces (API) than the operating system and network services to allow an application to [9]:

- locate transparently across the network, providing interaction with another application or service.
- be independent from network services.
- be reliable and available.
- scale up in capacity without losing function.

Database Middleware:

Database middleware is a software layer on top of an operating system that makes developing database applications easier, by providing tools, libraries, and services [4].

There are three main components in the database middleware as shown in Fig.(3)[8]:

- Application programming interface (API)
- Database translator
- Network translator

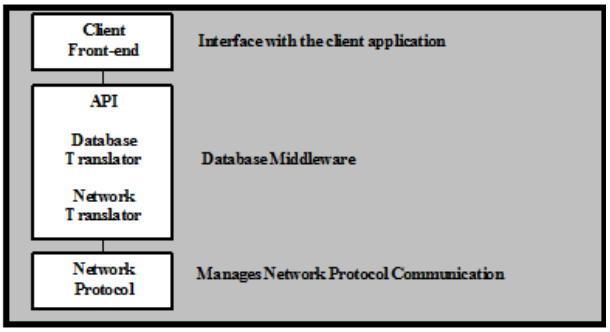

Fig. (3) Database Middleware Components.

Database middleware software can be classified according to the way clients and servers communicate across the network. There are four different kinds of middleware that have been developed. These vary in terms of the programming abstractions they provide and the kinds of heterogeneity they provide beyond network and hardware:

- **Distributed Tuples (DT):** A distributed relational database offers the abstraction of distributed tuples. Its SQL allows programmers to manipulate sets of these tuples (a database). Distributed relational databases also offer the abstraction of a transaction [10].
- **Remote Procedure Call (RPC):** Remote procedure call middleware extends the procedure call interface familiar to virtually all programmers to offer the abstraction of being able to invoke a procedure whose body is across a network [10].
- **Distributed Object Middleware (DOM):** Distributed object middleware provides the abstraction of an object that is remote yet whose methods can be invoked just like those of an object in the same address space as the caller. Distributed objects make all the software engineering benefits of object-oriented techniques. Encapsulation, inheritance, and polymorphism are available to the distributed application developer [10].
- **Message-Oriented Middleware (MOM):** Message Oriented Middleware is a popular asynchronous message exchange mechanism in heterogeneous distributed applications. It provides the applications in a distributed environment to send and receive messages, but still being loosely coupled. The Message based integration provides automation and simplifies the time consuming integration tasks like create, deploy and manage integration solutions. Asynchronous Messaging is a backbone for many of the event driven architectures due to the obvious advantages of asynchronous systems where the message client need not maintain the connection and session with the message receiver; no confirmation is required from the receiving application [10]. MOM provides the abstraction of a message queue that can be accessed across a network. It is very flexible in how it can be configured with the topology of programs that deposit and withdraw messages from a given queue. Many MOM products offer queues with persistence, replication, or real-time performance [9].

The Methodology of the Proposed Middleware

The proposed middleware provides simple authentication mechanism based on the user name and password to ensure that only authorized users can access the database. This middleware applied on two computers as a three-tier client/server model. It is designed to increase the interoperability, portability, and flexibility of an application by allowing the application to be distributed over multiple heterogeneous platforms. It reduces the complexity of developing applications that span multiple operating systems and network protocols by insulating the application developer from the details of the various operating system and network interfaces-APIs.

The designed middleware is a software that resides in both portions of client/server architecture. This proposed middleware increases the flexibility of architecture by enabling applications to exchange messages with other programs without having to know what platform or processor the other application resides on within the network. The proposed system provides a message queue between two interoperating middlewares, so if the destination middleware is busy or not connected, the message is held in a temporary storage location until it can be processed.

As shown in Fig.(4), a connection must be established between the client and the server where each of them has a middleware where each computer can be client or server (i.e peer – to – peer); the two middlewares use two queues, one on the client side and the other on the server side. These queues contain all requests issued by two computers. One queue includes the requests of the client (computer1 or computer2) and the other queue includes the requests of the server (computer1 or computer2). The middleware at the client sends the request as sql statement to middleware at the server. The middleware at the server serves the request and sends reply to middleware at the client. The queues are managed as first-come first-serve including the data using HTML.

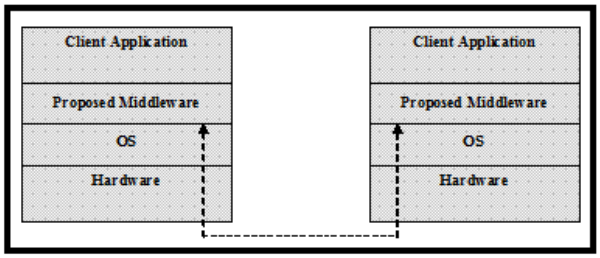

Fig. (4) Structure of Proposed Middleware.

The suggested middleware is a Message-oriented middleware (MOM) which uses messaging to communicate asynchronously between the client and the service provider.

It is designed to be suitable in a heterogonous Relational Distributed Database Management System (RDDBMS) environment, where two different DBMS are used (Oracle database and SQL Server 2008 database) running on two different platforms. The first platform is Pentium 4 running under windows seven, while, the second platform is Pentium 4 running under Windows vista. The proposed distributed system is a three-tier client/server architecture for database application. It includes: application programming interface (API), database translator and network translator. Fig.(5) shows the proposed middleware structure.

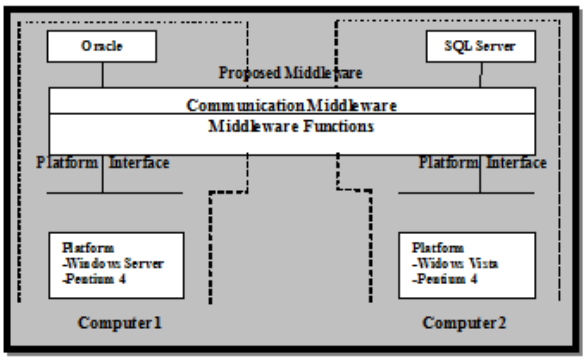

Fig. (5) Middleware Accessing Multiple Database Servers.

SQL is used in the proposed middleware as generic query language to access different and multiple database servers. Functions are defined, using SQL queries, to allow the users to manipulate two different databases (Oracle and SQL Server databases). The functions are Insert, Delete, Update and Select.

The designed middleware also ensures the Integrity of the databases. Database integrity refers to the correctness and consistency of stored data. It can be considered as another type of database protection. While it is related to security, it has wider implications; integrity is concerned with the quality of data itself. Integrity is usually expressed in terms of constraints, which are the rules the database is not permitted to violate. Therefore, Integrity is protecting database against authorized users.

The proposed middleware connects these two different databases using Open Database Connectivity (ODBC) which provides a standards interface to data regardless of the database platform in which it resides. Access to data through ODBC requires appropriate driver for a particular data source.

The ODBC architecture consists of four components: API, driver manager, driver, and data source. Fig.(6) illustrates the ODBC architecture and how its components interact.

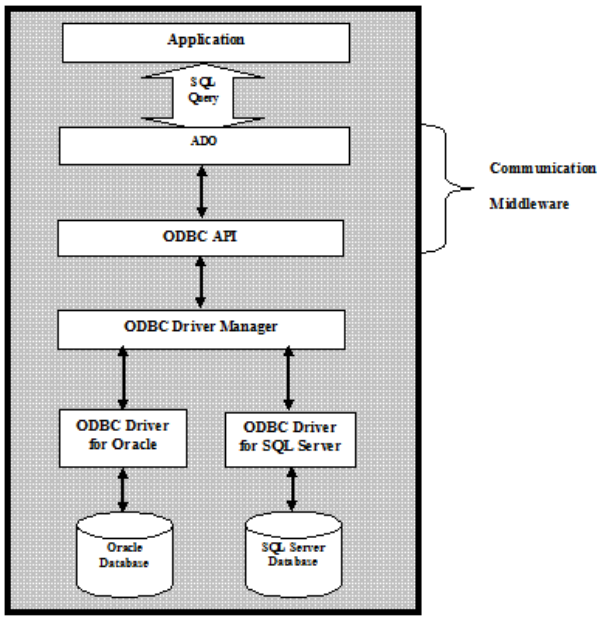

Fig. (6) ODBC architecture.

The suggested middleware was tested on two RDBMDs. They are : (i) Library database built using SQL Server database and , (ii) Student database built using Oracle database. The Library database includes two tables (Publishers and Books) see Table (1) and (2). The two tables are bind by a field contains a unique value.

Table (1) Publishers.

Name	*Address*	*Email*	*ID*
Ahmed Ali	Baghdad	Ahmed@yahoo.com	1
Muna Ahmed	Mousal	Muna@yahoo.com	2

Primary key

Table (2) Books.

BID	*Book Name*	*Book Year*	*ID*
004-23	Operating system	2002	1
004-55	Data structure	2003	1
004-66	Image processing	2002	2

Primary Key Foreign key

The two tables (Publishers and Books) tables are related by the ID field. This field is primary key in publishers table and foreign key in books table.

The Oracle database includes two tables these are (Student and Course) tables see Table (3) and (4).

Table (3) Student.

Student Name	Address	Birthdates	Phone_num	Department	Class	Year	SID
Ahmed Ali	Kirkuk	1978/3/4	5566778	Computer	Second	2004	1

Primary key

Table (4) Course.

Crs_Code	Prof_name	Class-Subject	SID
1001	Dr. Hasan Ali	Data structure	1
1002	Dr. Ahmed kamal	System analysis	1

Primarykey **Foreign key**

These two tables are related by SID field. This field is primary key in student table and foreign key in Course table.

To illustrate this work the following windows will be listed. The main interface is shown in Fig.(7).

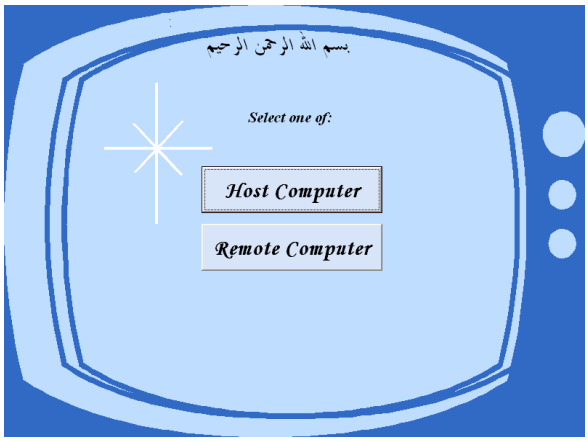

Fig. (7) Main Interface.

In this interface the user can select the Host Computer wether he wants to work with the local database, or selects Remote Computer or he wants to work with remote database. If the user wants to use the remote database, the user enters the Database Name, User Name and Password as shown in Fig.(8). If the user wants to use the local database, he enters the Database Name, User Name and Password database then the menu is shown in Fig.(9).

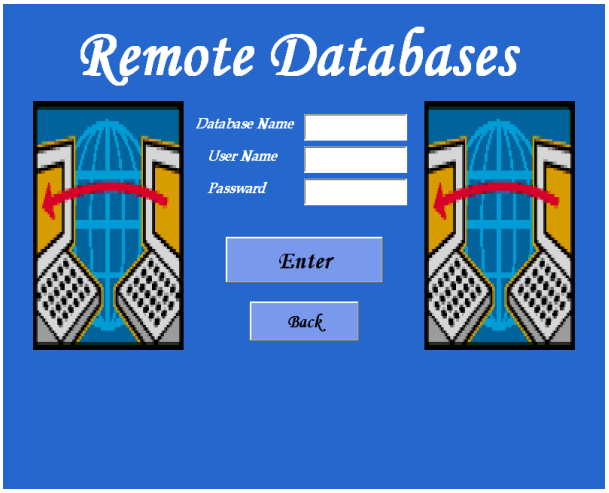

Fig. (8) Main Interface of Remote Databases.

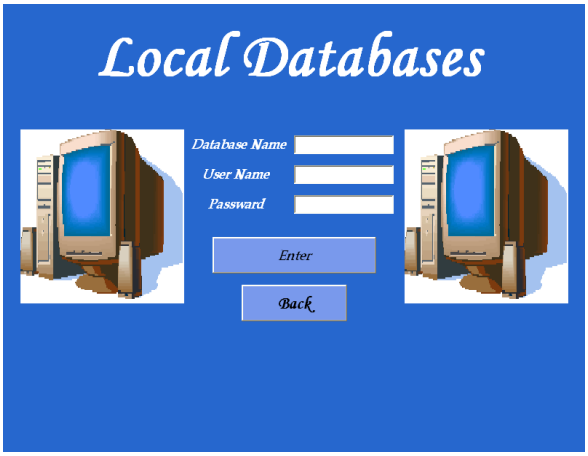

Fig. (9) Local Databases Interface.

If the information is correct press enter and the menu in Fig.(10) is presented, if not the massage will the displayed in the dialog box "This database does not exist".

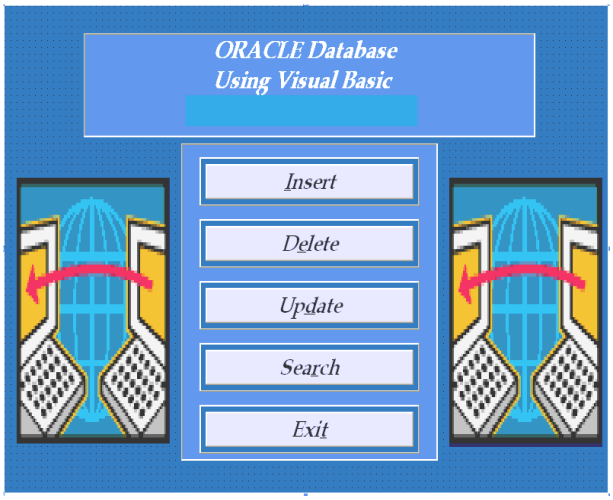

Fig. (10) The main Interface of the Oracle Database application.

As an example if the user selects the Search option then the window in Fig.(11) will be displayed.

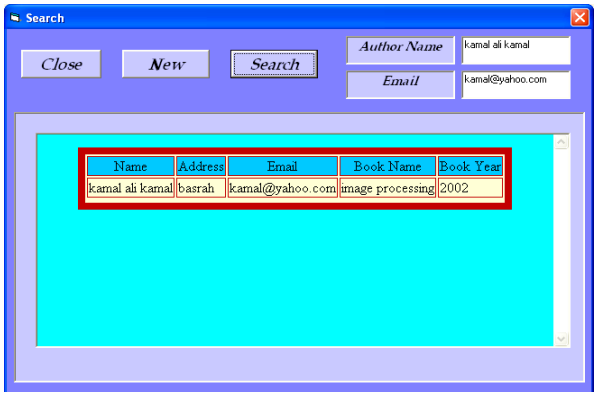

Fig. (11) Interface of the Search from Remote database by Author Name.

The user can send a request by printing the Author name and Email. The middleware on the remote computer replys to the request using html web browser. The reply is a table contains all information of the Author, but if the record is not exist it replys the message 'This record does not exist'.

The two middlewares can send requests to each other at the same time, and each can receive the request and send the required record if it is exist.

Conclusions

The conclusions that can be drawn from this work are listed as follows:

1. The designed middleware manages the complexity and heterogeneity inherent in distributed systems.
2. The designed middleware provides transparency using a common programming language through distributed database systems such as Oracle database and SQL Server database.
3. The designed middleware provides connectivity between database servers using SQL statements.
4. The designed middleware helps to ease the task of designing and managing distributed database systems.

References

[1] Ibrahim N., "Orthogonal Classification of Middleware Technologies", Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, Grenoble Informatics Laboratory, Grenoble, France, pp. 45-46, 2009.

[2] Megherbi D. B., Pandit D., "An Event-Driven Multi-Agent Middleware Architecture and Protocol Design for Intelligent Geographically Distributed Battlefield Training, Modeling and Simulation", CMINDS Research Center, ECE Department, University of Massachusetts Lowell, USA, pp. 21-23, 2011.

[3] Song X., Zhang R., "Research on Constructing Distributed Large Database based on J2EE", IEEE 3rd International Conference on Communication Software and Networks, pp.50, 2011.

[4] Choi J., Baek K., "A Middleware Architecture for Composing Robot Services on Distributed System Environments", School of Computer Science and Engineering, Soongsil University, 5th International Conference on New Trends in Information Science and Service Science, V.1, pp.50-51, 2011.

[5] Abood A.H., "Design and Implementation of Fast Retrieval Distributed Database Management System", College of Science/ Saddam University, pp.20-23, 2002.

[6] Murtadhaa S.K., "Design and Implementation of a Distributed Database

Applications, Vol. 3, No. 1, pp. 148-148, 2012.

الخلاصة

تمكن الوسائط التكنولوجيات الحديثة للتطوير والتنفيذ والنشر والتفاعل بين التطبيقات. توضع هذه الطبقات البرمجية بين أنظمة التشغيل والتطبيقات. حيث تطورت البرمجيات من بدايات بسيطة تخفي تفاصيل الشبكة من التطبيقات في نظم متطورة التي تتعامل مع الكثير من وظائف هامة للتطبيقات الموزعة التي تقدم الدعم لعدم التجانس والتوزيع والتنقل.

يهتم هذا البحث في وضع الوسيطة من نوع رسالة الموجه لنظام قاعدة البيانات الموزعة. والوسيطة هي طبقة البرمجيات الموزعة، التي تجرد أكثر تعقيد وعدم تجانس توزيع البيئة الأساسية مع العديد من تقنيات الشبكات، هندسة الآلات، وأنظمة التشغيل ولغات البرمجة. دور هذه الوسيطة هو تسهيل مهمة تصميم وبرمجة وإدارة تطبيقات قواعد البيانات الموزعة من خلال توفير بيئة البرمجة الموزعة ذات خاصية بسيطة ومتسقة ومتكاملة. أنها توفر السلامة والأمن لقواعد البيانات.

Middleware", College of Science/ Baghdad University, pp. 16-17, 2004.

[7] Carvalho R. L., "Quality of experience in distributed databases", Journal: Distributed and Parallel Databases ISSN: 09268782, **V**.29 **No**:5 pp. 361-396 2011**.**

[8] BATES J., "Software Engineering and Middleware", Wolfgang Emmerich, Dept. of Computer Science /University College London, pp.117-120, 2000.

[9] PETER R., and CARLOS C., "Database Systems Design, Implementation, and Management", an International Thomson Publishing Company ITP, 3rd Edition, U.S.A, pp. 475-477, 2007.

[10] BLAHA M. and PREMERLANI W., "Object-Oriented Modeling and Design for Database Applications", Prentice-Hall, Inc. Simon and Schuster, A Viacom Company Upper Saddle River, New Jersey 07458, U.S.A., pp.34-36, 1998.

[11] Ravi K. G ., Babu A.V., "Self-regulating Message Throughput in Enterprise Messaging Servers – A Feedback Control Solution", Bangalore, India, Hyderabad, India, (IJACSA) International Journal of Advanced Computer Science and