

University Of Sunderland

School Of Computing and Information Systems

Evaluation Study In

Migrating Database System To Web-Based System



By

Isa Abdulla Salman Alhawi

A Project Dissertation Submitted in partial fulfilment of

the regulations governing the award of the degree of

Master of Science

In Computer Based Information Systems

Project Supervisor: Dr. Mike Wyvill

UK, Sunderland
October, 2001



University of Sunderland

Isa Abdulla Salman Alhawi

has been awarded the degree of

Master of Science

having followed an approved programme in

Computer Based Information Systems

15 February 2002



Peter Fidler

Vice-Chancellor and
Chief Executive

Shant P. P.

Academic Registrar

Abstract

The project discusses the process of migrating a traditional database to the Web and the issues involved in this process. The project presented different viewpoints about this process and adopted the viewpoint, which stated that there were no difficulties in the process of transferring a traditional system to the Web. The project goes on to prove that viewpoint by conducting the necessary background research and building a prototype to illustrate the findings.

In addition, the project also shows that there are operational differences between a traditional and a web-based database. These operational differences enable web-based database to provide certain benefits to businesses. Such benefits include: ease of database access and manipulation, better control over the content and security of the database, instant database access from anywhere on the globe, lower costs, the ability to modify existing options and add new ones, and the ability to work with different platforms. The prototype illustrates all these benefits in more details.

The project introduces two main hypotheses. First, there are no difficulties in migrating a traditional database to the Web. This hypothesis has been proven correct by the project. In fact, migrating a database to the Web has a number of benefits such as the ones mentioned above. Second, there are no operational differences between a traditional and a web-based database, which the project shows to be incorrect.

Acknowledgements

I would like to acknowledge the full support of my parents during my studies in the MSc. Project, especially their financial support. The project could not have been completed without the assistance of my project supervisor Dr. Mike Wyvill and local Tutor Dr. Mansoor Al-Aali. I indeed owe them a debt of gratitude for their valuable consultation and assistance. I would also like to express my thanks and respect to the readers of my project.

Finally, I would like to extend my thanks to all people who devoted some of their valuable time and effort to help me in completing this project. Special thanks to Mr. Ahmed Ali Matter, Mr. Jaffar Isa and Mr. S. Sadiq Alawi who extended all available assistance to help produce this project in its present form.

**Evaluation Study In:
Migrating Database System to Web-Based System**

Table of Contents

Abstract.....	i
Acknowledgements	ii
Table of Contents.....	iii
 Chapter One: Introduction	
1.1 Introduction	1
1.2 The Project problem.....	1
1.3 Project Hypotheses.....	2
1.4 Project Objectives	3
1.5 Data Gathering Techniques	3
1.6 Chapters Summaries.....	3
 Chapter Two: Traditional and Distributed Database Systems	
2.1 Introduction	5
2.2 What is a Distributed Database Systems?	9
2.3 Architectural methodology used in Multi-database Systems	13
2.4 Distributed Database Advantages	25
2.5 Distributed Database Disadvantages.....	29
2.6 Issues in migrating Distributed Data to Web.....	36
2.7 Summary.....	38
 Chapter Three: Web-based Database Systems	
3.1 Introduction	39
3.2 What is a Web-based Database?.....	44
3.3 Migrating Database to the Web	46
3.4 Building A Web-based Database.....	49
3.5 Advantages of managing Web-based Database Systems.....	53
3.6 Difficulties of managing Web-based Database Systems	56
3.7 Security Concerns	59
3.8 Summary	60
 Chapter Four: Development and Implementation of an on-line Payroll Prototype	
4.1 Introduction.....	61
4.2 Characteristics of the proposed System	61
4.3 Development Methodology	61
4.4 System Specifications	62
4.5 Analysis and Design.....	64
4.6 Prototype Building	67
4.7 Testing the hypothesis.....	70

Chapter Five: Evaluation and Findings	
5.1 Prototype Evaluation	71
5.2 Project Evaluation and Findings.....	72
Chapter Six: Conclusion and Recommendations	
6.1 Conclusion	74
6.2 Recommendations	74
Appendices	
A. Project Terms Of Reference	75
B. Deliverables List	76
C. Schedule.....	77
D. Gannt Chart.....	78
References	79

Chapter One

Introduction

- 1.1 Introduction
- 1.2 The Project problem
- 1.3 Project Hypotheses
- 1.4 Project Objectives
- 1.5 Data Gathering Techniques
- 1.6 Chapters Summaries

Chapter One

Introduction

1.1 Introduction

Database Professionals choose Web servers based on a unique approach, which enables fast and reliable data transfer via the World Wide Web. With this approach, Web servers can utilize a wide range of tools and mechanisms in the process of database integration. In the past, a Webmaster, who was involved in database integration projects, needed third-party solutions to have access to an RDBMS such as Oracle or Informix. Nowadays, Web server technologies allow direct links between the Web server and the database.

This chapter outlines the project problem, hypotheses, objectives and data gathering techniques. The chapter also provides a general outline of the content of each subsequent chapter.

1.2 The Project problem

All companies search for a method with which they can access important and relevant data easily and cost efficiently. Most companies manage their database structures by using programming languages and platforms such as C++, VB, Oracle, Informix or Java. These companies, however, face a number of difficulties in developing and maintaining these structures, and in exchanging data between different servers. These difficulties arise because the old methods and models used to manage a database structure are costly and difficult to build and handle easily.

Due to the need for new methods to manage database structures, tools and technologies have been developed to replace the old methods with new ones. This development in turn led to the wide use of web applications in the field of database management. This project is concerned with evaluating the new methods, which are used today to change a traditional database into a Web-Based database such as SQL.

1.3 Project Hypotheses

The project hypotheses are developed to challenge the difficulties mentioned in the project problem section, namely; the difficulties associated with maintaining a database and exchanging data between servers. The aim is to test the validity of these hypotheses and examine their roles in the process of transferring data and managing database structures over the Web. The hypotheses of this project are as follows:

Hypothesis No 1: There is no difficulty associated with transferring traditional database platforms into Web-based Database platforms.

Hypothesis No 2: There are no operational differences between a Web-based system and a traditional database system.

In order to test the hypotheses of this project, a study case will be presented to examine the changes involved in the process of transferring a traditional database structure into a Web-based structure. This study case will present a company that is in the process of changing its Payroll System from its traditional structure to a Web-based structure. The payroll system is chosen as an example because payroll systems can easily illustrate the challenges involved in that

change. In addition, the size of the company is small enough to build an illustrative prototype without any difficulties.

1.4 Project Objectives

As mentioned earlier, the project focuses on evaluating the methods, which can be used to change a traditional database system into a Web-based system and allow companies to access their Database via the Internet with ease and cost efficiency from any point on the globe.

The objectives of this project are:

- To present a theoretical study of database management on the Web and Distributed environment.
- To conduct a brief survey of the methods and / or techniques used to change a database platform into a Web-based platform.
- To develop and implement a Web-based Payroll System Prototype.
- To evaluate the outcomes of the Project.

1.5 Data Gathering Techniques

The project will rely on books and research papers that deal with database management systems to obtain the necessary data for the purpose of testing the hypotheses of this project. In addition, other information will be obtained via interviews with organizations that deal with database management systems.

1.6 Chapters Summaries

The project will be divided into the following chapters:

Chapter 1: Introduction:

This chapter discusses the Management of Database Systems, the Web-based Systems, the project problem, the project objectives, the project hypothesis, and the data gathering techniques.

Chapter 2: Traditional and Distributed Database Systems:

This chapter provides the definition of Distributed Database Systems, and introduces the Architectural methodology used in Multi-database Systems. It also discusses the strengths and weaknesses of Web-based Distributed Database Systems. Finally, it presents the issues involved in transferring Distributed Database Platforms to the Internet.

Chapter 3: Web-based Database Systems:

This chapter discusses the definition of Web-based Database Systems, and illustrates the changes involved in transferring a traditional database structure to a Web-based structure. It also presents the advantages and disadvantages of the new methods of managing Web-based Database Systems.

Chapter 4: Development and Implementation of a Web-based Payroll Prototype:

The chapter discusses the characteristics of the proposal system and development methodology. It also covers the System Specifications, Analysis and Design, the development of a Web-based Payroll System Prototype, and an analysis of the hypotheses.

Chapter 5: Evaluation and findings:

This chapter focuses on evaluating the findings of the project and presents a general evaluation of the quality of these findings.

Chapter 6: Conclusion and Recommendations:

This chapter outlines the conclusions of the project and presents a general recommendation report.

Chapter Two

Traditional and Distributed Database Systems

- 2.1 Introduction
- 2.2 What is a Distributed Database Systems?
- 2.3 Architectural methodology used in Multi-database Systems
- 2.4 Distributed Database Advantages
- 2.5 Distributed Database Disadvantages
- 2.6 Issues in migrating Distributed Data to Web
- 2.7 Summary

Chapter Two

Traditional and Distributed Database Systems

2.1 Introduction

Today, the success of an organization depends on its ability to acquire accurate data operations in time. To manage such data effectively, one is expected to use, analyse and guide its activities use analyse and guide its activities.

With the vast amount of information available to today's researchers and managers, there is a great need for a mechanism with which the information can be organized in an understandable and useful format. Date and Darwen explain how this need has risen in recent years and the role that database management systems have played to fulfil this need: The amount of information available is literally exploding, and the value of data as an organizational asset is continuously recognized. Yet without the ability to manage this vast store of data, and to quickly find the information that is relevant to a given question. As the amount of information increases, it tends to become a distraction and a liability, rather than an asset. This paradox drives the need for increasingly powerful and flexible data management systems. To get the most out of their large and complex datasets, users must have tools that simplify the tasks of managing the data and extracting useful information meanwhile. Otherwise, data can become a liability, with the cost of acquiring it and managing it far exceeding the value that is derived from. Database management systems first emerged on the marketplace at the end of the 1960s and are now sufficiently mature that a powerful database management

system (DBMS) underlies virtually all information systems today. The aim of the DBMS is to provide powerful tools for managing a database, which is defined as an integrated collection of shared data, a reservoir of data. (Date and Darwen, 1998).

Distributed database technology is one of these tools, and probably the most powerful of them. It involves the merging of two divergent concepts, namely integration through the database element and distribution through the networking element, see Figure 2.1 (David B. and Jane G., 1992).

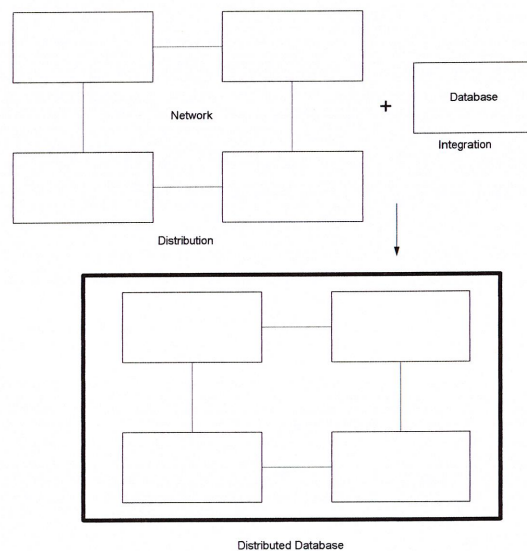


Figure 2.1. The distributed Database concept

As explained by Killen, distributed database (DDBS) technology is the union of what appears to be two diametrically opposed approaches to data processing. The first one is the so-called database system and the second is computer network technologies. Database systems have taken from a paradigm of data processing in which each application defined and maintained its own data, see Figure 2.2. (Killen M, 1992), to one in which the data is defined and administered centrally,

see Figure 2.3 (David B. and Jane G., 1992). With this orientation, data independence is achieved and the application programs will not be affected by changes in the logical structure of data, and vice versa. The need to integrate the operational data of an enterprise and to better control access to such data is the major reason for the development and use of database systems.

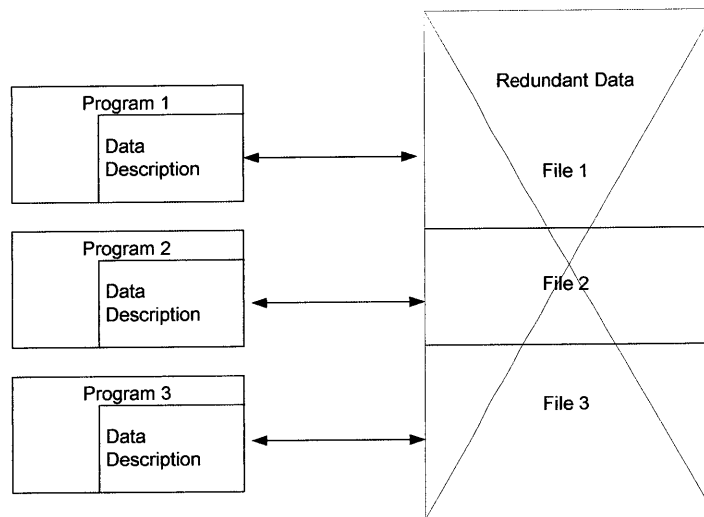


Figure 2.2. Traditional File Processing

The technology of computer networks, on the other hand, promotes a mode of work that goes against all centralization efforts. At first glance it might be difficult to understand how these two contrasting approaches can possibly be synthesized to produce a technology that is more powerful and more promising than either one alone. The key to this understanding is the realization that the important objective of the database technology is integration, not centralization.

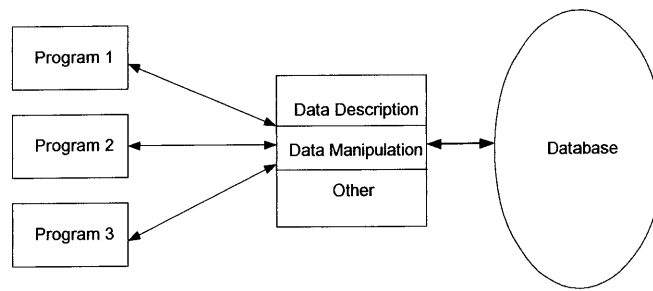


Figure 2.3. Database Processing

That is, it is possible to achieve integration without centralization, and that is exactly what the distributed database technology attempts to do. (Shaku A, 1992)

The integration of database technology is achieved through the so-called distributed processing or computing systems. According to Tamer and Patric, distributed computing systems are defined as a number of autonomous processing elements (Not necessarily homogeneous) that are interconnected by a computer network and that cooperate in performing their assigned tasks. Furthermore, the authors classify distributed computing systems according to a number of criteria such as:

- Degree of coupling: refers to a measure that determines how closely the processing elements are connected together. This can be measured as the ratio of the amount of data exchanged to the amount of local processing performed in executing a task. If the communication is done over a computer network, there exists weak coupling among the processing elements. However, if components are shared, that is strong coupling.
- Interconnection structure: is a point-to-point interconnection between processing elements, as opposed to those that use a common interconnection

channel. The processing elements might depend on each other to a great extent in the execution of a task, or this interdependence might be as minimal as passing messages at the beginning of execution and reporting results at the end.

- Synchronization between processing elements: might be maintained by a synchronous means. (M. Tamer and Patric V., 1999)

The distributed processing is used because most current applications of computer technology are inherently distributed. Such applications include: Electronic commerce over the Internet, multimedia applications such as news-on-demand or medical imaging, and manufacturing control systems.

Distributed database technology should be viewed within the context of distributed processing. That is, new innovations in distributed database technology has provided and continue to provide tools that make distributed processing easier and more efficient. In addition, there is no doubt that the development of general-purpose, adaptable, and efficient distributed database systems will help greatly in the task of developing distributed software.

2.2 What is a Distributed Database System?

A distributed database is defined as a collection of multiple, logically interrelated databases distributed over a computer network. A distributed database management system (DDBMS) is defined as the software system that permits the management of the DDBS and makes the distribution transparent to the users. The two important phrases in these definitions are “Logically Interrelated” and

“Distributed over a computer network”. They help eliminate certain cases that have sometimes been accepted to represent a DDBS (David B. and Jane G., 1992).

A DDBS is not a collection of files that can be individually stored at each node of a computer network. To form a DDBS, files should not only be logically related, but there should be structure among the files, and access should be via a common interface. There has been much recent activity in providing DBMS functionality over semi-structured data that are stored in files on the Internet. In light of this activity, the above requirement may seem unnecessarily strict. However, providing DBMS-like access to data is different from a DDBS (M. Tamer and Patric V., 1999).

It has sometimes been assumed that the physical distribution of data is not the most significant issue. The proponents of this view would therefore feel comfortable in labelling as a distributed database two (related) databases that reside in the same computer system. However, the physical distribution of data is very important. It creates problems that are not encountered when the databases reside in the same computer. The physical distribution does not necessarily imply that the computer systems are geographically far apart; they could actually be in the same room (Distributed database, <http://www.compapp.dcu.ie/databases>). It simply implies that the interconnection between them is done via a network instead of through shared memory, with the network as the only shared resource.

In addition, the above definition of DDBS rules out multiprocessor systems as distributed systems. A multiprocessor system is generally considered to be a system where two or more processors share some form of memory, either primary memory, in which case the multiprocessor is called shared memory, or called

tightly coupled, or secondary memory in which case the multiprocessor is called shared disk, or loosely coupled, see Figure 2.4 (Sten F. A., Joakim E. and Mats L., 2000).

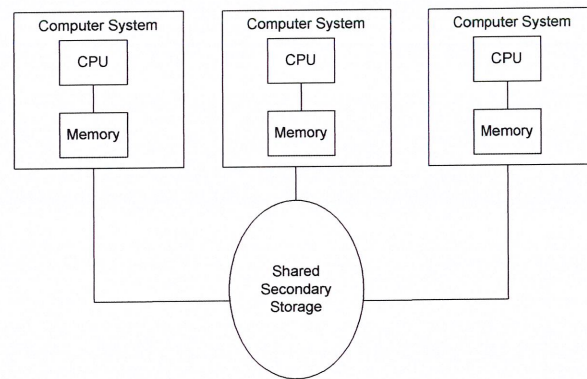


Figure 2.4. Shared Disk Multiprocessor

Sharing memory enables the processors to communicate without exchanging messages. The shared-nothing architecture is one where each processor has its own primary and secondary memories as well as peripherals, and communicates with other processors over a very high speed interconnect such as bus or a switch. In this sense the shared-nothing multiprocessors are quite similar to the distributed environment (Manjula K. and Johnny W., 2000).

However, there are differences between the tight interaction of multiprocessor architectures and the rather loose interaction that is common in distributed computing environments.

The major difference lies in the mode of operation. The multiprocessor design is rather symmetrical, consisting of a number of identical processors and memory components, and controlled by one or more copies of the same operating system, which strictly controls the task assigned to each processor. This is not true in

distributed computing systems, where heterogeneity of the operating system as well as the hardware is quite common (Manjula K. and Johnny W., 2000).

In addition, a DDBS is not a system where, despite the existence of a network, the database resides at only one node of the network. The database is centrally managed by one computer system and all the requests are routed to that site. See figure 2.5 (M. Tamer and Patric V., 1999).

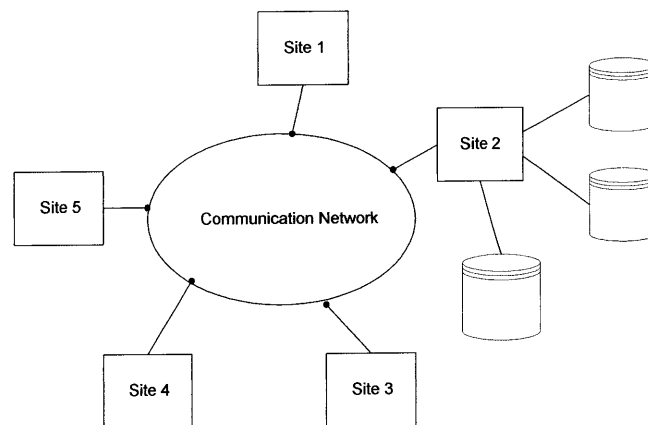


Figure 2.5. Central Databases on a Network

The only additional consideration has to do with transmission delays. It is obvious that the existence of a computer network or a collection of files is not sufficient to form a distributed database system. What researchers are interested in is an environment where data is distributed among a number of sites. See figure 2.6 (M. Tamer and Patric V., 1999).

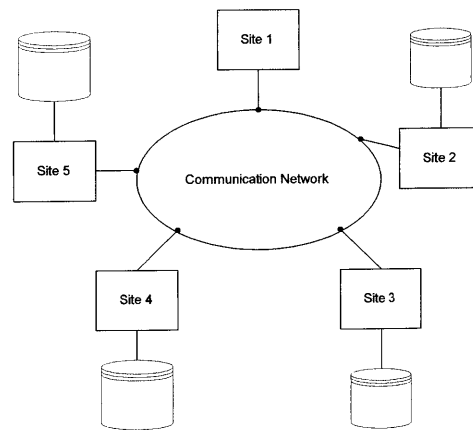


Figure 2.6. DDBS among a number of sites

2.3 Architectural methodology used in Multidatabase Systems

Object orientation is expected to play a significant role in addressing some of the modelling and architectural issues in database interoperability. The modelling issues are relevant if the interoperability framework includes object DBMSs as well as relational ones. In this case, the global conceptual schema has to be object-oriented, since that is the most general model (Maria A. and other, 1998).

According to Teresa, object modelling has a number of characteristics that are useful for managing database interoperability. Two of these characteristics are particularly important in addressing interoperability concerns. The first is encapsulation, which allows the differences in interfaces and implementations of component DBMSs to be hidden. However, when object-orientation is brought to bear, “translator” does not accurately capture the role of this piece of software. A more appropriate term that has been proposed is wrapper, since the component DBMS is “wrapped” and its internal operations are hidden from other system

components. The second characteristic of object modelling that is useful for managing database interoperability is that object models can be specialized or generalized. Specialized objects are modelled by sub typing, while generalized objects are modelled by super typing. This allows for the creation of types that abstract the similarities of entities stored in different databases (Teresa H., 1995). In addition to these modelling advantages, there are a number of distributed object computing platforms whose main purpose is to facilitate the development of open systems by allowing applications to easily communicate with each other. Two such platforms are Object Management Architecture (OMA) from Object Management Group (OMG) and Component Object Model (COM), Distributed Component Object Model (DCOM) and Object Linking and Embedding (OLE) environment from Microsoft. These platforms take the view that interoperability among information systems requires an environment that supports a common way to design each software piece whereby there are certain functions that can be expected of each component (Teresa H., 1995). The platforms provide an infrastructure that supports communication between components and between application programs and components, and incorporate services that are commonly needed by all distributed applications. The various platforms differ in how they accomplish these goals, and in the remainder:

2.3.1 Object Management Architecture (OMA)

Object Management Architecture (OMA) is an environment defined by the Object Management Group (OMG), which is a consortium of industry vendors committed to an object-oriented approach to building distributed systems. OMG puts forth standards and allows vendors to develop products

that meet those standards. OMA defines a common object model, a common model of interaction by means of object invocations, and a set of common object services and facilities. OMA modules consist of the application objects; the Common Object Request Broker Architecture (CORBA), which directs requests and responses between objects; a set of common object services (COSS), which are the basic functions required for object management, e.g. naming, transaction, and life cycle management; and a set of common facilities, which are generic object-oriented tools for various applications, see figure 2.7 (Dogac A. and other, 1998). Common facilities are divided into horizontal ones that are used by all applications, e.g. user interface, and class browser, and vertical ones that are developed specifically for vertical market segments, e.g. healthcare, finance, and telecommunications (Fred R., and other, 1994).

OMA's object model is a generic one that provides objects, values, operations, types, classes, and sub-type/super-type relationships among types. An object is an abstraction with a state and a set of operations. The operations have well-defined signatures, and the operations together with the signatures form the interface of each object. Each object and operation has a type (Teresa H., 1995).

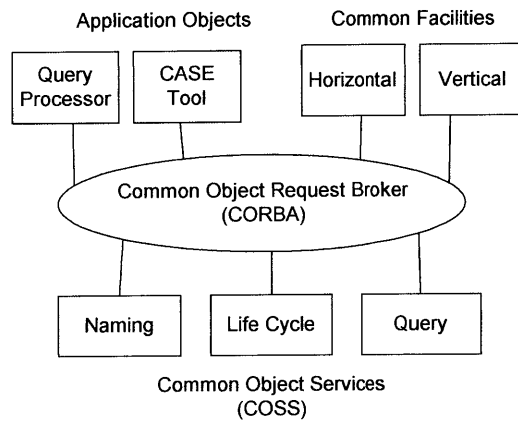


Figure 2.7. (CORBA) Architecture

Objects communicate with each other by sending requests, which are operation calls with one or more parameters. Any of these parameters can identify an object. The arguments and results are then passed by value.

2.3.2 Common Object Request Broker Architecture (CORBA)

CORBA is a distributed object framework proposed by OMG. The core of the CORBA architecture is the Object Request Broker (ORB) that acts as the object bus over which objects transparently interact with other objects located locally or remotely. A CORBA object is represented to the outside world by an interface with a set of methods. A particular instance of an object is identified by an object reference. The client of a CORBA object acquires its object reference and uses it as a handle to make method calls, as if the object is located in the client's address space (S. Vinoski, 1997). The ORB is responsible for all the mechanisms required to find the object's implementation, prepare it to receive the request, communicate the request to it, and carry the reply (if any) back to the client. The object

implementation interacts with the ORB through either an Object Adapter (OA) or through the ORB interface.

CORBA is the key communication mechanism of OMA, in which objects communicate with each other via an Object Request Broker (ORB) that provides brokering services between clients and servers, see figure 2.8 (S. Vinoski, 1997). Brokering involves target object location, message delivery, and method binding. Clients send a request to the ORB asking for certain services to be performed by whichever can fulfil those needs. ORB finds the server, passes it the message from the client, and receives the result, which is passed to the client (Siegel, 1996).

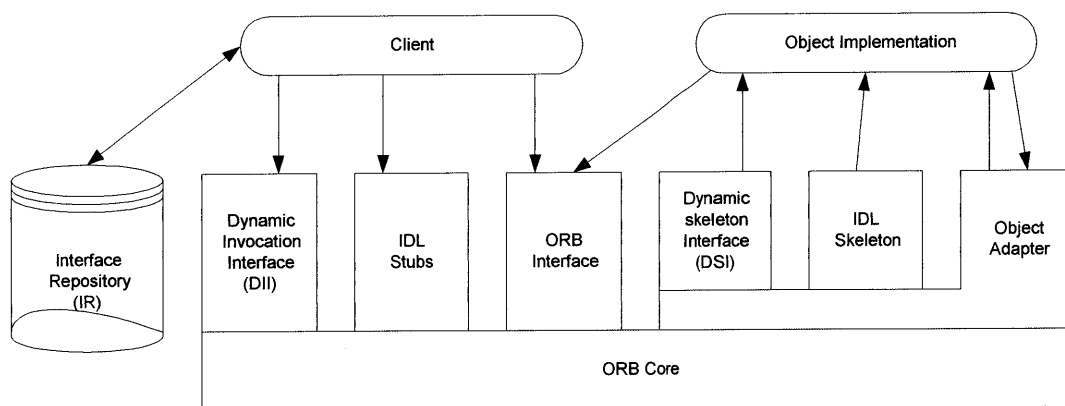


Figure 2.8. (CORBA) Architecture

The ORB core accomplishes the basic communication between two CORBA objects. OMG does not place any restriction on how ORBs are implemented. In most ORB implementations, existing Inter-Process Communication (IPC) methods, such as Unix socket libraries, shared memory and multithreaded libraries, are used to achieve actual communication among clients, servers and the ORB. Yet ORB can be as simple as a library that supports

communication among objects and their clients that are co-resident in the same process space (Dogac A. and other, 1998).

The interfaces are defined by means of the Interface Definition Language (IDL). It is a declarative, language-independent host. It forces interfaces to be defined separately from object implementations. Objects can be constructed using different programming languages and still communicate with one another. IDL enables a particular object implementation to introduce itself to potential clients by “advertising” its interface. The IDL interface specifications are compiled as declarations in the programmer’s own language. Language mappings determine how IDL features are mapped to facilities of a given programming language. There are standardized language mappings for C, C++, Smalltalk, Ada and Java. IDL language mappings are those in which the abstractions and concepts specified in CORBA meet the “real world” of implementation (Siegel, 1996).

2.3.3 Component Object Model (COM)/Object Linking & Embedding (OLE)

COM’s architecture is quite different from CORBA’s. Blakeley sites the main differences between the two architectures in the following:

- A COM/OLE object is one that supports one or more interfaces as defined by its Class. Thus, there could be multiple interfaces to an object. All objects support one interface called IUnknown.
- COM objects have no identifiers.

- There is no inheritance defined among COM object classes. The relationship among them is defined by means of containment/delegation and aggregation.
- COM objects do not have states; applications obtain a pointer to interfaces that point to the methods that implement them.
- There are two definition languages: Interface Definition Language (IDL) for defining interface, and Object Definition Language (ODL) for describing object types.

All COM objects have the following characteristics:

1. A Globally Unique Identifier (GUID) identifies them. It is not an object identifier in the classical sense.
2. They must register with the Window's registry. This registration is then used by COM library functions to locate, start and stop the components.
3. They must publish and implement a set of interfaces. COM also defines a set of interfaces that must be supported by all COM objects, in addition to the particular interfaces they support in the context of the application in which they intend to participate.

A COM server can create object instances of multiple object classes. A COM object can support multiple interfaces, each representing a different view or behaviour of the object. An interface consists of a set of functionally related methods. A COM client interacts with a COM object by acquiring a pointer to one of the object's interfaces and invoking methods through that pointer, as if the object resides in the client's address space (Kazerouni L.

and Karlapalem K., 1997). COM specifies that any interface must follow a standard memory layout, which is the same as the C++ virtual function table. Since the specification is at the binary level, it allows integration of binary components possibly written in different programming languages such as C++, Java and Visual Basic. COM objects exist in two forms: DLLs and EXEs. Regardless of the form it is in, a COM object must meet all the standards prescribed by the COM component architecture. DLLs must reside on the same platform as the client; they are often referred to as “local servers” (Kazerouni L. and Karlapalem K., 1997). EXE objects can be accessed via networks; they are often referred to as the “remote servers”.

Clients access COM objects by means of the interface defined for each object, see figure 2.9 (Siegel, 1996). This is accomplished by indirection through an Interface Function Table, each of whose entries points to an interface implementation inside the COM object. There is one Interface Function Table per each interface that the object supports. The client obtains a pointer to the Interface Function Table that corresponds to the particular interface that it wishes to access, and invokes the interface functions contained therein. This method isolates clients from interface implementations.

The IUnknown interface has three methods: The first is the QueryInterface method, by which the client can interrogate the COM objects as to whether it supports a particular interface, the second is the AddRef method and the last is the Release method. The last two methods are for reference counting, which is used for garbage collection.

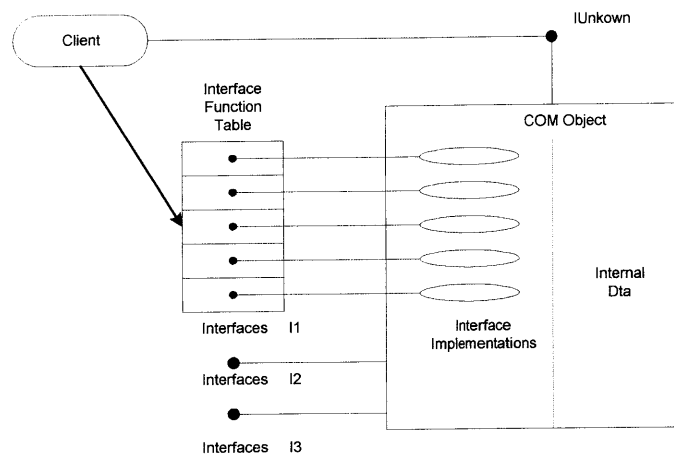


Figure 2.9. COM Objects and Interfaces

OLE database provides database interoperability in the COM/OLE environment. It also extends the OLE environment to data repositories. It defines a uniform interface for all data repositories in the form of “rowsets”. In other words, all OLE database providers expose data as rowsets, which are tabular representations of their data. COM objects that deal with repositories, therefore, communicate in terms of rowsets, see figure 2.10 (Blakeley, 1996).

A rowset object serves the function of a wrapper in providing a uniform interface. In its basic form, a rowset object has three interfaces in addition to IUnknown. The first interface is the Irowset, which provides methods for sequential iteration over the rows of a rowset. The second interface is the IcolumnsInfo, which provides information about the columns of the rowset. The third interface is called the Accessory, which permits the definition of column bindings to client program variables. More elaborate OLE database interfaces are defined for inserting, deleting, and updating rows in a rowset,

as well as for providing sophisticated ways of accessing a rowset, e.g. direct access, and scrolling (Blakeley, 1996).

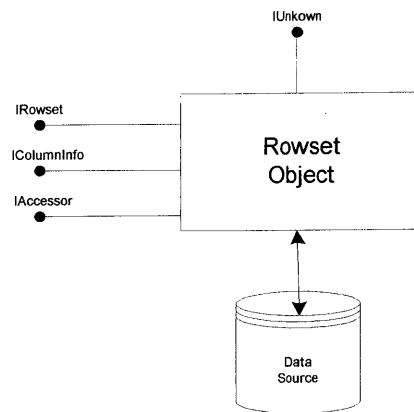


Figure 2.10. Rowset Object Abstraction

Using OLE database, many of the designs discussed within the context of CORBA and database interoperability are made automatically for the system designer. There are commercial OLE DB providers for many of the relational DBMSs, which can be used to put together an interoperability framework for relational DBMSs. Similar providers are supposed to be available for object DBMSs as well.

2.3.4 Distributed Component Object Model (DCOM)

DCOM is the distributed extension to COM. DCOM is an alternative to (CORBA). DCOM/OLE is similar in functionality to CORBA ORB, while OLE is the complete environment for componentisation. DCOM/OLE is a single vendor proposal and therefore, its contents are somewhat fluid and changing (Siegel, 1996).

Both DCOM and CORBA frameworks provide client-server type of communications. To request a service, a client invokes a method implemented by a remote object, which acts as the server in the client-server model. The service provided by the server is encapsulated as an object and the interface of an object is described in an Interface Definition Language (IDL). The interfaces defined in an IDL file serve as a contract between a server and its clients. Clients interact with a server by invoking methods described in the IDL. The actual object implementation is hidden from the client. Some object-oriented programming features are present at the IDL level, such as data encapsulation, polymorphism and single inheritance. CORBA also supports multiple inheritances at the IDL level, but DCOM does not. Instead, the notion of an object having multiple interfaces is used to achieve a similar purpose in DCOM. CORBA IDL can also specify exceptions (Kazerouni L. and Karlapalem K., 1997).

In both DCOM and CORBA, the interactions between a client process and an object server are implemented as object-oriented RPC-style communications. To invoke a remote function, the client makes a call to the client stub. The stub packs the call parameters into a request message, and invokes a wire protocol to ship the message to the server. At the server side, the wire protocol delivers the message to the server stub, which then unpacks the request message and calls the actual function on the object, see figure 2.11 (Joe S., 2nd edition).

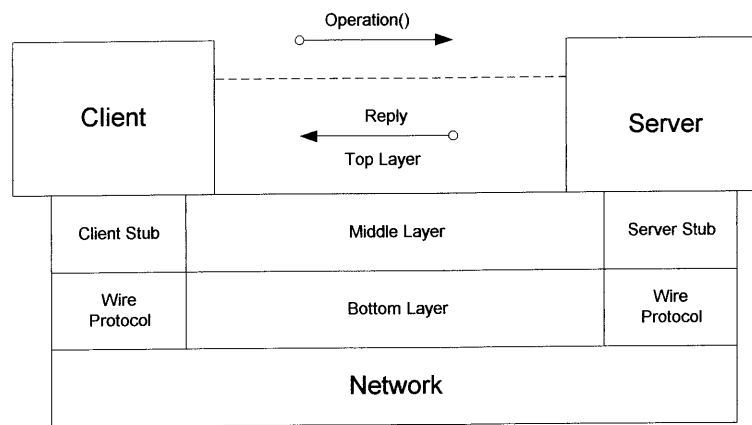


Figure 2.11. RPC structure

In DCOM, the client stub is referred to as the proxy and the server stub is referred to as the stub. In contrast, the client stub in CORBA is called the stub and the server stub is called the skeleton. Sometimes, the term "proxy" is also used to refer to a running instance of the stub in CORBA (Siegel, 1996).

DCOM's root is Microsoft's Component Object Model. The transition from COM to DCOM is conceptually simple. DCOM is "COM with a longer wire". In fact, the move from COM to DCOM was straightforward, because most COM library functions supported distributed and networked components. Further, Windows provides the code needed to locate and communicate with components over a network. This facility allows clients to locate components across the network either transparently or by requesting a component residing on a particular machine (D. Box, July 1997).

A remote component requires no change to its implementation or the clients that access it; the communication is handled via Windows' registry and

networking facilities. These facilities provide the elegant “longer wire” feel of COM/DCOM’s way of building distributed software (D. Box, July 1997).

2.4 Distributed Database Advantages

Many advantages of DDBSs have been cited in literature, ranging from sociological reasons for decentralization to better economics:

2.4.1 Transparency of Distributed and Replicated Data:

Transparency refers to the separation of the higher-level semantics of a system from lower-level implementation issues. In other words, transparency allows the system to hide the implementation details from users. The advantage of a fully transparent DBMS is the high level of support that it provides for the development of complex applications such as E-Commerce applications.

A distributed DBMS should provide a number of features that make the distributed nature of the DBMS transparent to the user. The distributed database should look like a centralized system to the users (Alexandr Z. and Natalya R., 2000). These features include the following:

- **Location transparency**

Location transparency, in a distributed DBMS, means that users/programs should not need to know where data is stored. This is a system catalogue’s concern only. This simplifies application program logic and allows data movement within the distributed database as usage patterns emerge.

- **Replication transparency**

Replication transparency means that partitioned/replicated updates must be propagated through to all copies in existence. Replication is desirable and transparent.

- **Performance transparency**

According to Peddemors and Hertzberger, the benefits of performance transparency are:

1. A distributed DBMS fragments the concepts database, enabling data to be stored in close proximity to its points of use. Fragmenting the concepts database has two potential advantages. First, since each site handles only a portion of the database, contention for CPU and I/O services is not as severe as for centralized database. Second, localization reduces remote access delays that are usually involved in wide area networks.

Most distributed DBMSs are structured to gain maximum benefit from data localization. Full benefits of reduced communication overhead can be obtained only by a proper fragmentation and distribution of the database.

2. The inherent parallelism of distributed systems may be exploited for inter-query and intra-query parallelism. Inter-query parallelism results from the ability to execute multiple queries at the same time while intra-query parallelism is achieved by breaking up a single query into a

number of sub queries each of which is executed at a different site (A. J. H. Peddemors and L. O. Hertzberger, 1999).

- **Transaction/Concurrent transparency**

A transaction is a basic unit of consistent and reliable computing, consisting of a sequence of database operations executed as an atomic action. A transaction transforms a consistent database state to another consistent state even when these transactions are executed concurrently, and even when failures occur (Castro M. and other, 1997). Therefore, a DBMS that provides full transaction-transparency support guarantees that concurrent execution of user transactions will not violate database consistency in the case of system failures as long as each transaction is correct. Distributed transactions are executed at a number of sites at which they access the local database. With full support for distributed transactions, user applications can access a single logical image of the database and rely on the distributed DBMS to ensure that their requests will be executed correctly no matter what happens in the system, which means that user applications do not need to be concerned with coordinating their accesses to individual local databases nor do they need to worry about the possibility of site or communication link failures during the execution of their transaction (Alexandr Z. and Natalya R., 2000).

- **Catalogue transparency**

The entries of distributed database catalogue must specify the site(s) at which data is being stored in addition to original data in a system

catalogue in a centralized DBMS. Because of data partitioning and replication, this extra information is needed (Castro M. and other, 1997).

2.4.2 Capacity and Growth

An important advantage of distributed databases is that as the organization grows, new sites can be added with little or no upheaval to the DBMS. Compare this to the situation in a centralized system, where growth entails upgrading with changes in hardware and software that effect the entire database.

2.4.3 Distributed Database Sharing

An important advantage of distributed databases is that it allows users to obtain data stored in a number of sites without having to be physically located in any one of these sites. In addition, control over the obtained data is not lost.

2.4.4 Reliability and Availability

Another advantage of distributed databases is that even when a portion of a system (i.e. a local site) is down, the overall system remains available. With replicated data, the failure of one site still allows access to the replicated copy of the data from another site. The remaining sites continue to function. The greater accessibility enhances the reliability of the system (Castro M. and other, 1997).

2.4.5 Efficiency and Flexibility

An advantage of distributed databases is that data is stored close to the anticipated point of use. Hence if usage patterns change then data can be dynamically moved

or replicated to where it is most needed (Beg, M.M. Sufyan; Ravikumar, C.P., 2000).

2.5 Distributed Database Disadvantages

The disadvantages of the distributed approach to DBMS implementation are its cost and complexity. A distributed system, which hides its distributed nature from the end user, is more complex than the centralized system. Increased complexity means that the acquisition and maintenance costs of the system are higher than those for a centralized DBMS. The parallel nature of the system means that errors are harder to avoid and those in the applications are difficult to pinpoint (Alexandr Z. and Natalya R., 2000). In addition, the distributed system, by its very nature, entails a large communication overhead cost in coordinating messages between the different sites.

2.5.1 Distributed Database Design

According to Castro, there are two basic alternatives to placing data in a DDBS:

2.5.1.1 Data replication

Replicated designs can be either fully replicated, also called fully duplicated, or partially replicated. In a fully replicated design, the entire database is stored at each site. In a distributed DBMS, a relational table or a partition may be replicated or copied, and copies

may be distributed throughout the database. This feature can cause problems for propagating updates and concurrency control.

2.5.1.2 Data partitioning

Partially replicated or partially duplicated designs allow users to store each partition of the database at more than one site, instead of at all sites. In a distributed DBMS, a relational table may be broken up into two or more non-overlapping partitions or slices. A table may be broken up horizontally, vertically, or a combination of both. Partitions may in turn be replicated. This feature causes problems for concurrency control and catalogue management in distributed databases (Castro M. and other, 1997).

The two fundamental design issues are fragmentation, the separation of the database into partitions called fragments, and distribution, the optimum distribution of fragments. Fragmentation and distribution involve mathematical programming in order to minimize the combined costs of storing the database, processing transactions against it, and communication (Beg, M.M. Sufyan; Ravikumar, C.P., 2000).

2.5.2 Concurrency control

Concurrency control can be done in several ways. Locking and timestamping are two techniques that can be used, but timestamping is generally preferred.

The problems of concurrency control in a distributed DBMS are more severe than those in a centralized DBMS because of the fact that data may be replicated and partitioned (Thomasian, 1998). If a user wants a unique access to a piece of data, to perform an update for instance, the DBMS must be able to guarantee the unique access to that data, which is difficult if there are copies throughout the sites in the distributed database. The concurrency control techniques most commonly used are:

- **Locking**

Locking is a method of concurrency control, which involves placing one of a set of types of locks on a piece of data in the database. Thus, locking prevents other concurrent users of the database from reading or writing that data, depending on the type of lock placed. While conceptually simple, this method can lead to problems of deadlock that do not arise with other methods of concurrency control like timestamping. The database management system contains a "lock manager" whose function is to manage and control locking of data by concurrent users (Thomasian, 1998).

- **Timestamping**

Timestamping is a method of concurrency control with which all transactions are given a timestamp or unique date/time/site combination and the database management system uses one of a number of protocols to schedule transactions, which require access to the same piece of data.

While more complex to implement than locking, timestamping does avoid deadlock occurring by avoiding it in the first place (Thomasian, 1998).

Most DBMS's are multiple-user systems; that is, they are systems that allow any number of users to access the same database at the same time. In such a system some kind of concurrency control mechanism is needed to ensure that concurrent transactions do not interfere with each other's operation. There are essentially three ways in which things can go wrong. That is, there are three ways in which a transaction, though correct in itself, can nevertheless produce the wrong answer because of interference on the part of some other transaction. The interfering transaction may also be correct in itself. It is the interleaving of operations from the two correct transactions that produces the overall incorrect result. The three problems are:

- The lost update problem.
- The uncommitted dependency problem.
- The inconsistent analysis problem.

2.5.3 Distributed update propagation

Update propagation in a distributed database is problematic because of the fact that there may be more than one copy of a piece of data due to replication, and data may be split up because of partitioning. Any updates to data performed by any user must be propagated to all copies throughout the database. The use of snapshots is one technique for implementing this. Snapshots, like a view, are a derived relation (however, snapshot are physically stored in the database), and are used for implementing updated

propagation in distributed databases (Fessant F. and other, 1998). Snapshots allow freezing without interfering with transactions that are suitable for applications that tolerate earlier versions of data.

2.5.4 Distributed Query Processing

Query processing deals with designing algorithms that analyse queries and convert them into a series of data manipulation operations. The challenge of using query processing is how to decide on a strategy for executing each query over the network in the most cost-effective way. The factors to be considered are the distribution of data, communication costs, and the lack of sufficient locally available information (Fessant F. and other, 1998). The objective is to optimise where the inherent parallelism is used to improve the performance of executing the transaction.

2.5.5 Distributed Directory Management

A directory contains information about data items in the database.

Disadvantages related to directory management have to do with the so-called database placement problem, which causes updates to be lost due to improper handling of linked data. That is, when data is updated at one part of the database, other related data is not updated at the same time. A directory may be global to the entire DDBS or local to each site; it can be centralized at one site or distributed over several sites; there can be a single copy or multiple copies (M. Tamer and Patric V., 1999).

2.5.6 Distributed Deadlock Management

The disadvantage of deadlock is similar in nature to that encountered in operating systems where data is lost due to freezing problems. The competition among users for a set of resources can result in a deadlock if the synchronization mechanism is based on locking. The well-known alternatives of prevention, avoidance and detection / recovery also apply to DDBSs (M. Tamer and Patric V., 1999).

These disadvantages are not isolated from one another because: 1- it is difficult to study them and almost impossible to present them all together. 2- it might be possible to characterize the effect of one disadvantage on another one through the use of parameters and constraints. It is obvious that each disadvantage is affected by the solutions found for the others, and in turn affects the set of feasible solutions for them, see figure 2.12 (Bouganim L. and other, 1999).

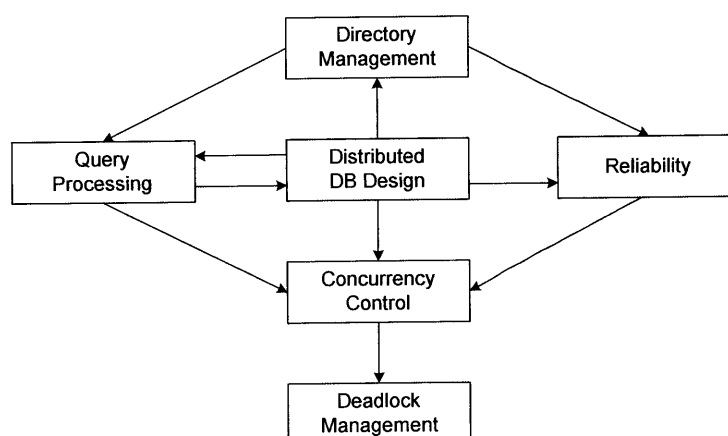


Figure 2.12. Relationship Among Research Issues

The relationship among the components is shown in the previous figure. The design of distributed databases affects many areas. It affects directory management, because the definition of fragments and their placement determine

the contents of the directory as well as the strategies that may be employed to manage them. To determine the query evaluation strategy, the query processor uses the same information. Also the access and usage patterns that are determined by the query processor are used as inputs to the data distribution and fragmentation algorithms. Similarly, directory placement and contents influence the processing queries (Bouganim L. and other, 1999).

The replication of fragments when they are distributed affects the concurrency control strategies that might be employed. Some concurrency control algorithms cannot be easily used with replicated database. Similarly, usage and access patterns to the database will influence the concurrency control algorithms. If the environment is updated intensively, the necessary precautions are quite different from those in a query-only environment. There is a strong relationship among the concurrency control problem, the deadlock management problem, and reliability issues. This is to be expected, since together they are usually called the transaction management disadvantage. The concurrency control algorithm that is employed will determine whether or not a separate deadlock management facility is required. If a locking-based algorithm is used, deadlocks will occur, whereas they will not if timestamping is the chosen alternative (Thomasian A., 1998).

Reliability mechanisms are implemented on top of a concurrency control algorithm. Therefore, the relationship among them is self-explanatory. Also reliability mechanisms have an effect on the choice of the concurrency control algorithm. Techniques to provide reliability make use of data placement information since the existence of duplicate copies of the data serve as a safeguard to maintain reliable operation (M. Tamer and Patric V., 1999).

2.6 Issues in Migrating Distributed Data to the Web

With the ongoing advance in Web technology, everyone can migrate Data to the Web independently at any time. The flexibility and autonomy of accessing and sharing data on the Web is impressive. On the other hand, one has to learn to deal with the increase of volume and diversity of online Database and the constant changes of Data sources in number, content, and location. The diversity of applications that are deployed on the Web requires different data delivery modes and protocols in different frequencies (Liu L. and other, 1998).

With the Web, the client is highly portable and is sometimes called “universal client”. This yields a new form of client/server architecture called “three-tier client/server”. With the traditional client/server architectures, the server runs the database programs, and the client runs the application programs and the graphical interface, see figure 2.13. The problems with this architecture are: 1- the number of application programs increases. 2- the clients get fatter and require more storage and computing power. 3- propagating changes to application programs on many heterogeneous clients may be difficult (M. Tamer and Patric V., 1999).

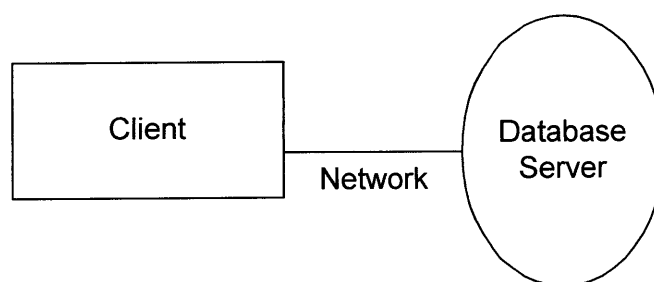


Figure 2.13. Traditional Client/Server architecture

With the “three-tier client/server” architecture, an application server is introduced to run application programs, see figure 2.14. So in this case, after migrating the database, the client is a Web browser dedicated to the graphical interfaces. Application logic that may be needed at the client, e.g., to check data entry, can be supported through applets, which are small application programs downloaded from the application server (Nguyen T. and Srinivasan V., 1996).

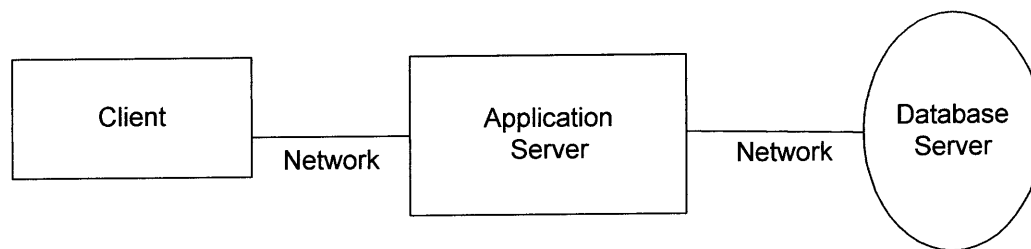


Figure 2.14. Three-Tier Client/Server architecture

The Web browser communicates with the Web server-using HTTP, see figure 2.15. The database gateway is a CGI program that performs the mapping between HTML inputs and query strings, and between result tuples and HTML report forms (M. Tamer and Patric V., 1999).

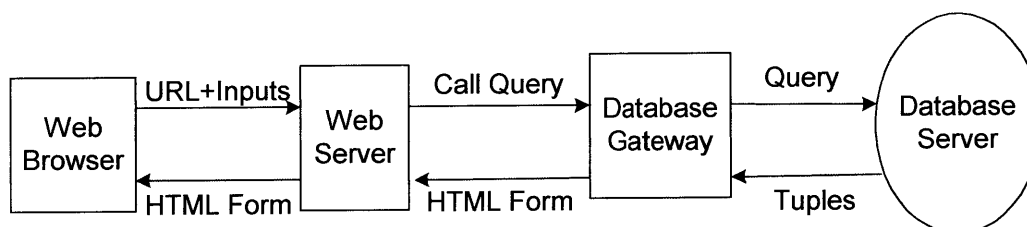


Figure 2.15. Accessing Database from a Web Browser

Nguyen and Srinivasan propose a general-purpose cross language variable substitution to achieve this in the DB2 WWW Connection system. The Web server using the URL and associated inputs calls the database gateway, and sends the corresponding query to the database server. After the query has been executed,

it transforms the result tuples into dynamic query forms and reports, and the database language (SQL) for querying (Nguyen T. and Srinivasan V., 1996).

The previous system allows a database and its applications to be distributed through the Web. Data can be stored where it is used and/or applications can be located where the activities they support take place.

2.7 Summary

This chapter provided the definition of Distributed Database Systems, and introduced the Architectural methodology used in Multi-database Systems. It also discussed the strengths and weaknesses of Web-based Distributed Database Systems. Finally, it presented the issues involved in transferring Distributed Database Platforms to the Internet.

Chapter Three

Web-based Database

Systems

-
- 3.1 Introduction
 - 3.2 What is a Web-based Database?
 - 3.3 Migrating Database to the Web
 - 3.4 Building A Web-based Database
 - 3.5 Advantages of managing Web-based Database Systems
 - 3.6 Difficulties of managing Web-based Database Systems
 - 3.7 Security Concerns
 - 3.8 Summary

Chapter Three

Web-based Database Systems

3.1 Introduction

Once upon a time, in the primitive and barbarian days before computers, the amount of information shepherded by a group of people could be collected in the wisdom and the stories of its older members. In this world, storytellers, magicians, and grandparents were considered a great and honoured storehouse for all that was known. Dealing with vast storehouses of wheat, rice, and maize became quite a chore for the monarchs and emperors that developed along with the new economy. There was simply too much data to be managed in the minds of the elders, who by now were feeling the effects of hardware obsolescence as they were being pushed quietly into the background.

And so, in order to store all the new information, humanity invented the technology of writing. And though great scholars like Aristotle warned that the invention of the alphabet would lead to the subtle but total demise of the creativity and sensibility of humanity, data began to be stored in voluminous data repositories, called books. As we know, eventually books propagated with great speed and soon, whole communities of books migrated to the first real "databases", libraries. Unlike previous versions of data warehouses (people and books), that might be considered the australopithecines of the database lineage, libraries crossed over into the modern-day species, though they were incredibly primitive of course (Franklin M. and Zdonik S., 1998).

Libraries helped users to develop codes by which data could be stored and easily obtained. Without these codes, the vast amount of information stored in these libraries, in the form of books, would be impossible to find and the library would turn out to be a place of chaos.

And for the next couple thousand years, Database grew, and grew along with associated storage/retrieval technologies such as the filing cabinet, collared tabs, and three ring binders. All this until one day about half a century ago, some really bright folks including Alan Turing, working for the British government were asked to invent an advanced tool for breaking German cryptographic "Enigma" codes. That day the world changed again. That day the computer was born. The computer was an intensely revolutionary technology of course, but as with any technology, people took it and applied it to old problems instead of using it to its revolutionary potential (Understanding Web Databases www.hubcom.com).

Almost instantly, the computer was applied to the age-old problem of information storage and retrieval. After all, by World War Two, information was already accumulating at rates beyond the space available in publicly supported Database Systems. Thus, the first attempts at information storage and retrieval followed traditional lines and metaphors. The first systems were based on discrete files in a virtual library. In this file-oriented system, a bunch of files would be stored on a computer and could be accessed by a computer operator. Files of archived data were called "tables" because they looked like tables used in traditional file keeping. Rows in the table were called "records" and columns were called "fields" (Barry E. and Mick R., 2000).

Essentially, in order to find a record, someone would have to read through the entire file and hope it was not the last record. It will be dilemma with a hundred thousands records. Attempts to achieve random access processing, which is the ability to efficiently access a single record without searching the entire file to find it, led to the creation of the indexed file-oriented system in which a single index file stored "key" words and pointers to records that were stored elsewhere. This made retrieval much more efficient. To find data, one needed to only search for keys rather than to read entire records. However, even with the benefits of indexing, the file-oriented system still suffered from problems including:

- Data Redundancy - the same data might be stored in different places
 - Poor Data Control - redundant data might be slightly different. For example, if Mr. Essa wanted to change his name to Mr. Isa, then the change would only be reflected in some of the files containing his data.
 - Inability to Easily Manipulate Data - it was a tedious and error prone activity to modify files by hand
 - Cryptic Work Flows - accessing the data could take excessive programming effort and was too difficult for real-users, as opposed to programmers
- (Franklin M. and Zdonik S., 1998).

A database is a computerized record-keeping system. This system contains a number of components, namely; the data, the hardware that stores that data, the software, which manipulates the data stored in the hardware, and the users who access the data and utilize it in form of information.

Databases, another creature of the 60s, were created to solve the problems with file-oriented systems in that they were compact, fast, easy to use, current,

accurate, allowed the easy sharing of data between multiple users, and were secure. A database might be as complex and demanding as an account tracking system used by a bank to manage the constantly changing accounts of thousands of bank customers, or it could be as simple as a collection of electronic business cards on laptop. The important thing is that a database allows storing data and getting it or modifying it when needed easily and efficiently regardless of the amount of data being manipulated (internet.com, 2001).

What the data is and how demanding it will be when retrieving and modifying that data is simply a matter of scale. Traditionally, databases ran on large, powerful mainframes for business applications such as Oracle 8 or Sybase SQL Server packages (SQLWebFAQ www.sqlWeb.com).

In addition, the widespread of personal computers has allowed people to utilize the tremendous computing capabilities of computers and to use PC-based database programs such as Microsoft Access. The widespread of PCs has also led to the development of design programs, and the establishment of service vendors on the Web.

Today's engineers can handle the large quantity of information they require in their professional work by storing that information on their PCs, which provide easy access to the information. In addition, one can apply some sorting as well as interpolation through the generated database in order to have a quick and usable plan without the hassle of analysing or computing problems specific to their individual design. To have good databases for a typical engineering design, it always requires individual efforts to create a set of usable data; equally important are the problems as to how to maintain and update them, as well as to share such

information with others. In the past, to have remote access to information in a mainframe and/or workstation seemed to be an easy task via a UNIX based operating system and the associated networks, see figure 2.1 (Barry E. and Mick R., 2000). It should be noted that only computers with identical software and hardware capabilities could be linked together, and the server controlled almost all networking functions.

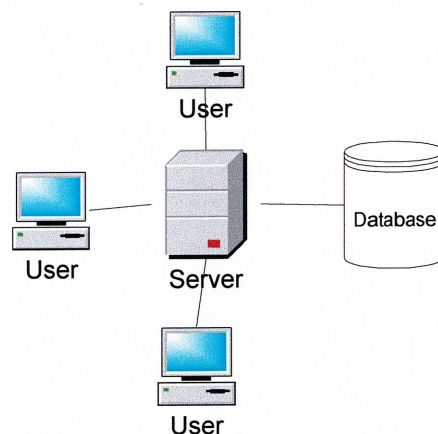


Figure 2.1. Traditional Database exchange

Today the World Wide Web (WWW) is the best example of distributed information and electronic publishing. It can connect different platforms easily and transfer different information, such as text, image, voice, or even computer animation. After accessing the network through a server and asking for a file, the computer executes the remaining processes. The power of the World Wide Web comes not simply from static HTML pages, which can be very attractive, and the important first step into the WWW but especially from the ability to support those pages with powerful software (Ash, Tim, 1998). With its friendly interface, easy to use controls and buttons, and graphics and animations, the Internet has allowed users to share, manipulate and obtain information that suits their individual needs.

3.2 What is a Web-based Database?

Web Data is an Internet Database Portal, specializing in finding, categorizing and organizing online databases. Web Data provides businesses and researchers a centralized location for searching and accessing Web databases. Web Data offers targeted access to information that text-based search engine sites do not. The Web Data portal provides fast and concise access to Web Data (SQLWebFAQ www.sqlWeb.com).

A Web database is a way of holding the WWW content, in a structured or useable format. A CGI (Common Gateway Interface) application is a program, which manages communication between a Web database and a Web server. These tools, with the help of the Internet, enable to build Wide Area Databases. Such a database lets users publish or collect information from anywhere in the world. This technology is at the centre of electronic trading, on-line education, and digital news services, see figure 2.2 (Jim B., 2000).

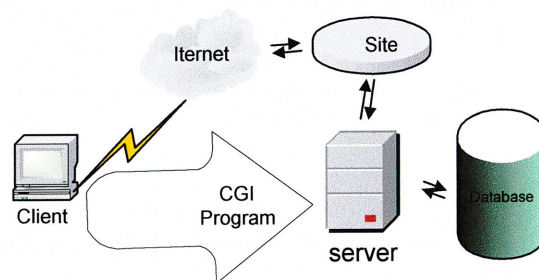


Figure 2.2. Web Database System

Web database is accessed On-Line via the Web application, specifically forms developed using standard HTML, ASP or XML. Using facilities available in HTML, applications programs on the Web server are accessed through a server-side mechanism defined as the Common Gateway Interface (CGI). This interface enables to create applications that integrate database functionality and provides access to organizational data repositories on behalf of Web clients. Applications can be designed solely for the purpose of querying a database and returning specific information. Also, the application can incorporate information pulled from a database for use as part of a larger application (Understanding Web Databases www.hubcom.com).

Web Database System is a system in which both Web and database technologies are used. These technologies provide wider access to database systems, ways of distributing these systems, and more services through the integration of such systems. In addition, data is distributed in such a way that users are made unaware of this. That means that the data can be stored where it is used (Berners-Lee, T. 1999).

Furthermore, a database is a structured method of organizing information. The purpose of a database is to make the information stored in it easy to search, so that specific information of interest can be found quickly and easily. Databases range from a simple list of information pertaining to an individual category, to many tables of information related in some way one to the other (Cattell R., 2000). The Internet provides many examples of data stored within databases. For example, stock quotes; job and movie lists, and airline flight numbers and schedules can be found throughout the Internet. In fact, a visit to any search engine such as

Yahoo.com will reveal the vast amount of information stored on the Web that is easily accessed by end-users. These databases can be found via an HTML form, which is easily built by programmers and easily accessed by end-users.

3.3 Migrating Database to the Web

There are many products like File-Maker and Tango, Cold-Fusion and so on, which are nice Web Database Server solutions. They can deliver data from a database, but they choke when trying to do too much business logic, try to make an interface that feels like more than a database, or if trying to scale them beyond the one database they were designed for. The Application servers tend to go way beyond just serving databases; it can often serve multiple databases and combine data, it can manage security, it can work with other systems (like ERP and commerce systems), and so on. An Application server does not only serve the database, but it provides a much bigger and broader solution. It is more about processing that data in new ways, and being the intermediary (middleware) between many business systems, or making the users feel like they are working with the data and adding value to it, instead of just letting them search and print reports based on it (Labrinidis A. and Roussopoulos N., 2000). Many Access applications were written by using “Microsoft Visual Basic for Applications” or the “Visual Basic for Applications Access user interface”. Applications that use Visual Basic for Applications as the development environment can run against SQL Server, using the Jet ODBC driver. Applications that use the forms and reports found in the Access user interface can access SQL Server using linked tables. If the application will use linked

tables, it is important to make sure that all Access tables get moved (migrated) to SQL Server to increase performance. Creating queries against a mix of Access (Jet) and SQL Server using linked tables can be very resource-intensive. The first step in migrating a file-server application to a client/server model is to ensure that the application works against the new database, see figure 2.3 (Barry E. and Mick R., 2000).

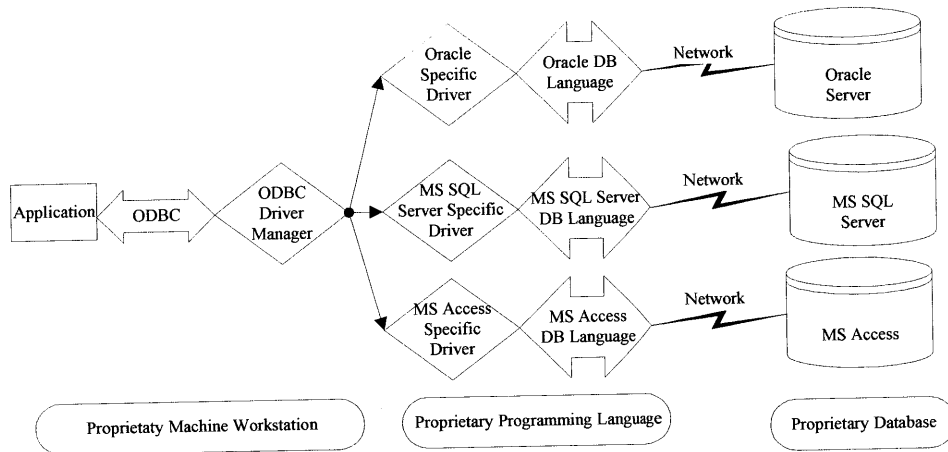


Figure 2.3. The Database Accessing on ODBC data

The next step is to optimise the application for the client/server environment by:

- Monitoring Transact-SQL statements being sent to the server: SQL Server Profiler is a useful tool for monitoring how Transact-SQL statements are sent to the database. It can be run as an unmodified Access application on SQL Server, which then sends sub optimal Transact-SQL to the database by using Data Access Objects (DAO) with the Jet/ODBC driver. For example, a DELETE statement that uses the Jet/ODBC driver to delete 1,000 rows makes 1,000 calls to the database, negatively impacting the performance of a production database. In this example, SQL Server Profiler displays 1,000 DELETE statements, allowing modifying the application to use Microsoft

ActiveX Data Objects (ADO) with the Microsoft OLE DB Provider for SQL Server, and, thus, it improves the application's efficiency.

- Implementing efficient indexes: After determining that the Transact-SQL statements being sent to the database are efficient, it should be possible to fine-tune those statements by using indexes more effectively. The Index Tuning Wizard allows finding bottlenecks, and it makes recommendations to help users to resolve these bottlenecks. Transact-SQL statements are not modified, but their performance improves with the correct use of indexes (Jim B., 2000).

The process of migrating a database to the Web must be constructed within the client-server architecture. To recap, a Web database system will involve the following parts as illustrated in figure 2.4:

1. Client systems: these display the Web pages that provide the user interface to the Web database system, by using a browser.
2. Web Server systems: these store the Web documents, scripts and programs.

The Web documents are the marked-up versions of the Web pages that are displayed by the browsers, and may also include scripts relating to the dynamic aspects, such as retrieval, manipulation and updating of data in the database.

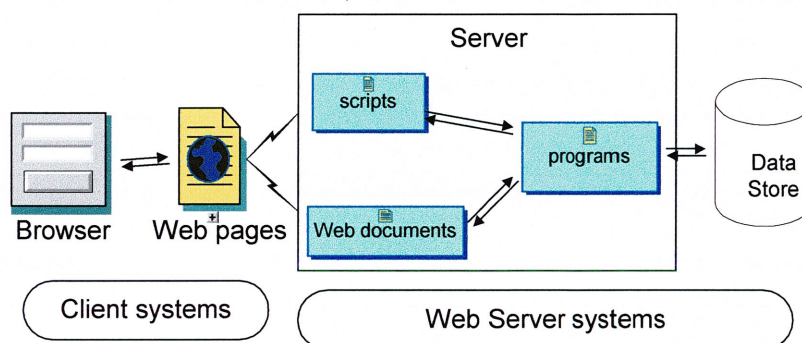


Figure 2.4. Client/Web Server Systems

The database system component may be implemented as extensions to the servers or browsers, or may be external to the Web. The application software may therefore be constructed either to execute within the relevant clients “client-side” or the relevant servers “server-side”. In either case, the software that executes can act as a (gateway) to some other system that provides database services (Barry E. and Mick R., 2000).

3.4 Building A Web-based Database

Any database that has an Application Programming Interface (API) can be accessed from Common Gateway Interface (CGI) programs and various gateway products. This is tremendously useful to organizations with legacy databases and information repositories that want to make this information accessible to internal employees and/or external users. Because the database and query models already exist for legacy databases, often all that is required is to develop the interface. (Often, this is just a matter of developing HTML documents; with perhaps the addition of some HTML tags extensions that support SQL query development within HTML), see figure 2.2 (Jim B., 2000).

Gateways are being developed for desktop databases such as Microsoft Access and FoxPro. In addition, CGI programs are commonly being developed in languages such as Visual Basic and Visual C++ and are using database APIs or ODBC functions.

The widespread of PC web servers has allowed many small companies to build their Intranet structures, which store each company's databases and make data easily accessible to all employees via their desktops.

CGI database programs and scripts contain the code that accepts data from the Web server (most often passed on from its Web clients) and uses the data as the basis for forming a query (SQL) to a database. These CGI programs generally are responsible for capturing database query results, formatting the results, and generating the appropriate HTML necessary to present these results to the user. It should be noted that any language, which is supported on the Web server host machine and also supports access to the database via a database API or ODBC, could be used as a CGI development platform. Strictly speaking, however, CGI languages must meet three requirements as illustrated in figure 2.5:

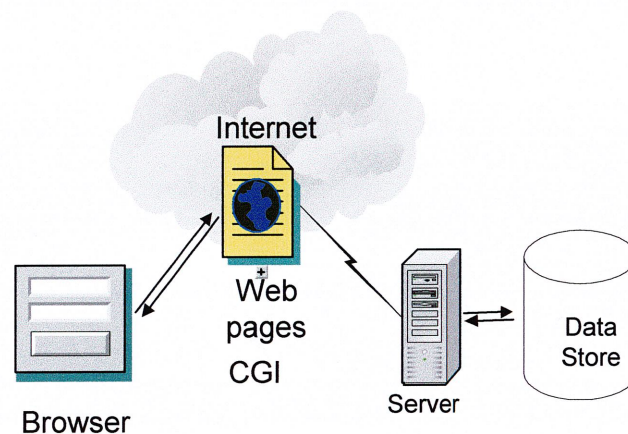


Figure 2.5 CGI development platforms

- The language must provide a mechanism for accessing standard input and standard output on the server. This enables user input from HTML forms (using the POST method) to be obtained. The capability to write to standard

output enables sub-forms, database query results, and other on-the-fly HTML documents and data to be returned to the Web client.

- The language must provide access to CGI environment variables. This enables user input from HTML forms (using the GET method) to be obtained. It enables CGI applications to determine a variety of valuable information, such as the user's Web browser type, additional path information, IP address, and so on. Environment variables provide one mechanism for maintaining the state of user input between connections and between the Web client and server.
- The language must provide some mechanism or API for integrating or interfacing with other applications, such as a relational database system. One of the primary reasons for CGI applications is to extend the functionality of the Web server and to offer Web clients access to data, information, and services they otherwise would not be able to access. The capability to provide access to services such as databases requires some sort of API. The capability to integrate CGI applications with other custom-developed applications or across multiple APIs enables these applications to offer more complex and functional services to users (Zhu L., Ren Y. and Hu J. 2001).

Potential CGI languages for database applications include (but are not limited to) compiled and interpreted languages, such as C, C++, FORTRAN, Perl, UNIX shells, Awk, Expect, and Tcl (and some variants that support Sybase and other database access), as well as some of the newer Perl extensions for client/server database access, for example, Sybperl and Oraperl, (Barry E. and Mick R., 2000).

HTML forms provide the standard interface by which users interact with a Web database application. Typically, an HTML form includes user-input objects such as checkboxes, radio buttons, scrollable menus, and text-input boxes. These input elements enable the user to make selections or provide input that ultimately is used by CGI programs or database gateways to formulate SQL queries to a database when the form is submitted to the Web server. This input can be used to fill in one or more parts of a standard SQL query, such as a table; fields in a SELECT statement; or values required to generate WHERE, ORDER BY, GROUP BY, or COMPUTE clauses. Well-designed forms abstract the database details from the user simply by providing the input objects and range of choices appropriate for each (Jim B., 2000).

Designing a Web Database system is the same as the traditional designing of a Database with two additional terms:

1. Web page design: which is concerned with: A) Web Data representation, such as presentation of data as Web pages, which is retrieved from the database or input by the user. B) Web Data association, such as the design of the links for navigation within and between Web pages. C) Web interface design, such as the design of Web page features, including the use of graphics, animation, etc.
2. Design of the connectivity between Web pages and the database: which is concerned with: A) Web database logical mapping, such as definition of the mapping between the data displayed on the Web pages and the data stored in the database. B) Web database physical mapping, such as implementation of

the mechanisms by which data is passed between the Web pages and the database (Hilton C. and Willis J., 2000).

3.5 Advantages of managing Web-based Database Systems

The benefit of a Web database is a two-way conversation with a client. There are other ways of achieving database interaction on the Internet with existing Web Database systems.

An organization might want to use databases in its Web applications for several reasons:

- To better manage the serving of large, document-based information repositories to internal and external users of the information.
- To have a wider access to a database by connecting a database system through the Web
- To better leverage and use legacy database systems, the information they contain, and existing applications.
- To unlock the potential of unused data held in organizational databases. Data from databases within various parts of an organization (for example human resources and project management) can be consolidated by using Web-based applications and served to users as though it were from a single source.
- To extend the functionality of Web servers so that information maintained could be made available to the general public and internal users. This functionality is currently sought by many organizations and government agencies whose primary product is information.

- To extend the functionality of Web servers so that new and expanded services can be provided to Web surfers (Chain L., 1999).

Web applications leverage benefits of the client/server model of computing.

Because an application resides on the Web server host (or other computers it subsequently accesses), the application is developed for a single runtime environment. This means that any computer running a browser, whether it's a PC running Windows, a Macintosh, or a UNIX workstation, can access the same applications. Because HTML provides the capability to easily invoke server-side programs via CGI or a Web server API, developers only need to know how to work within the CGI/server API environment in order to integrate CGI/API programs with new or existing applications such as databases or graphics-generation utilities (Cattell R., 2000). To programmers, this is often no more than an extension of development environments with which they are already familiar.

Databases can be accessed in CGI programs using familiar database such as APIs. CGI/server APIs are the glue that connects HTML GUIs on a Web browser to full-fledged applications on a Web server. Anyone who has developed applications using a windowing system, such as X Windows System, will attest that the most time-consuming aspect is coding the user interfaces. All the work of the application is done in the callback routines invoked from the interfaces, however.

Using HTML (as well as ActiveX controls and VBScript) as a GUI development environment reverses the above situation. Often, browser-based interfaces can be developed in a matter of minutes or days rather than weeks or months, which enables developers to concentrate on the development of the underlying

application. The result is shorter development cycles, allowing applications to reach end-users more quickly. Also, modifications and enhancements to the HTML and VBScript code are typically much faster than traditional development environments (Hilton C. and Willis J., 2000).

Furthermore, the primary benefit of Web databases is that users of Web database applications typically do not need to know a query language or the underlying database table structures and relationships in order to submit a query to the database. Users generally interact with database applications via HTML forms developed for them by the application developers. User input to these forms then is used to formulate a query that is sent to the database by a CGI program or database gateway program (Barry E. and Mick R., 2000).

Web page databases accessing small files on the client computer are usually faster than searches against a server side database or searches for text in an HTML document. Using traditional CGI search techniques can frustrate users who do not get quick results or any results. Many client server systems are not designed for repeated searches because they forget the previous search criteria. This forces the user to type in all the criteria for each search request. Browsers do not provide search functions necessary for effective data query and reporting. The browsers "find" command does not allow complicated queries that require searching more than one column within the same record, (Nielsen J., 2000).

Web page databases are stored on the Web server as a Java class file and an optional sequential text file. These files are delivered over the Internet when a Web page user requests the HTML that includes them. After the download, the Web page database performs the query and report processing on the client

computer. The Internet server only delivers documents and files to the client computer. This reduces the load on the server resources such as CPU and data transmission (Barry E. and Mick R., 2000).

3.6 Difficulties of managing Web-based Database Systems

Some difficulties face the developers when they attempt to manage Web database systems, such as Client/Server problems, validity, functionality, and stateless operation problems. The database development process on the Web is associated with a large number of technical decisions and, thus, requires deep understanding of differences between the client/server systems and the Web architecture. One of these differences is that the client/server program in local computing network can use PC-client resources to perform the task of processing and keeping data. On the other hand, DBMS for the Web remains as an old program for PC, where the browser plays the role of the “thin” client. As a rule when programming for a local computing network, programmers can utilize all computing resources of the client, while when programming for the Web, the developers are limited by the HTML standard.

Client/Server systems may execute on different computers, possibly with different operating systems, architectures, etc. However, they may often execute on the same machine. For example, the client may be literally a system on the individual user’s desktop in the form of a PC or workstation. Alternatively it may execute on a larger computer, implemented as a local server system but operating as the client to a mainframe, for example, in an organisation’s remote headquarters. Often one reason for the latter situation is to minimise the amount of

network traffic between clients and servers of Web Database Systems (Cattell R., 2000).

In general, client-side scripting is more appropriate for implementing small-scale functionality, such as animation of Web pages, data validation and calculation, whereas server-side should be used for large scale complex processing. One obvious practical advantage to distributing processes on this basis, between the client and server, is that if some functions take place at the client, a number of possibly slow connections can be avoided. However, the consequences of the allocation of processing to the client or server require careful analysis, both from performance and human computer interaction perspectives (Alexandr Z. and Natalya R., 2000).

The most important problem, which meets the developers, is the unstructured nature of Web. The Web servers answer on the query, sending the HTML page or starting external application program by means of Application Program Interface (API) server. As soon as the page has represented or executed the API program, transactions will be finished and connection will break. The server does not provide important information about the application program and user. This approach is good when spreading hypertext documents, but it is a serious problem for anyone, trying to create a compact program for working with a multi-page database. For solving this problem Web-program need has means session information service or data exchange means about the system with the user by means putting information in HTML forms (Alexandr Z. and Natalya R., 2000).

Another complex problem for Web applications is the unpredictable overload. The overload control on the Web must consider some important factors: 1- the ability

of server applications to control a large number of simultaneous connections. In the very complex situations, distributed architectural processing is used, which gives direct queries to several workstations. 2- the quantity of required loading (Alexandr Z. and Natalya R., 2000).

Web database system is made complicated by the problem of optimum processing distribution, and by the wide rang of alternative technologies and approaches available to the implementer. However, there are some disadvantages for client and server side approaches:

- Disadvantages of the client-side: 1- Environment dependencies: server-side processing may be dependent on the features of the platform and browser of the client where it is executed. 2- Security: the source code, delivered from the server, is not secured. 3- Download time: because the pages must also include the code to be executed by the client they will be larger and take longer to download. 4- Programming limitations: the restrictions on processes executing in the client mean that client side processes cannot access certain systems and there is only limited error handling and debugging support.
- Disadvantages of Server side: 1- Difficulty: There is generally a lack of debugging tools for server-side scripting. Also, this tends to be used for more complex problems, and the programming environment and languages are consequently more complex. 2- there is no direct control over the user interface (Barry E. and Mick R., 2000).

Also designing the interfaces for the Web is a serious problem. Many tools allow developers to programme only manually. However, more serious program

products allow the generation of HTML. There is also a set of the standard HTML forms. A good instrument for the Web must be based on a good programming language, and can utilize the main Web components and powerful debugging means (Zhu L., Ren Y. and Hu J., 2001).

One more serious problem is flexibility and adaptability in developing software for specific customers. The developers of large client/server systems are proud of their own single-purpose-rework internal languages, but real adaptation comes with more difficulties. In the case of Internet technologies, a developer, as a rule, will send all system texts to a customer. But system texts present the programs, written in DHTML, XML and JavaScript languages. The learning specialists and reworking take sufficiently little time, because all these technologies are absolutely known (Alexandr Z. and Natalya R., 2000).

3.7 Security Concerns

In building an on-line database, one must discuss the issues relating to security. In today's business environment, security has become a major concern for businesses, customers and government agencies. There is a number of Internet legislative bodies throughout the world that continue to issue statements and rules regarding Internet Safety and the protection of customers and businesses who have made the Internet their medium of exchange. The possibility that data can be lost, manipulated and stolen raises difficult problems for the Web developers, who must continue to investigate new ways to make the Internet a safer place for all its players. In addition to building firewalls, setting up online training classes for employees and hiring database specialists, many businesses have opted to insure themselves against data loss or data manipulation. However, the costs of such

insurances are high and continue to rise as the demand for such services increases. In addition, many companies keep back-up copies of their databases. These back-up copies are kept totally private and are never connected to the Internet because the more secure system is the one that is not exposed to the public via the Internet.

3.8 Summary

This chapter discussed the definition of Web-based Database Systems, and illustrated the changes involved in transferring a traditional database structure to a Web-based structure. It also presented the advantages and disadvantages of the new methods of managing Web-based Database Systems. Finally, it presented a brief overview of some of the security issues associated with Web database systems.

Chapter Four

Development and Implementation of an Online Payroll Prototype

- 4.1 Introduction
- 4.2 Characteristics of the proposed System
- 4.3 Development Methodology
- 4.4 System Specifications
- 4.5 Analysis and Design
- 4.6 Prototype Building
- 4.7 Testing the hypothesis

Chapter Four

Development and Implementation of an Online Payroll

Prototype

4.1 Introduction

The literature review chapters have made it clear that there was no migration difficulty in transferring a traditional database to the Web. The objective of this chapter is to conduct a small study to verify the benefits of Web-based databases. The chapter will describe the methodology used in conducting this study. It will include: characteristics of the proposed system, development methodology, an overview of the selected company, analysis and design, prototype building, and finally testing the hypothesis.

4.2 Characteristics of the proposed System

Information Technology Resources (ITR) has been chosen because: 1- it is a small company. This feature allows the building of a prototype without difficult problems such as managing huge data or taking too much time. 2- ITR wants its database to be transferred to the Web. The company's database; namely its payroll system, will provide a good sample of migrating database to the Web.

4.3 Development Methodology

The development methodology of the project will help to build an online payroll prototype. To build such a prototype effectively, it is important to implement an

on-line payroll prototype for ITR by following the Systems Development Life Cycle (SDLC), which includes:

- 1- Studying and investigating the current system
- 2- Preparing user requirements
- 3- Analysing user requirements
- 4- Designing the prototype
- 5- Building the prototype
- 6- Testing and evaluating the prototype

In designing the prototype, the Data Flow Diagram (DFD) and Entity Relation (ER) will be used to improve the development methodology of the prototype.

4.4 System Specifications

Information Technology Resources (ITR) is a small company, which sells Software and Hardware. The company has a number of employees who work as Sales assistants, engineers, technician and programmers. It has three managers: the first is the executive manager, the second is the personal manager and the third is the sales manager.

The current payroll system for ITR runs on Microsoft Windows. It has three main forms:

1. Staff Information Form: This form allows the managers to add, edit or delete records that have all needed information about the employees. (See Figure 4.1)

ID/CPR: 3		Employer Name: Rajo	
Position Title: Engineer		Annual Leave Balance: 5.25	
Basic Salary: 150,000 BD	Birthday: 10/19/1968	Service Date: 1/2/1997	Sick Leave Balance: 3.75
Address: Manama		Absent: 5	
Telephone: 32143243			
Nationality: Indian			

Figure 4.1: Staff Information Form

2. Leave Form: This form allows the managers to add, edit or delete records that have information about Annual Leave, Sick Leave and Requested Period of Leave for all employees. (See Figure 4.2)

ID: 3	Name: Rajo
Serial No:	Advance Payment Requested: Yes
Date:	2. Requested period of leave:
3. Balance of:	From:
Annual Leave: 5.25	Until:
Sick Leave: 3.75	Date returning to duty:
	Number of weekly and official holidays in leave period:
	Number of days leave requested:

Figure 4.2: Leave Form

3. Payroll Form: This form is for adding, editing or deleting records that have all information needed for payroll records. (See Figure 4.3)

Employee Information		Job Details		Absence		Overtime		Leave	
ID/CFR	Employee Name	Slip No	Date	Month-Year	Job	Absent	Overtime	Earned Leave	Annual
3	Alago	8	2/17/2000	Feb-2000	30	1	3	10	1.75
Basic Salary		Amount		Amount		Amount		Sick	
0.000 BD		5.000 BD		2.344 BD		0.000 BD		1.25	

Allowances		Deductions	
1. Transport	0.000 BD	1. Social Insurance	0.000 BD
2. Social	0.000 BD	2. Loans	0.000 BD
3. Others	0.000 BD	3. Rents	0.000 BD
		4. Others	0.000 BD

Leave Used	
Annual Used	A Leave P. Sick Used S. Leave P.
0	0.000 BD 0 0.000 BD

Remark: **Net Pay**
147.344 BD

Figure 4.3: Payroll Form

This system has been developed using MS Visual Basic.

4.5 Analysis and Design

The current system works well (see figure 4.4), but it does not cope with the expansion of the company and advancements in new technologies because:

1. The company will open new branches in different locations and has to distribute and exchange data between branches quickly, easily and in the most cost-effective manner.
2. The managers are very busy and need to travel for conferences, exhibitions and business trips. They need to do their job and at the same time process their employees' personal and payroll requests in the company.

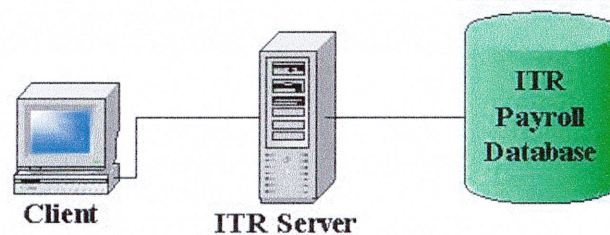


Figure 4.4: Current ITR Payroll Database System

ITR Company intends to develop its system by migrating its existing Database to a Web-enabled environment (On-Line Payroll) as indicated by figure 4.5. This

allows the managers to access the Database Payroll System on-line throughout the world easily and inexpensively.

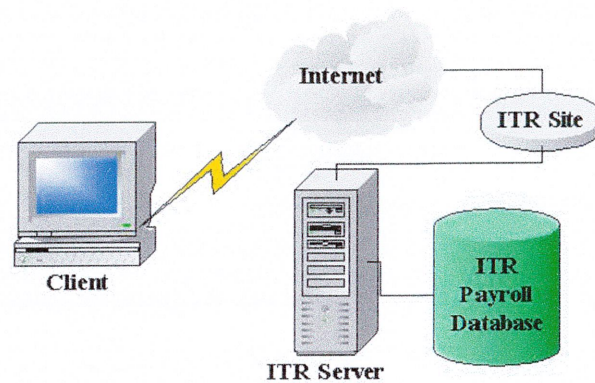


Figure 4.5: Web ITR Payroll Database System

The ITR Payroll Database System has three main tables, which are:

Employee Table	Payroll Table	Leave Table
Record ID Name Basic-Salary Position_Title Service_Date Annual_Leave Sick_Leave Absent Birthday Nationality Address Telephone	Record Slip_No ID Date Month Basic_Salary Job_Days Absent_Days Overtime25% Overtime50% Trans Social A_Others Social_Insurance Loans Rents D_Others Net_Pay Earned_AnnualLeave Earned_SickLeave AnnualLeave_Used SickLeave_Used Remarks	Record ID Date Advance_Payment LeaveFrom Until Date_Reurning_To_Duty Number_of_Days_Leave_Requested Number_Holidays_In_Leave_Period Paid

Table 4.1: ITR Payroll Database Tables

Using the scenario that described the ITR Payroll Database System earlier, a DFD has been developed, as showing in Figure 4.6.

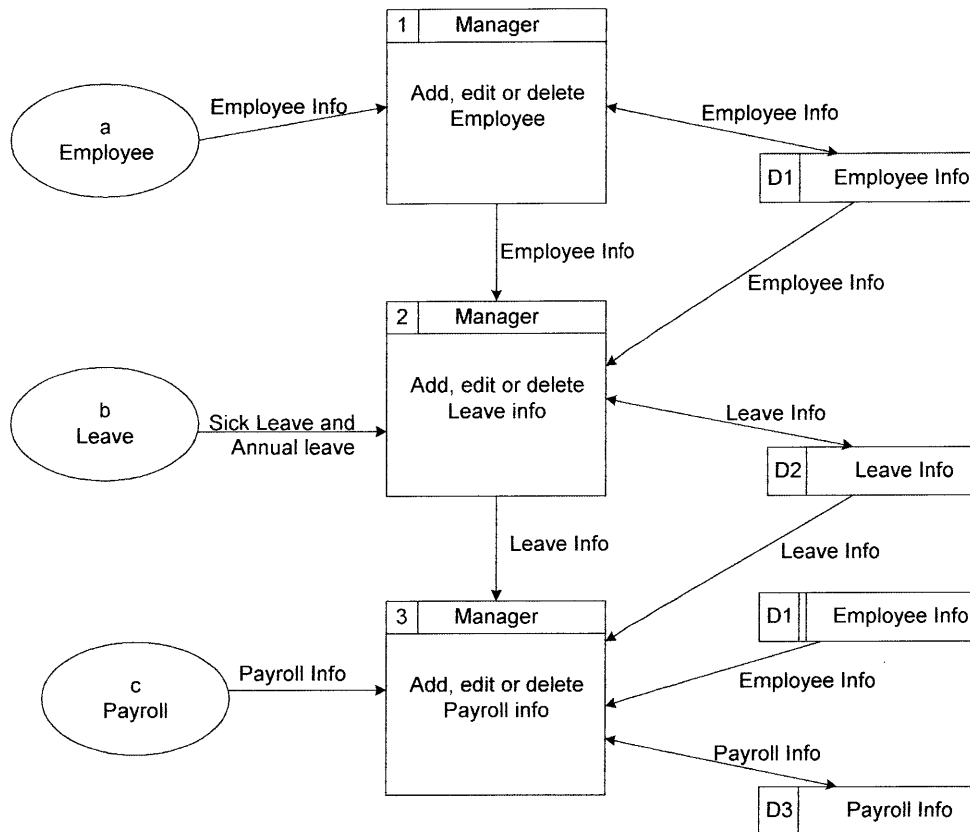


Figure 4.6: Data Follow Diagram DFD for ITR Payroll Database

As shown in the above diagram, there are three entities for the system, which are Employee, Leave and Payroll. It is important to show the Entity Relationships between entities to illustrate the benefits of data modelling.

Entity	Key Attribute	Some non-key Attributes
Employee	Employee Code	Leave Code Payroll Number
Leave	Leave Code	Employee Code Employee Name Payroll Number
Payroll	Payroll Number	Employee Code Employee Name Employee Address Leave Code

Table 4.2: Entities with their key and non-key attributes

	Employee	Leave	Payroll
Employee		X	X
Leave	X		X
Payroll	X	X	

Table 4.3: Relationships between entities

	Employee	Leave	Payroll
Employee		1:M	1:1
Leave	M: 1		M: 1
Payroll	1:1	1:M	

Table 4.4: Entity relationship grid

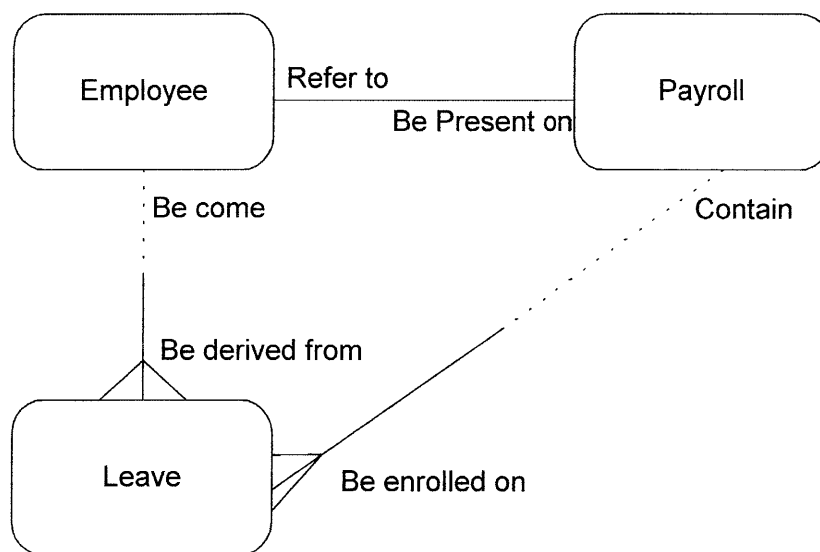


Figure 4.7: Entity Relationship ER for ITR Payroll

4.6 Prototype Building

As the client's needs grow, and the demand for a high-performance database system increases, the client wants to move from the file-server environment of the Microsoft Access Jet engine to the client/server environment of Microsoft SQL Server and move Access tables and queries into SQL Server 7.0.

The prototype has been built for ITR Company using DHTML in MS Visual Basic6. The prototype uses a copy of the original database of ITR, and therefore, no new data is created. The prototype has one HTML form as the main menu to enable the managers to access the Payroll Database through the Internet as shown in figure 4.8.

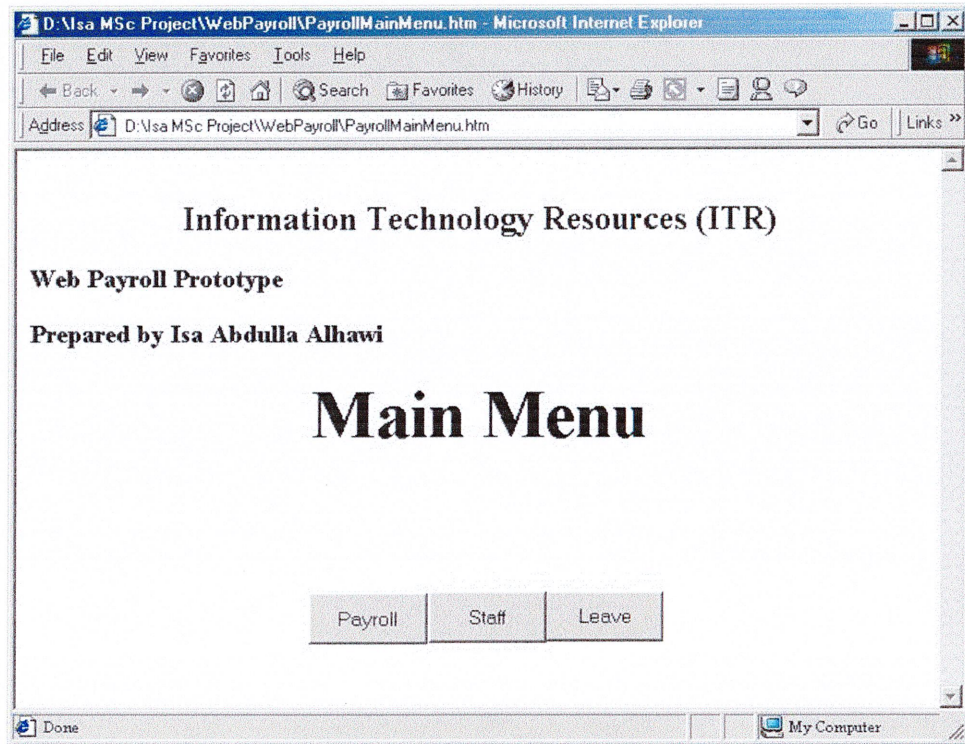


Figure 4.8 Web Payroll Main Menu

The main menu has three buttons, which are Payroll, Staff and Leave. These buttons allow users to access the Payroll, Staff and Leave forms:

1. Payroll form: This form allows adding, editing and deleting records that have the payroll information as shown in figure 4.9.

Information Technology Resources (ITR)
Payroll Information Form

ID/CPR	Employer Name	Slip No	Date	Month-Year
Job	Absent	Overtime 25%	Overtime 50%	Earned Leave
Job Days	Absent Days	Hours	Hours	Annual
Basic Salary	Amount	Amount	Amount	Sick
Allowances		Deductions		
1. Transport:		1. Social Insurance:		
2. Social:		2. Loans:		
3. Others:		3. Rents:		
		4. Others:		
Leave Used		Net Pay		
Annual Used	A. Leave P.	Sick Used	S. Leave P.	
Remark				

Save

Open

Delete

Search

Print

Exit Payroll Information

Web Payroll Prototype
Prepared by Isa Abdulla Alhawi

Figure 4.9. Web Payroll Information Form

2. Staff form: This form allows the managers to add, edit and delete records that have all needed information about the employees as shown in figure 4.10.

Information Technology Resources (ITR)
Staff Information Form

ID/CPR	Employer Name		
Position Title			
Basic Salary	Birthday	Service Date	
Address			
Telephone			
Nationality			
Annual Leave Balance		Sick Leave Balance	
Absent			

Save

Open

Delete

Search

Print

Exit Staff Information

Web Payroll Prototype
Prepared by Isa Abdulla Alhawi

Figure 4.10 Web Staff Information Form

3. Leave form: This form allows the managers to add, edit and delete records that have information about Annual Leave, Sick Leave and Requested Period of Leave for all employees as shown in figure 4.11.

Information Technology Resources (ITR) Leave Information Form	
ID:	Name:
Serial No:	Advance Payment Requested:
Date:	2. Requested period of leave: From: Until: Date returning to duty: Number of weekly and official holidays in leave period: Number of days leave requested:
3. Balance of: Annual Leave: Sick Leave:	
Save Open Delete Search Print Exit Leave Information Web Payroll Prototype Prepared by Isa Abdulla Alhawi	

Figure 4.11 Web Leave Information Form

4.7 Testing the hypothesis

By migrating the traditional database of ITR Payroll System to a Web-enabled environment (On-Line Payroll), the new system meets the user requirements in the following ways 1- it allows the distribution and exchange of data between branches quickly and cost-effectively. And 2- it allows the company to access its online database easily and inexpensively throughout the world. Furthermore, it is flexible and welcomes new modifications and enhancements, and can work with different platforms on the Internet.

Chapter Five

Evaluation and

Findings

- 5.1 Prototype Evaluation
- 5.2 Project Evaluation and Findings

Chapter Five

Evaluation and Findings

5.1 Prototype Evaluation

The prototype has a number of strengths and weaknesses. The strengths include:

- Meeting the customers' requirements: with online databases, managers can have direct access to important data from any point on the globe. Thus, they can always make sure that any changes made to the database are accurate and consistent with the overall objectives of their companies.
- The online database can be easily accessed: the online database enables users of the network to access the database without having to physically connect their hardware to any special portals. Therefore, branches of a company can have easy access to information and communicate instantly from any point on the globe.
- The online database is easy to modify: online databases allow users, with special access codes, to modify the database whenever needed.
- The prototype supports different platforms.
- The building of such prototypes is not expensive, and they are easy to maintain.

Weaknesses of the prototype include:

- Security is not yet assured: security presents a major problem to all people working online. Thus, this problem is not unique to the prototype itself. New measures of safety ought to be designed to insure that online transactions are kept private and online databases are kept secure.

- Change can be difficult to implement throughout an organization: the change from a traditional database to a Web-based database requires training employees and getting them ready to utilize the new technologies. However, some employees may find it hard to adapt to such changes, and may, in fact, try to prevent such changes from happening.

5.2 Project Evaluation and Findings

The project has successfully illustrated the process of transferring a traditional database to the Web. By doing so, this project has achieved its main objectives, which were stated in chapter 1. First, the project has presented a thorough analysis of database management on the web and distributed environment. Second, It has introduced the main techniques and methods used to migrate a traditional database to a web-oriented environment. Such techniques included File-Maker, Tango, MS Access and Cold-Fusion. Third, a prototype has been developed to show the steps involved in the migration process of traditional databases to the Web. The project has also succeeded in verifying the first hypothesis as stated in the first chapter. Clearly, migrating a traditional database to a web-oriented environment can be done with no difficulties. In fact, as the development of the prototype illustrated, such a migration can be done easily. In addition, the research has proved that the second hypothesis should be rejected. The research has shown that there are, in fact, operational differences between a web-based database and a traditional database. It has been shown that the operations of web-based databases have three distinct points. First, it can be managed easily from any point on the globe. Second, access to the database can be granted to interested parties who can then access the database without having to physically be located close to its networks.

Third, web-based databases support different platforms, thus, adding more flexibility to the system.

However, the world of database management is constantly changing, which makes it impossible for any research to encompass all that is happening in that field. For example, the recent development of the so-called “page-based and component-based web applications” promises a new and more exciting future for the applications of web servers and the management of databases. However, these developments are yet to be implemented worldwide.

In addition, the research could not present the full consequences of migrating a traditional database to the Web. For example, it was beyond the scope of this research to illustrate the cultural changes that should occur in an organization as a result of transferring its database to the Web. Although these changes are important, they were somewhat irrelevant to the subject of this research.

Nevertheless, such changes should be studied in a separate research if a full evaluation of database management is to be achieved.

Furthermore, the project faced a number of constraints. First, time was limited, and thus, the research could not cover all aspects of database management.

Second, there has been a lack of material availability. Many resources had to be ordered and some information could not be obtained. These constraints affected the quality of this research to some extent. However, the author of this project made all efforts to produce a well-written project using available data.

Chapter Six

Conclusion and Recommendations

- 6.1 Conclusion
- 6.2 Recommendations

Chapter Six

Conclusion and Recommendations

6.1 Conclusion

As mentioned earlier, the findings of this research has shown that migrating a traditional database to the Web can be done with no significant difficulties. It was also concluded that there were, in fact, operational differences between a traditional database and a web-based database. The research has shown that it is beneficial for companies to migrate their databases to the Web to gain a competitive advantage, speed up inter-company communications, better serve their customers, reduce costs, and join the ever-increasing number of online players.

6.2 Recommendations

The author's recommendations are:

- Other projects should be conducted to study other aspects of database management. For example, some projects should concentrate on the effects of database migration on corporate cultures and employee morale. In addition, some projects should be conducted to specifically look at the aspect of security in database management, since security has become the major issue in the computer and Internet world.
- Managers and computer analysts should embrace database migration and encourage their organizations to make such a change to insure themselves a competitive advantage in the business and computer worlds.

Appendices

- A. Project Terms Of Reference
 - B. Deliverables List
 - C. Schedule
 - D. Gantt Chart
-

Appendix A

**University Of Sunderland
School Of Computing and Information Systems
MSc Computer Based Information Systems
Isa Abdulla Salman Alhawi
(Project Terms Of Reference)
Evaluation Study In
Migrating Database System to Web-Based System**

Overview

All companies search for a method, which can be accessed to Data very easily, and cheaply by using a good analysis and design. Most of companies manage their Database by using different programming languages such as C++, VB, Oracle, Informix or Java. But they face a lot of difficulties in developing, maintenance and exchanging data between different servers. This happens so often because most of the methods used for managing Database are very costly and difficult to be built and handled easily.

Web applications are becoming more and more popular in that field. Since the use of the tools and technologies is very required to develop the old methods.

This project is concerned with the evaluation of the new methods which have already migrated Database system into new technique in Web Base system e.g. SQL. Applying the results will implement a Web Base system for a payroll, which allows the companies to access their own DataBase On-Line easily, and inexpensively throughout the world.

The evaluation of this Project will provide a platform for a Database Payroll System for a Company in Bahrain called ITR (Information Technology Resources), which allows the user to access it On-Line. The project shall then be critically evaluated for its strengths and weaknesses.

Objectives

- Research into methods of managing Database on Web.
- Research into distributed Databases.
- Development and Implementation of a Web based on-line Payroll Prototype
- Critical evaluation of the outcomes of the Project

Constraints

- Limitations of time and acquaintance with SQL and Microsoft Visual Basic 6.0 will provide constraints to the completed project.
- Lack of resources in libraries and bookstores in my country.

Resources

- PC with MS Word
- Microsoft Visual Basic 6.0

Reporting

Reporting at regular intervals to Dr. Mike Wyvill at University of Sunderland.

Appendix B

Evaluation Study In: Migrating Database System to Web-Based System

Status of Project Deliverables

Item No	Deliverable	Description	Status
1	Project TOR	Submitting Project Term Of References to the supervisor.	Completed
2	Schedule & Gantt Chart	Submitting Schedule & Gantt Chart to the supervisor.	Completed
3	Research Outline	Submitting Research review outline to the supervisor.	Completed
4	Research report: - Chapter 1 - Chapter 2 - Chapter 3 - Chapter 4 - Chapter 5 - Chapter 6 - List of Ref. - Contents Page	Introduction Traditional and Distributed Database Systems. Database Systems On Web Development and Implement of Web-Base on-line Payroll Prototype Evaluation Conclusion Reading and research references Contents page of the research	Completed Completed Completed Completed Completed Completed Completed Completed
5	System Developing - Study current system - System requirement - Analysis & Design - System Building - Test the System	Produce a report on the current document flow procedure Produce system requirement & specification Context Diagram, Data flow Diagram and Entity Relationship Coding and building the system Testing all the system function and confirming that it meets the client need. System documentation	Completed Completed Completed Completed Completed
6	Evaluation of the system	Evaluation report for the developed application	Completed
7	First Draft	Submitting First Draft dissertation report to the supervisor for reviewing.	Completed
8	Second Draft	Submitting Second Draft dissertation report to the supervisor for reviewing, after implementing the supervisor advice and comments.	Completed
9	Poster Presentation	Prepare poster presentation on A3 paper size to present the project.	Completed
10	Final Draft	Submitting two copies of the final draft dissertation to the supervisor	Completed

Appendix C

Schedule Version 1.0

Item	TASK Description	Hours	Planned Finish Date	Actual Finish Date	Task Deliverable
1	Terms of Reference	20	11.2.2001	11.2.2001	Terms of Reference
2	Receive/update Terms of Reference	10	17.2.2001	17.2.2001	
3	Task Schedule & Gannt Chart	10	27.2.2001	27.2.2001	Task Schedule & Gannt Chart
4	Receive/update Terms of Reference, Task Schedule & Gannt Chart	10	3.3.2001	3.3.2001	
5	Preparation of draft outline	20	15.3.2001	15.3.2001	Draft outline
6	Receive/update outline dissertation	10	25.3.2001	25.3.2001	Updated Draft outline
7	Preparation of list of deliverables	10	1.4.2001	1.4.2001	Draft list of deliverables
8	Preparation of material for current systems analysis	20	10.4.2001	10.4.2001	
9	Draft of list of Reference	20	28.4.2001	28.4.2001	Draft of list of References
10	Current system investigation	20	20.5.2001	20.5.2001	Current system report
11	Preparation of user requirement	15	4.6.2001	4.6.2001	
12	Analyse User requirement	20	18.6.2001	18.6.2001	
13	Design a prototype	25	5.7.2001	5.7.2001	Draft design a prototype
14	Building of a prototype	30	20.7.2001	20.7.2001	
15	Test of a prototype	20	6.8.001	6.8.001	
16	Evaluation of a prototype	15	15.8.2001	15.8.2001	
17	Prepare first Draft dissertation	20	1.9.2001	1.9.2001	First Draft dissertation
18	Prepare Poster Presentation	25	6.10.2001	6.10.2001	
19	Prepare second Draft dissertation	20	20.10.2001	20.10.2001	Second Draft dissertation

Appendix D

Gantt Chart

ID	Task Name	Duration	Start	Finish	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct
1	Project Period	186days	Wed2/7/01	Sat10/20/01									
2	Preliminary work	22days	Sat2/3/01	Sat3/3/01									
3	Initiate Project	2day:	Sa2/3/01	Mor2/5/01									
4	Terms of Reference	9day:	Tue2/6/01	Fri2/16/01									
5	prepare schedule	2day:	Mor2/19/01	Tue2/20/01									
6	prepare Gantt chart	6day:	Wed2/21/01	Wed2/28/01									
7	Finalised All	5day:	Thu3/1/01	Tue3/6/01									
8													
9	Research	99days	Thu3/1/01	Mon7/16/01									
10	Identify & Collect Refrencess	99day:	Thu3/1/01	Mor7/16/01									
11	Study Refrencess	99day:	Thu3/1/01	Mor7/16/01									
12	Prepare list of Delivrables	6day:	Thu3/15/01	Thu3/22/01									
13	Prepare research outlines	15day:	Fri3/23/01	Thu4/12/01									
14	Submit outlines to Supervisor	1day:	Fri4/13/01	Fri4/13/01									
15	Receive outlines	7day:	Mor4/16/01	Tue4/24/01									
16	Update outlines	3day:	Wed4/25/01	Fri4/27/01									
17	Prepare report sections	56day:	Mor4/30/01	Mor7/16/01									
18													
19	Analyse Requirments	3days	Tue7/17/01	Thu7/19/01									
20	obtain details from clients	1day:	Tue7/17/01	Tue7/17/01									
21	Analys clients requirments	2day:	Wed7/18/01	Thu7/19/01									
22													
23	System Development	16days	Fri7/20/01	Fri8/10/01									
24	desgin System	4day:	Fri7/20/01	Wed7/25/01									
25	Build System	6day:	Thu7/26/01	Thu8/2/01									

Continued

ID	Task Name	Duration	Start	Finish	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct
26	Evaluate System	3day:	Fr8/3/01	Tue8/7/01									
27	Prepare System Document	3day:	Thu8/9/01	Sur8/12/01									
28													
29	Produce desertation	52days	Sur8/12/01	Thu10/18/01									
30	Prepare Draft Desertation	16day:	Sur8/12/01	Fr8/31/01									
31	Submit Draft Desertation	1day:	Sa9/1/01	Sa9/1/01									
32	Receive & update Draft Dese	30day:	Mor9/3/01	Fri10/12/01									
33	Prepare Poster Presentations	1day:	Sa10/13/01	Sa10/13/01									
34	Complete Dissertation	5day:	Mor10/15/01	Fri10/19/01									
35	Submit final Desertation	1day:	Sa10/20/01	Sa10/20/01									

Note: The date is as USA format (e.g. Wed 2/7/01 = Wed Feb 7th 2001)

References

- A. Birrell and B. J. Nelson (1984) "Implementing Remote Procedure Calls", ACM Transactions on Computer Systems, Volume 2, Issue 1, pp.39-59.
- A. J. H. Peddemors and L. O. Hertzberger (1999) "A high performance distributed database system for enhanced Internet services", Future Generation Computer Systems, Volume 15, Issue 3, PP. 407-415
- Alexandr Z. and Natalya R. (2000) "Difficulties And Advantages Of Access To Corporative Information In Internet", IEEE, Tomsk, Russia PP. 112-117.
- Amir D. (1998) "Databases and the Internet", by Eric j. Gallegos
<http://www.amir.fullerton.edu/msis409/Fall%201998/Projects/proj10/Databases%20and%20the%20Internet%20Project.html>. Accessed 12th June 2001
- Ash, Tim, (April 1998) "Mission-critical Web databases" Performance Computing/Unix Review, Volume 1, Issue 1.
- Barry E. and Mick R. (2000) Web Database Systems, McGRAW-HILL Book Company Europe, England.
- Beg, M.M. Sufyan; Ravikumar, C.P. (2000) "Application of parallel and distributed data mining in e-Commerce", IETE Technical Review (Institution of Electronics and Telecommunication Engineers, India), Volume 17, Issue 4, PP. 189-195
- Berners-Lee T. (1999) "Weaving the web" Orion Business Books, London.
- Blakeley J. (1996) Data Access for the Masses Through OLE DB. In Proc. ACM SIGMOD Int. Conf. On Management of Data, PP.161-172
- Bouganim L., Florescu D., and Valduriez P. (1999) "Multi-Join Query Execution with Skew in NUMA Multiprocessors", Distributed and Parallel Databases.
- Casrto M., Adya A., Liskov B. and Myers A. (1997) HAC: Hybrid Adaptive Caching of Distributed Storage Systems. In Proc. ACM Symp. On Operating System Principles, PP. 102-115.
- Cattell R. (2000) "The Object Database Standard", ODMG 3.0, Morgan Kaufman, San Francisco.
- Chain L., Hsien C., Jiun L. and Feng W. (1999) "A Web Database Application Model for Software Maintenance", IEEE, National Chiao-Tung University, PP. 338-344
- D. Box (July 1997) "Q&A ActiveX/COM", Microsoft Systems Journal, pp.93-108.
- David B. and Jane G. (1992) Distributed Database Systems, Addison-Wesley Publishers Ltd. British Uk.

- Date C.J. and Darwen H. (1998) Foundation for Object/Relational Databases, The Third Manifesto, Reading, Mass.: Addison-Wesley.
- Dogac A., Kalinichenko, L., Ozsu M., and Sheth A. (1998) Advances in Workflow Systems and Interoperability, Berlin: Springer-Verlag.
- Elmagarmid. A., Rusinkiewicz M., and Sheth A. (1999) Management of Heterogeneous and Autonomous Database Systems, San Francisco: Morgan Kaufmann.
- Fessant F., Piumarta I. And Shapiro M., (1998) “An Implementation of Complete, Asynchronous, Distributed Garbage Collection”. In Proc. ACM SIGPLAN Conf. On Programming Language Design and Implementation, PP. 152-161.
- Franklin M. and Zdonik S. (1998) “Data In Your Face”, Push Technology in Perspective. In Proc. ACM SIGMOD Int. Conf. On Management of Data, PP. 516-519
- Fred R., McFadden, Jeffrey A. and Hoffer. (1994) Modern Database Management, 4th Edition The Benjamin / Cummings Publishing Company, Inc. U.S.
- Hilton C. and Willis J. (2000) Building Database Applications on the Web Using PHP3, Addison-Wesley.
- Jim B. (2000) Web Database Development, Step by Step. Microsoft Press, USA.
- Joe S. Guide To: Client/Server Databases, 2nd Edition Ziff-Davis Press. U.S.
- Kazerouni L. and Karlapalem, K. (1997) Stepwise Redesign of Distributed Relational Database. Technical Report HKUST-CS97-12. Hong Kong University of Science and Technology, Department of Computer Science.
- Kevin Reichard (October 1996). Web Servers for Database Applications
<http://www.dbmsmag.com/9610i08.html> Accessed 13 April, 2001
- Killen M (1992) SAA, managing distributed data, McGraw-Hill, Inc. U.S.
- Kung, H.T. and Wang, S.Y (1997) “Client-server performance on flow-controlled ATM networks: A web database of simulation results”, Proceedings - IEEE INFOCOM, Volume 3, PP. 1218-1226.
- Labrinidis A. and Roussopoulos N. (2000) “Generating Dynamic Content at Database-Backed Web Servers” Cgi-bin vs. mod-perl, ACM Sigmod Record 29(1). PP. 26-31
- Liu L., Yan L., and Ozsu M., (1998) “Interoperability in Large-Scale Distributed Information Delivery Systems”. In Dogac et al., PP.246-280

- Manjula K. and Johnny W. (2000) "Non-blocking distributed transaction processing system", Journal of Systems and Software, Volume 54, Issue 1, PP. 65-76
- Maria A., Harold F., Ruth W., Miyi C. and Kevin B. (1998) "An OO Database Migrates To The Web", Focus IEEE Software, PP. 22-30.
- M. Schrefl, E. Kapsammer, W. Retschitzegger and B. Proll (2001) "Self-Maintaining Web Pages – An Overview", IEEE, University of South Australia, PP. 83-90
- M. Tamer and Patric V. (1999) Principles of Distributed Database Systems, 2nd Edition Prentice Hall. U.S.
- Nguyen T. and Srinivasan V., (1996) "Accessing Relational Databases from the World Wide Web". In Proc. ACM-SIGMOD, Int. Conf. On the Management of Data. PP. 529-540.
- Nielsen J. (2000) Designing Web Usability, New Riders.
- Ramakrishnan R. (1998) Database Management Systems, McGraw-Hill, Inc. U.S.
- S. Vinoski (Feb. 1997) "CORBA: Integrating diverse applications within distributed heterogeneous environments", in IEEE Communications, volume 14, Issue 2.
- Selena Sol (August 16, 1998). Database for the web
<http://www.wdvl.com/Authoring/DB/Intro/index.html>. Accessed 19 February 2001
- Seng-cho T. (1998) "Migrating to the Web: a Web financial information system server", Decision Support Systems, Volume 23, Issue 1, PP. 29-40.
- Siegel J. (1996) CORBA Fundamentals and programming. New York, Wiley
- Shaku A (1992) Distributed database, cooperative processing, and networking. McGraw-Hill, Inc. U.S.
- Sten F. A., Joakim E. and Mats L. (2000) "Distribution of control using active storage", Control Engineering Practice, Volume 8, Issue 6, PP. 681-687
- Teresa H. (1995) Distributed Relational Database Architecture, 4th Edition Prentice Hall. U.S.
- Thomasian A. (1998), "Distributed Optimistic Concurrency Control Methods for High Performance Transaction Processing", IEEE Trans. Data and Knowl. Eng. Volume 10, Issue 1, PP.173-189.
- Zhu L., Ren Y. and Hu J. (2001) "Design and implementation of a Web-based supervisory control system for the communication network of electric power system", Dianli Xitong Zidonghuae/Automation of Electric Power Systems, Volume 25, Issue 2, PP. 56-59