Mat. No. 14901

St Clements University

Integrated Multi-Database Information Systems

A Thesis Submitted to the St Clements University in Partial Fulfillment of the Requirements for the **Degree of Doctor of Philosophy in Computer Science**

by

Haitham Sabah Hasan

Supervisor

Dr. Jane Jaleel Stephan

Feb	Sufar
2010	1431

جمهورية العراق

الجامعة سانت كليمنت

تكامل قواعد بيانات متعددة لنظم المعلومات

رســالــــة

مقدمة إلى جامعة سانت كليمنت كجزء من متطلبــات نيل شهــادة

دكتوراه في فلسفة في علوم الحاسبات

من قبل

هيثم صباح حسن

بأشراف

الدكتورة جين جليل اسطيفان

لز بيدي

1431 2010

بسم الله الرحمن الرحيم

قِالْوٌا سِبْحانِْك لا عِلْمَ لِْنا

إلا مْا عْلمَتْنَا

إنَّكَ أنَتَ الْعَلَيِمِ الحْكَّيَم

صدق الله العظيم

Dedication

To my beloved Father

The Teacher

His taintless soul

Resulted from soft touch of his gorgeous life

Spent for the big Family

Haitham

Examination Committee Certification

We certify that we have read this thesis entitled”**Integrated Multi-Database information System**“and as an examining committee, examined the student “**Haitham Sabah Hassan**” in its contents and in what is concerned with it and that in our opinion it meets the standards of a thesis for the degree of Doctor of Philosophy in computer Science.

Signature:	Signature:
Name:	Name:
(Chairman)	(Member)
Date:	Date:

Signature:
Name:
(Member)
Date:

Signature:	Signature:
Name:	Name:
(Member)	(Supervisor)
Date:	Date:

<u>SUPERVISOR CERTIFICATION</u>

I certify that this thesis entitled ”Integrated Multi-Database Information System” was prepared under my supervision by Haitham Sabah Hasan at the University of St-Clement as a partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science.

Signature:

Name: Assistant Professor Dr.Jane Jaleel Stephan

Date: / / **2010**

Acknowledgement

I Would like to express my gratitude to my supervisor (Assistant Professor) Dr.Jane Jaleel Stephan For her support, and assistance during the whole phases of my research, and for the technical literature , she provided me with during the work. Her assistance and encouragement have contributed essentially to finishing this work.

Special thanks are due to my mother, and my relatives for their endless support, understanding, and encouragement, who had their part of suffering and sacrifice during the entire phrase of my research.

Haitham

Abstract

A multi database consists of a collection of autonomous local databases. Systems used to manage multi databases are called multi database systems (MDBSs). In such a system global transactions are executed under the control of the MDBS. Independently local transactions are submitted directly to a local Database system (LDBS) by local applications. An Multi Database System should provide a mechanism to globally manage transactions. However global transactions are long- live and involve operation on multiple and autonomous local databases. Moreover MDBSs do not have any information about the existence and execution order of local transactions. The aim of this research is to propose a design that integrates three Heterogeneous multi databases with the same model relational but different software environments by doing the following:

1- Accepts a query, determine the appropriate set of information sources to answer the query.
2- Obtain results from the information so appropriate translation, filtering and
merging of the information and return the final answer to the user or application (client).
3- The server computer can manipulate transaction (Delete, update, search) while the client computer each of them can process (delete, update and search).

This system has minimum requirements (***Pentium 4***) with processor (***1.8 MHz***) and memory (***128MB***) by the operating system (***XP***) and has been designed with ***Visual Basic 6.0*** environment , Microsoft ***Access 2000***, Microsoft visual ***FoxPro***, **SQL Server 2000** through ***LAN*** network .

ملخص البحث

أن قواعد البيانات المتعددة تتكون من مجموعة مستقله من قواعد بيانات

أن الانظمة التي تستخدم لتديرقواعد البيانات المتعددة تسمى انظمة قواعد البيانات المتعددة في هذه الانظمة المعالجات تتعامل و تنفذ تحت سيطرة نظام قاعدة البيانات المتعدد. ان المعالجات المحلية الغير معتمدة ترسل بشكل مباشر و مستقل الى انظمة قواعد بيانات محلية من خلال تطبيقات محلية لذلك ان انظمة قواعد البيانات المتعدد يجب ان توفرميكانيكية للمعالجات العامة وعليه مثل هكذا تعاملات تتطلب حياة طويلة لقواعد بيانات محلية مستقله علاوة على ذلك ان انظمة قواعد البيانات المتعددة لا تحوي اي معلومات حول وجود وتنفيذ اوامر التعاملات المحلية.

الهدف من هذا البحث تقديم مقترح للتصميم الذي يقوم لدمج و معالجة ثلاثة مصادر من قواعد البيانات غير المتجانسة من نفس النموذج (العلائقي) تحت بيئة برمجية مختلفة .هذا بالاضافة الى دراسة موضوع المشروع الذي يهدف الى:

- تطوير فهم كيفية الاستفسار عن مصادر المعلومات الغير متجانسة المنفذة في بيئة موزعة
- تحصيل النتائج من خلال مترجم متخصص لفلترت و دمج المعلومات وارجاع الاجوبة النهائية للمستخدم او تطبيقات الزبون.
- تتم عمليات المعالجة (حذف و تحديث و بحث) عن طريق الحاسبة الخادم (Server) اما حاسبات الزبائن (client) الاخرى فتتولى كل منهم عمليات التحديث و الاضافة و البحث الخاص بها.

لقد نفذ هذا المشروع على حاسبة (4) Pentiumذات سرعة (1.8 MHz)و ذاكرة خزن (128 MB) بنظام تشغيل XP حيث تم تصميم النظام باستخدام لغة البرمجة فيجوال بيسك (Visual Basic 6.0) وقواعد البيانات تم بنائها باستخدام (Microsoft Access 2000)و(Microsoft FoxPro) و (SQL Server 2000)من خلال شبكة من نوع LAN

List of Abbreviations

CS	**Communication System**
DAO	**Data Access Object**
DB	**Database**
DBA	**Database Administrator**
DBTG	**Database Task Group**
DBMS	**Database Management System**
DC	**Data Communication**
DD	**Data Dictionary**
DDB	**Distributed Database**
DDBMS	**Distributed Database Management System**
DDBS	**Distributed Database System**
DDL	**Data Description Language**
DML	**Data Manipulation Language**
GTM	**Global Transaction Manage**
HDDBS	**Heterogeneous Distributed Database System**
LAN	**Local Area Network**
LTM	**Local Transaction Manage**
MDBS	**Multi-Database System**
MDBMS	**Multi-Database Management System**
RDF	**Resource Description Framework**
WAN	**Wide Area Network**
WWW	**World Wide Web**

List of Figures

List of Tables

Chapter One

Overview

1.1 Introduction

Various types of database systems are currently in use today these information resources could be immediately made available for many users through existing computer systems. However since these systems often use different data models and different query languages users of one system cannot easily access the data stored in other systems. Thus there is a growing need for tools to maximize the reusability and interoperability of arbitrary computing services while keeping the autonomy of pre-existing databases in federated approach. One way of achieving interoperability among heterogeneous and autonomous DataBase Management Systems (DBMS) is to develop a multi database system which supports a single common data model and a single query language on top of different types of existing systems. The global schema of a multi database system is the result of the integration of schema exported from the underlying databases (i.e. local databases).

A global query language is made available to the users of them multi database system to specify the queries against the global schema which are called global queries [Aru 97].

1.2 Database System

The word database *(DB)* is in such common use that must begin by defining a database:

A ***database (DB)*** is a collection of interrelated data, stored together without harmful of unnecessary redundancy to serve multiple application. The data is stored such as to be independent from the program which uses it. [Aru 97]

A database represents some aspects of the real world, sometimes called miniworld. Changes to the miniworld are reflected in the database.

Recently a database system in a large organization consists of *hardware*, *software*, *data*, and *people*. The hardware configuration is comprised of one or more computers, disk drive, terminal, printer, and other physical devices. The software includes a Database Management System (***DBMS***) and application program which uses the DBMS to access and manipulates the database. The data, representing the recorded facts important to the organization, resides physically on disk but logically structured in a way to make its access easy and efficient. People, both database system users and applicationers, work together to define the characteristics and structure of the database system to create the application programs which will provide the information essential to the company's success[Gar 92].

Database system eliminates problems with data redundancy on data control. Data is control via a *data dictionary* system which is itself controlled by a group of company employees known as *Database Administrator* (***DBA***), who has this control responsibility and might consider him as part of the database system. All of these features database systems simplify the effort and reduce program maintenance.

Let us now consider the advantages and the disadvantage of database system [Gar 92]:

1. The ability to operate in different data structures.
2. Redundancy can be reduced.
3. Independent of data from the medium on which it stored.
4. High speed of retrieval and fast online use.
5. Inconsistency can be avoided.
6. Stored data can be shared.
7. Standards can be enforced.
8. Minimum costs, by minimizing the required total storage.
9. Data integrity can be maintained.
10. Security restrictions can be applied.
11. Conflicting requirements can be balanced.

While the disadvantages

1. Database systems are complex difficult and time consuming to design
2. Substantial hardware and software start-up costs
3. Damage to database affects virtually all applications programs
4. Initial training required for all programmers and users

1.3 Distributed Database System

The availability of database and of computer network rises to a new field "Distributed database system" which is an integrated database built on top of computer network rather than on a single computers. The data which constitute the databases are stored at the different sites of computer networks, and the application programs which are run by the computers access data at different sites. Therefore, distributed database is a collection of data which belong logically to the same system but spread over the sites of a computer network [Ozs 06].

1.4 Types of Distributed Database Systems

The term distributed database management system can describe various systems that differ from one another in many aspects. The main thing that all such systems have in common is the fact that data and software are distributed over multiple sites connected by some form of communication network. The first factor we consider is the degree of homogeneity of the DDBMS software.

If all servers (or individual local DBMSs) use identical software and all users (clients) use identical software the DDBMS is called homogenous otherwise it is called heterogeneous. Another factor related to the degree of homogeneity is the degree of local autonomy. If there is no provision for the local site to function as stand-alone DBMS then the system has no local autonomy. On the other hand if direct access by local transactions to a server is permitted the system has some degree of local autonomy [Ozs 06].

1.5 Integration of DataBase

In today's enterprises information is typically distributed among multiple heterogeneous database management systems. The heterogeneity exists at three basic levels. The first is the platform level. Database system resides on different hardware, use different operation systems and communicate with other systems using different communications protocols. The second level of heterogeneity is the database management system level. Data is managed by a variety of database management systems based of different data models and languages (e.g. file systems, relational database systems, object-oriented database systems etc.). Finally the third level of heterogeneity is that of semantics. Since different databases have been designed independently semantic conflicts are likely to be present. This includes schema conflicts and data conflicts.

Commercially available technology offers inadequate support both for integrated access to multiple databases and for integration multiple applications into a comprehensive framework. Some products offer dedicated gateways to other DBMSs with limited capabilities. Thus they require a complete change of the organizational structure of existing database to cope with heterogeneity [Ozs 06].

1.6 Literature Survey

A number of works relating to heterogeneity issues of data, different technologies used in the retrieval of distributed databases.

Some of the research papers, which have been reviewed in this context, are as follow.

Joachim Hammer, Dennis Mcleod," ***An Approach to Resolving Semantic Heterogeneous in a Federation of Autonomous, Heterogeneous Database Systems***", [Joa 96].

An approach to accommodating semantic heterogeneity in a federation of interoperable, autonomous, heterogeneous database is presented. A mechanism is described for identifying and resolving semantic heterogeneity while at the same time honoring the autonomy of the database components that participate in the federation. An approach serves as a basis for the sharing of related concepts through (*partial*) schema unification without the need for a global view of the data that is stored in the different components. The mechanism presented in this paper is can be seen in contrast with more traditional approach such as " integrated database" or "distributed database".

Richard Hull, ***"Managing Semantic Heterogeneity in Databases",*** [Ric 97].

The paper discusses some of the key things regarding relational database but the reading tends to show that relational databases are good at handling

huge volume of data but they are not good when the data is distributed and also they do not have the tendency to understand the meaning of data. The paper also discusses different architectures for database interoperations, which are of very importance
in managing Semantic Heterogeneity in databases. The paper tends to create a theoretical perspective in the heterogeneity issue but does not discuss in detail about how actually data can be integrated in a distributed environment and does not also refer to the technologies, used in this area.

Andrei Lopatenko UM, Anne Asserson, UiB, Keith G Jeffery CLRC,"***Information Retrieval of Research Information in a Distributed Heterogeneous Environment",*** [And 01].

User demands to have access to complete and actual information about research may require integration of data from different CRISs. CRISs are rarely homogenous systems and problems of CRISs integration must be addressed from
technological point of view. Implementation of CRIS providing access to heterogeneous data distributed among a number of CRISs is described. A few technologies – distributed databases, web services, and semantic web are used for distributed CRIS to address different user requirements. Distributed databases serve to implement very efficient integration of homogenous systems, web services - to provide open access to research information, semantic web – to solve problems of integration semantically and structurally heterogeneous data sources and provide intelligent data retrieval interfaces. The problems of data completeness in distributed systems are addressed and CRIS-adequate solution for data completeness is suggested.

Norio Katayama, Masanori Sugimoto, and Jun Adachi, ***"A Universal Query Interface for Heterogeneous Distributed Digital Libraries",*** [Nor 02].

Various kinds of digital libraries are accessible on the WWW (World Wide Web). The interoperability among those systems is one of the major topics of the digital library community. This paper presents a Hy-NeSS model and its query
interface for that purpose. The HyNeSS model integrates the entity relationship model and the predicate logic, and provides a universal and formal expression for network-oriented data. It has developed a prototype system of the HyNeSS database system. The query interface is universal in the sense that it does not depend on any particular database schema. It loads the schema information from a remote HyNeSS database system and adapts itself to the given schema. On the submission of a query, the composed expression is converted into a declarative query expression based on the predicate logic. It provides a universal expression to retrieve and identify objects in heterogeneous distributed environment.

Tapio Niemi, Marko Niinim¨aki, Vesa Sivunen, ***"Integrating Distributed Heterogeneous Databases and Distributed Grid Computing",*** [Tap 03].

The aim of this paper is to present a middleware that combines the flexibility of distributed heterogeneous databases with the performance of local data access. The middleware will support both *XML* and relational database paradigms and applies Grid security techniques. The computing and database access facilities are implemented using Grid and Java technologies.

In this system, data can be accessed in the same way independently of its location, storage system, and even storage format. The system will also support distributed queries and transaction management over heterogeneous databases. This system can be utilized in many applications related to storing, retrieval, and analysis of information. Because of the advanced security components, e-commerce is a potentical application area, too.

*Khurram Allah Ditta,"****Integrating Information from Heterogeneous Distributed Databases"***, [Khu 04].

Heterogeneity issues regarding databases are one of the much acknowledged research areas in recent years. This is because of the growing interest of developers and researchers in the semantic web.

The optimum solution would be to find a technique, which would not only provide data integration among semantically heterogeneous sources but would be designed in such a manner that it could be applied to different applications. Although there are not many applications, which are designed to provide such functionality, still there has been extensive research in this area.

The project has proposed a design to manage semantic the heterogeneity and data integration.

Xuequn. Wu , Schmelzer R. “***A CORBA-Based Architecture for Integrating Distributed and Heterogeneous Databases***”,[Xue 04].

This research discusses one of the key technologies namely CORBA used in distributed computing. The paper gives a lot of technical details about how heterogeneous information sources can be integrated but since the researcher does not know much about CORBA so it is not easy to understand the technology properly, hence it’s convenient to stick to XML based solution to the project.

Some of the other technologies, which have been reviewed as a part of project him research, such as RDF. Resource Description Framework (RDF) is a framework for representing information in the Web. It is a simple data model which is based on XML syntax. RDF provides a unique way to interconnect applications, these application cab be databases and can also be used to combine data from several applications to arrive at new information. Like HTML is said to be simple and can be displayed anywhere, for RDF it can be said that it could be understood everywhere.

1.7 The Aim of thesis

The aim of this thesis is to construct integrated multi-DataBase information system by merge and manage three heterogeneous databases sources (Access,Foxpro,SQl). Besides looking into the issue of this project which aims at:

1. Developing and understanding of the how to query heterogeneous data sources implemented in a distributed environment.
2. Studying the issues of data synchronization among the distributed databases.
3. Investigating issues regarding the schema integration of heterogeneous sources.

Chapter Two

Distributed Database System

And

Client Server Architecture

2.1 Introduction

Distributed databases are important for economical, organization and technological reasons. They can implement in large geographical computer networks and in small local network. Distributed database systems eliminate many of the shortcomings of centralized database and fit more naturally in the decentralized structure of many organizations.

The amount of information stored in different databases is increasing rapidly all the time. Many of these databases are accessed from the Internet or intranets. Even the whole Internet can be seen as a huge distributed heterogeneous database and it is often used as such even by ordinary end users.

Thus, the principle motivating factors for implementing distributed database systems include increased reliability; improve response performance, location and replication independences, integration of heterogeneous DBMSs, and organizational congruence.

2.2 Distributed Database Management System (DDBMS)

A distributed database management system *(*DDBMS*)* is a software system that permits the management of the distributed database and makes the distribution transparent to the users. Distribution Transparency means that, the independence of the application program from the distribution of data, and has been considered to be conceptually equivalent to data independences in centralized DB. DDBMS supports the creation and maintenance of DDBs.

The software components which are typically necessary for building a DDB in this case are [Ste 85]:

1. The Database Management Component (DB).

2. The Data Communication component (DC).

3. The Data Dictionary (DD), which is extended to represent information about the distribution of data in the network.

4. The Distributed Database Component (DDB).

5. the End user, i.e. terminals (T).

These components are connected as shown in Figure (2.1) for two–site network.

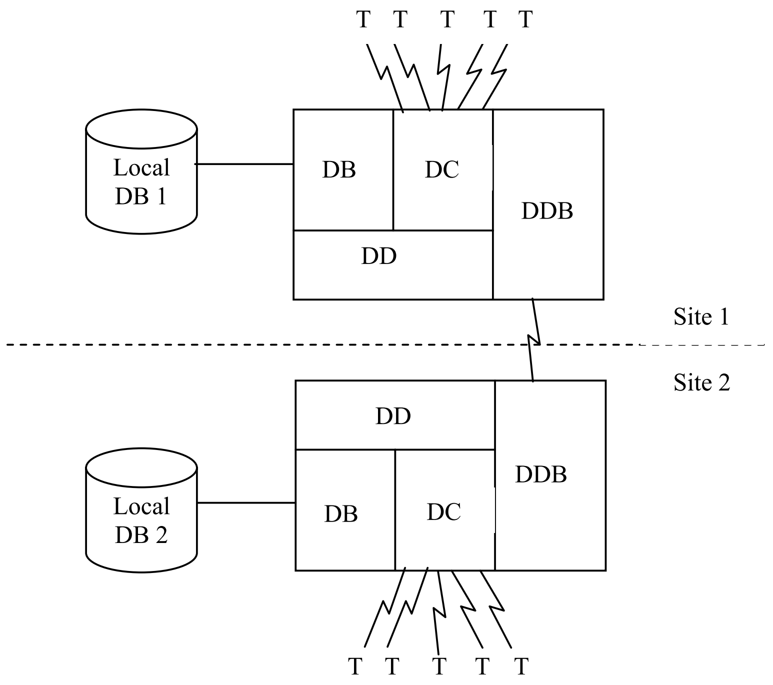


Figure (2.1): Components of DDBMS[Ste 85]

2.2.2 Functions of DDBMS

It expected that DDBMS has at least the functionality of a DDBMS. It also has following functionality [Ami 04]:

1- Extended Communication Services.

2- Extended Data Dictionary.

3- Distributed Query Processing.

4- Extended Concurrency Control.

5- Extended Recovery Services.

2.2.3 DDBMS Implementation Alternatives

Basically, there are dimensions for database, DDBMSs and application. Figure (2.2) shows the alternatives in implementing DDBMSs [Zho 01]:

1. Distribution: means that database, DDBMSs and applications can be located centrally on the same machine or distributed in a network.
2. Autonomy: means that the control of database, DDBMSs and application can be done centrally, autonomous, or in between., its may be on Various versions:
    a. Design autonomy: Ability of a component DBMS to decide on issues related to its own design.
    b. Communication autonomy: Ability of a component DBMS to decide whether and how to communicate with other DBMSs.
    c. Execution autonomy: Ability of a component DBMS to execute local operations in any manner it wants to.
3. Heterogeneity: means that database, DDBMs and application can be of the same type or of different types.

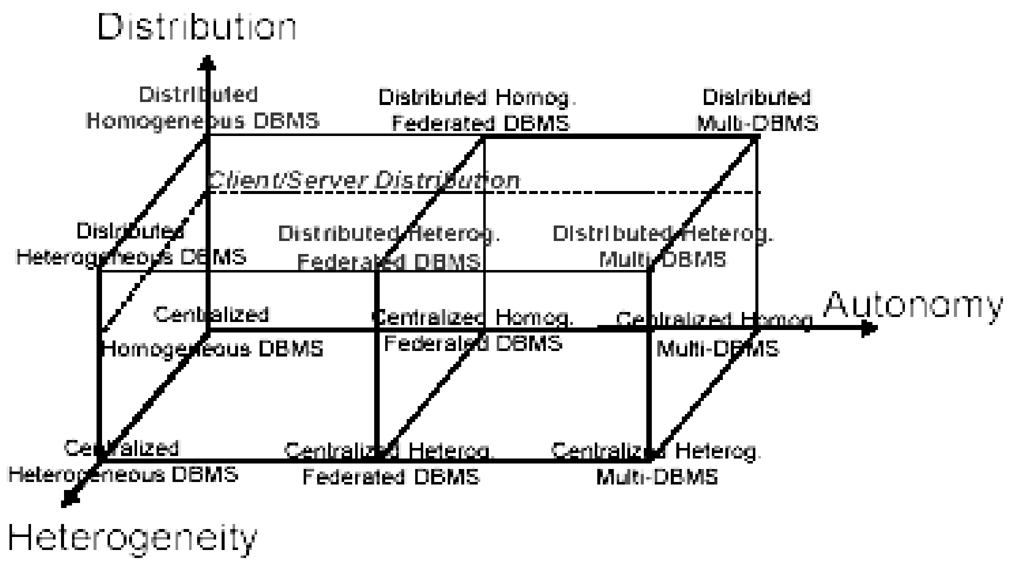

Distribution
Distributed Homogeneous DBMS
Distributed Homog. Federated DBMS
Distributed Multi-DBMS
Client/Server Distribution
Distributed Heterogeneous DBMS
Distributed Heterog. Federated DBMS
Distributed Heterog. Multi-DBMS
Autonomy
Centralized Homogeneous DBMS
Centralized Homog. Federated DBMS
Centralized Homog. Multi-DBMS
Centralized Heterogeneous DBMS
Centralized Heterog. Federated DBMS
Centralized Heterog. Multi-DBMS
Heterogeneity

Figure (2.2): DDBMS Implementation alternatives

2.3 Classification of Database Managements Systems

Several criteria are normally used to classify DBMDSs. The first is the data model on which the DBMS is based [Ozs 06]. The main data model used in many current commercial DBMSs is the relational data model. The object data model was implemented is some commercial systems but has not had widespread use. Many legacy (order) applications still run on database systems based on the hierarchical
and network data models .The relational DBMSs are evolving continuously and in particular have been incorporating many of the concepts that were developed object databases. This has led to a new class of DBMSs called object –relational DBMSs. Can hence categorize DBMSs based on the data model: relational object relational hierarchical network and other. The second criterion used to classify DBMSs is the number of users supported by the system. Single –user systems support only one user at a time and are mostly used with personal

computers. Multiuser systems which include the majority of DBMSs support multiple users concurrently. A third criterion is the number of sites over which the database is distributed. A DBMS is centralized if the data is stored at a single computer site. A centralized DBMS can support multiple users but the DBMS and the database themselves reside totally at a single computer site. A distributed DBMS (DDBMS) can have the actual database and DBMS software distributed over many sites connected by a computer network. Homogenous DDBMSs use the same DBMS software at multiple sites. A recent trend is to develop software to access several autonomous preexisting databases stored under heterogeneous DBMSs. This leads to a federated DBMS (or multi database system) in which the participating DBMSs are loosely coupled and have a degree of local autonomy. Many DDBMSs use a client –Server architecture.

2.4 Paradigms for Data Distribution

The important paradigms of data distribution are:-

1. Client-server architecture: separation of the database server from the client
2. Distributed databases: several database servers used by the same application
3. Parallel databases: several data storage devices and processors operate in parallel for increasing performances
4. Multi databases: is the software that provides for the management of collection of autonomous sites and access to them
5. Data warehouses: servers specialized for the management of data dedicated

2.5 Distributed Database System (DDBS)

The distribution of the data is transparent to the users who can access the data as if they were located at one site. In reality, to access data distributed in different computer sites, the transmission of data over communication links is needed. Since communication delay is substantial, an efficient query processing mechanism has to be designed [Sey 98] .

There are two equally important aspects of the distributed database:

A. Distribution: The fact that the data are not resident at the same site (processor).

B. Logical Correlation: The fact that the data have some properties which tie them together, so that we can distinguish a distributed database from a set of local databases or files which are resident at difference sites of a computer network.

There exist two types of application that are used by distributed database. These are:

A. Global Application (or Distributed Application)*:* That accesses data at the discriminating characteristics of distributed database with respect to a set of local database.

B. Local Application: Those which can access data resident at the same site or computer that this application is working in [Ste 85].

With these two types of application, can give a complete definition of the distributed database systems ***(***DDBS***)*** which is a collection of data which are distributed over different computers of a computers network. Each site of the network has autonomous processing capability and can perform local applications. Each site also participates in the execution of at least one global application, which requires accessing data at several sites using a communication subsystem. As shown in Figure (2.3). The key here is integration, not centralization because the distributed database system attempt to achieve integration without centralization.

The sites in the distributed system communicate through different types of communication links, such as telephone lines, coaxial cables, or satellite links. Computer networks are usually classified as Local Area Network (LAN) or Wide Area Network (WAN) depending upon the distance between the sites. Usually WAN links are telephone lines, satellite channels, or microwave links [Rob 88].

The proximity (less than a mile) of workstations or same building in a LAN makes it possible to use coaxial cable and fiber optic links, in addition to twisted wire pairs. Typical LAN link are higher speed and subject to lower error rates than WAN links.

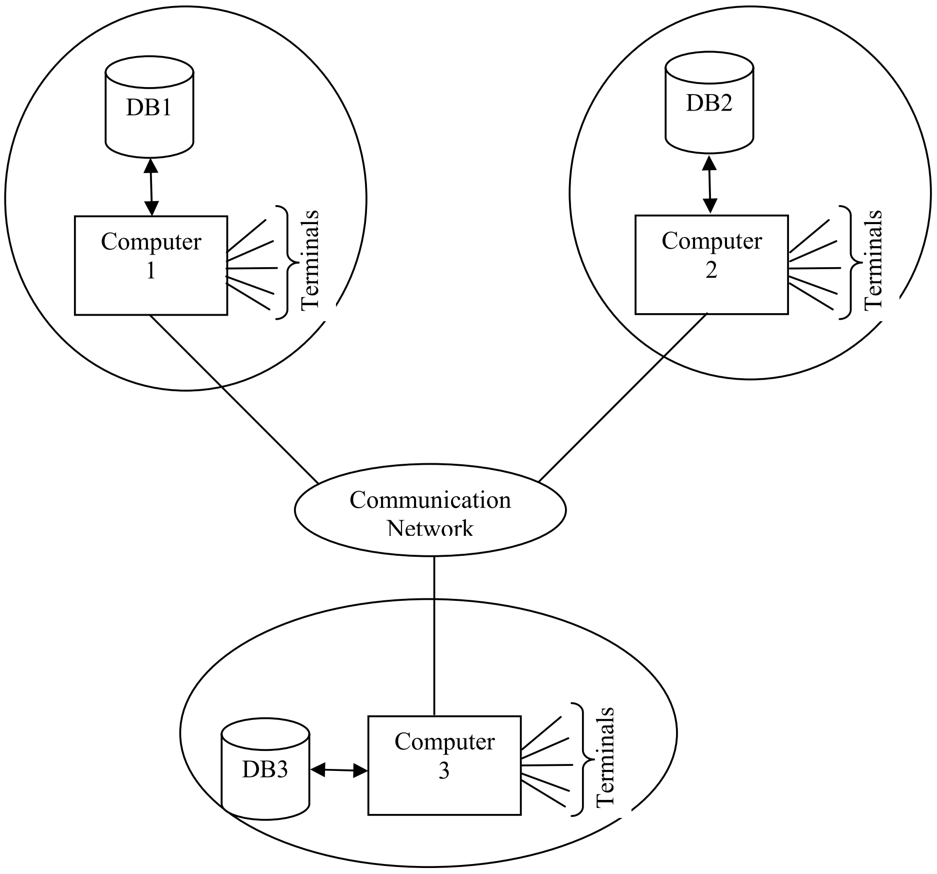


Figure (2.3): Distributed Database System [Rob 88]

2.5.1 The Advantages and Disadvantages of DDBS Usage

Distributed database systems can be complex, but they offer capabilities that extend the advantages of database technology.

I. Advantages of DDBS

A list of the main advantages of DDBS can be presented of the following: [Ram 00],[Nin 03]and[Ste 85].

1. Organizational and Economic Reasons

Many organizations and enterprises are decentralized, because it's often have branches or division in different locations and distributed approach fits more naturally the structure of organization.

2. Incremental Growth and Easier System Expansion

Distributed system can growth better no distributed systems. So that, if an organization grows by adding new node (new branch, new ware houses ,etc), then the DDB approach supports a smooth incremental growth with a minimum degree of impact on the already existing nodes, there is no need to re-configure the whole database ,the new node almost automatically becomes part of the global database .

However, expansion of the system is much easier, because of the ability to add a new site by data and users over time without major restructuring.

3. Reduce Communication Overhead

Many application are local clearly reduces the communication overhead with respect to a centralized DB. Therefore, the maximization of locality of applications is one of the primary objectives in DDBS design.

4. Interconnection of Existing Database

DDBSs are the natural solution when several databases already exist in an organization and the necessity of performing global applications arises. In this case, the DB is created Bottom-up from the preexisting database.

5. Improved Performance

Since each site handles only a portion of a DB (Data Localization), the contention for CPU and I/O resources is reduced, and simultaneously reduces access delays involved in Wide –Area Network .When a large DB distributed over multiple sites, smaller DB exist at each site .As result, local queries and transactions accessing data a single site have better performance because of the smaller local DB. Moreover, the speed of query processing is another contributes to improved performance, it means that, if a query involves data at several sites, it may be possible to split the query into sub queries that can be executed in parallel.

6. Data Sharing

The major advantages in building a DDBS is the prevision of an environment where users at one site may be able to access the data residing at other sites .For instance, in a distributed banking system, where each branch stores data related to that branch, it is possible for a user in one branch to access data in another branch. Without this capability a user wishing to transfer funds from one branch to another would have to resort to some external mechanism that would couple existing systems.

7. Increased Reliability and Availability

These are two of the most common potential advantages cited for DDBS. Reliability is broadly defined as the probability that the system is running (not down) at a certain time point, whereas availability is the probability that the system is continuously available during the time interval. DDB can upgrade reliability. If one site computer fails, or if a communication link goes down, the rest of the network can probably continue functioning .Moreover, when data is replicated as
two or more sites; the required data may still be available from a site that is still operable. This facility is crucial for DDBS used for real time applications.

8. Transparency of Distribution

Transparency refers to separation of the higher level sometimes of a system from lower level implementation details. It means that the user and user programs should not need to know where (at which site) any particular item of data is located. This advantage simplifies the logic of application programs, and it allows data to be moved from one site to another as usage patterns change without necessitating any reprogramming.

9. Local Autonomy and Local Control

The possibility of local autonomy is often a major advantage of DDBS, it allows each site to store and mention its own database allows immediate and efficient access to data that is used most frequently. Such data may be used at other sites, as well, but usually with less frequency. Similarly, data stored at other locations are usually needed less often and can be accessed as required.

10. Security

The DDB allowing local control over the data used most frequently at a site can improve user satisfactions with the data system, since local databases can more nearly reflect an organizations administrative structure and thereby better service its manager's needs**.**

In a centralized system, the DBA of control site controls the database. In DDBS, there is a global DBA responsible delegated to the local DBA for each site.

11. Efficiency and Flexibility

Efficiency and flexibility are most basic concepts of the DDB. It makes the data very close to the user, in order to make the communication to data fast. If any user needs unavailable data in his site, the system will bring him the required data unconsistoustly.

II. Disadvantages of DDBS

The primary disadvantage of DDBS is added complex required to ensure proper coordination among the sites. The main disadvantages of DDBS are as follows [Ste 85],[Sil 02],[Kam 01]:

1. Complexity of Design and Implementation.

This approach is a complex task to build a DDB on top of computer network and a set of local DBMSs at each site; it would be an infeasible effort without these building blocks. Thus, DDBS problems are inherently more complex than those of the centralized DBMS.

2. Software Development Cost

It is more difficult to implement a DDBS, thus, it is very costly (more: *H/W, S/W*, and *people* costs).

3. Greater Potential for Bugs

Since the sites that constitute the DDBS operate in parallel, it is harder to ensure the correctness of algorithm. The potential exists for extremely subtle bugs.

4. Increased Processing Over Head

The exchange of message and the additional computation required to achieve inter site coordination are a form of overhead that does not arise in the centralized system.

5. Difficult to Convert

No tools to convert centralized DBMSs to DDBSs.

There are additional disadvantages of DDBS, for example when replicated data is mentioned at several sites, which require extra resources, and synchronization, and coordination to ensure that concurrent updates are consistent, and transaction processing and recovery become more difficult, that make data available to users throughout a network makes security inherently more complex for a DDB than with centralized counterpart.

Although these limitations suggest caution in planning and implementing DDBSs, the technology for mitigating such limitations is rapidly improving. In order for firms to be responsive, productive, and competitive, DDBSs will increasingly be selected as the central component of an effective information system strategy.

2.6 Distributed Database System Architecture

Architecturally, a distributed database system consists of a (possibly empty) set of query sites and a non-empty set of data sites. The data sites have data storage capability while the query sites do not. The latter only run the user interface (in addition to applications) in order to facilitate data access at data sites.

There are two points of view for the DDBMS architecture: The level of distribution transparency and the alternative of Client/Server models, as highlighted in the subsequent sections [Ozs 06].

2.6.1 Level of Distribution Transparency

The reference architecture for a DDBS is defined to represent different level of distributed transparency. Figure (2.4) shows the reference architecture for a DDB [ste 85].

In this architecture, we can see several levels or schemas, these are:

II. Global Schema

At the top level of this architecture, there is the global schema. It is defines all data which are contained in the DDB as if the DB were not distributed at all. For this reason, the global schema can be define exactly in the same way as in a non distributed DB .However, the data module used for the definition of global schema should be convenient for the definition of the mapping to the other levels of the DDB.

Thus, using relational module will generate a set of global relations, each one can be split into several non overlapping portion called fragments.

II. Fragmentation Schema

defines the mapping between relations and fragments .This mapping is one to many such that several fragments correspond to one global relation ,but only global relation correspond to one fragment .Fragments are logical portions of global relations, which are physically located at one or several sites of the network.

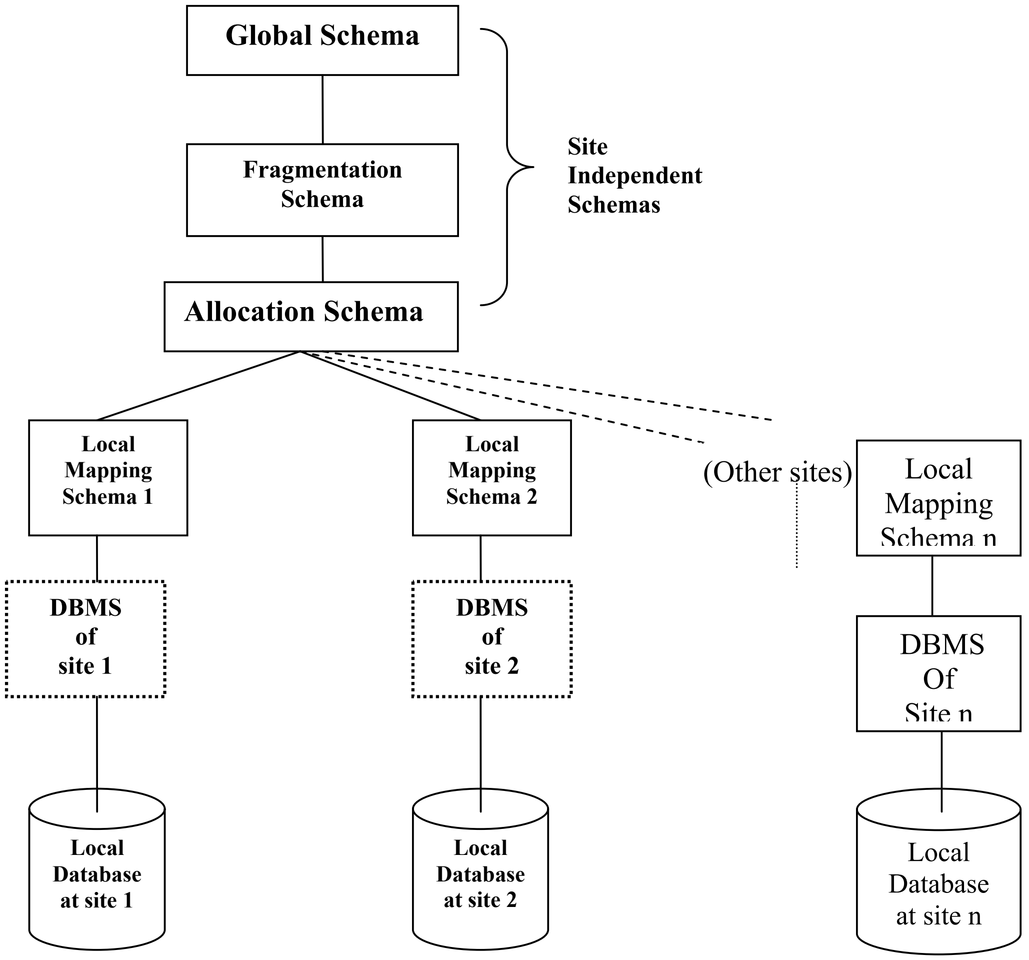


Figure (2.4): Reference Architecture for Distributed Databases.

III. Allocation Schema

Defines at which site (sites) a fragment is located .All the fragments which correspond to that same global relation are located at the same site. Constitute the physical image of the global relation.

IV. Local Mapping Schema

At a lower level of this architecture is local mapping schema which defines the map between the physical images and the object which are manipulated by the DBMSs. This mapping depends on the type of local DBMS.

There are three important objectives which motivate the feature of the architecture of DDB, these are:

1-Sperating the Concept of Data Allocation. This separation allow us to distinguish two different levels of distribution transparency, these are "Fragmentation Transparency" and" Location Transparency", fragmentation transparency is the highest degree of transparency and means that the user or application programmer works on global relation. While, location transparency is a lower degree of transparency and means that the user or application programmer works on fragments instead of global relations, however, he or she does not know where the fragments are located.

2-Exiplicit Control of Redundancy. The reference architecture explicit control of redundancy at the fragment level.

3-Independence from Local DBMSs. Allows us to study several problems without having into consideration the data model local DBMSs. This feature may be called local mapping transparency.

2.6.2 The Alternative of Client/Server Models

There are a number of different architectural models for the development of a distributed DBMS, ranging from client/server systems [Orf 96]; where, Client-server is a general model of interaction between software processes, where interacting processes are sub-divided among *clients* (which require services) and *servers* (which offer services), that means, the query sites correspond to clients while the data sites correspond to servers, to a peer-to-peer system where no distinction is made among the client machines and the server machines.

These architectures differ with respect to the location where each DBMS function is provided. In the case of client/server DBMSs, the server does most of the data management work. This means that all of the query processing and optimization, transaction management and storage management are done at the server. The client, in addition to the application and the user interface, has a *DBMS client* module that is responsible for managing the data that is cached to the client and (sometimes) managing the transaction locks that may have been cached as well. Figure (2.5) shows a typical client/server functional distribution.

Normally, a client process requests few services in sequence from one or more server processes, while a process server responds to multiple requests from many process clients [Pao 99].

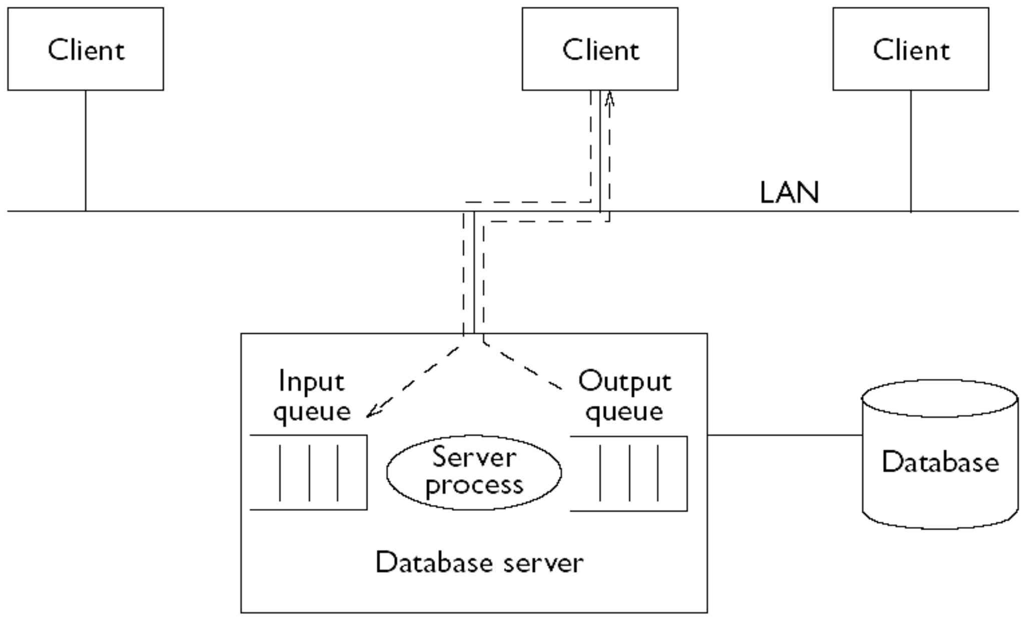
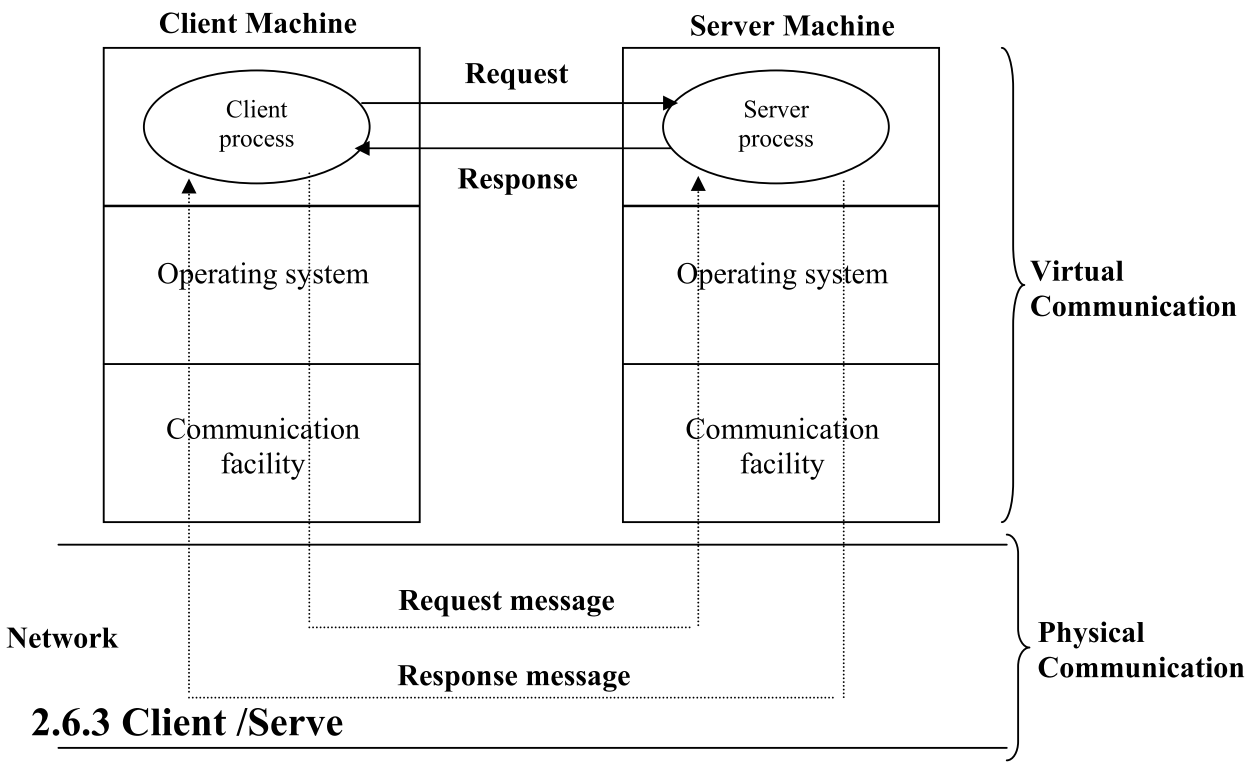


Figure (2.5): Client - Server Model

The simplest client/server architecture is a multiple-client/single-server system. From a data management perspective, this is not much different from centralized databases since database is stored on only one machine (the Server) which also hosts the software to manage it. However, there are some important differences from centralized systems in the way transactions are executed and caches are managed. More sophisticated client/server architecture is one where there are multiple serves in the system (the so-called multiple-client/multiple-

server approach). In this case, two alternative management strategies are possible:
either each DBMS client manages its own connection to the appropriate server or each client knows only its home server which then communicates with other servers as required. The former approach simplifies server code, but a load the client machines with additional responsibilities (heavy client) while the latter approach concentrates data management functionality at the servers and provides transparency of data access at the server interface. A typical client/server architecture is given in Figure (2.6) [Ozs 06].



2.6.3 Client /Serve

Figure (2.6): Client - Server Architecture

First discuss client /server architecture in general then see how it is applied to DBMSs. The client /server architecture was developed to deal with computing environments in which a large number of PCs workstations file server printer database derivers Web servers and other equipment are connected via a network. The idea is to define specialized servers with specific functionalities. For example it is possible to connect a number of PCs or small workstations as clients for a file server that maintains the files other client machines. Another machine could be designated as a printer server by being connected to various printers thereafter all print requests by the clients are forwarded to this machine. Web servers or E-mail servers also fall into the specialized server category. In this way the resources provided by specialized servers can be accessed by many client machines. The client machines provide the user with the appropriate interfaces to utilize these servers as well as with local processing power to run local applications. This concept can be carried over to software with specialized software-such as a DBMS or a CAD (Computer –Aided Design) package –being stored on specific server machines and being made accessible to multiple clients [Ozs 06].

Figure (2.7) illustrates client /server architecture at the logical level

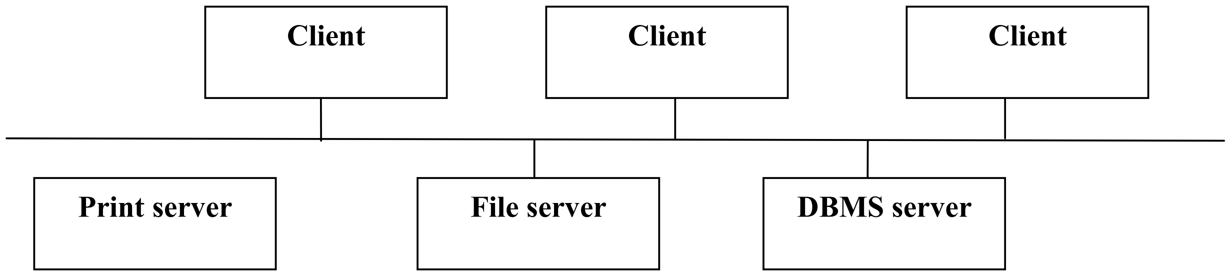


Figure (2.7): logical two –tier client /server architecture

II. Advantages and disadvantage of Client-Server Architectures

In data management, allocation of client and server processes to distinct computers is now widespread, because [Ham 80]:

1. More efficient division of labor
2. Horizontal and vertical scaling of resources
3. Better price/performance on client machines

4. Ability to use familiar tools on client machines
5. Client access to remote data (via standards)
6. Full DBMS functionality provided to client workstations
7. Overall better system price/performance
8. The functions of client and server are well identified
9. They give rise to a convenient separation of design and management activities
10. SQL offers an ideal programming paradigm for the identification of the 'service interface', which lists the services offered by the server

While the disadvantages is

a. Dependability - when the server goes down, operations cease
b. Lack of mature tools - it is a relatively new technology and needed tools are lacking e.g.. Automated client software distribution
c. Lack of scalability - network operating systems (e.g.. Novell Netware, Windows NT Server) are not very scalable.
b. Higher than anticipated costs
e. Can cause network congestion

2.7 Distributed Database System Design

The designing of DDBS is very difficult, and the problem design is how the DDBMS software and the applications that run against it should be placed across the sites, In the general setting, making decision about the placement of data and programs across the sites of computer network as well as possibly designing the network itself.

The two fundamental design issues are fragmentation (the separation of DB into portions called fragments), and allocation, the optimum distribution of fragments.

The design of centralized database amounts to [Ste 85]:

1- Designing the "conceptual schema", which describes the integrated DB (i.e. all the data which are used by the database applications).

2- Designing the "physical database", this is mapping the conceptual schema to storage area and determining appropriate access methods.

In a DDB these two problems become the design of the global schema and the design of the local physical databases at each site, the techniques which can be applied to these problems are the same as in the centralized databases .The distribution of the DB adds to the above problems two new ones.

3- Designing the Fragmentation, this determines how global relations are subdivided into horizontal, vertical or mixed fragments.

4- Designing the Allocation of fragments, which determines how fragments are mapped to physical images, in this way, also the replication of fragments is determined.

These two problems fully characterize the design of data distribution .Thus the design of DDBS can be a complex task, careful consideration must be given to the objective and strategies to be served by the design.

2.7.1 Objective of DDBS Design

Some of the objectives that are common to most DDBS implementations are discussed in this section.

1- Location Transparency: Location transparency enables a user to access data without knowing, or being concerned with the site at which the data resides. The location of the data hidden from the user.

2- Configuration Independence: Configuration independence enables the organization to add or replace hardware without changing the existing software components of the DDBMS. Its result in a system that is expandable when its current hardware is saturated.

3- Integration of Heterogeneous DBMSs: Its sometimes desirable to integrate DB maintained by different DBMSs are supplied by different computers .Often the DBMS are supplied by different vendors ,one approach to integrating these databases is to provide a single user interface that can be used to access the data maintained by the heterogeneous DBMSs .

4- Processing Locality: Distribution of data to maximize processing locality corresponds to the simple principle of placing data as close as possible to the applications which use them.

5- Availability and Reliability of DDB: A high degree of availability for applications is achieved by storing multiple copies of the same information ,the system must be able to switch to an alternative copy when the one that should be accessed under normal conditions is not available .Reliability is also achieved by storing multiple copies of same information ,since it is possible to recover from crashes or from the physical destruction (such as, fire, earthquake) of one of the copies by using the other ,still available copies .

6- Workload Distribution: Workload distribution is done in order to take advantages of the different power or utilizations of computers at each site, and to maximize the degree of parallelism of execution of applications.

7- Storage Cost and Availability*:* Database distribution should reflect the cost and availability of storage at the different sites. It is possible to have specialized sites in the network for data storage, or conversely to have sites which do not support mass storage at all.

2.7.2 Strategies of DDBS

I. Data Fragmentation

The decomposition of global relation into several fragments, each fragment is stored in distinct sites .However, if the relation R is fragmented, R is divided into a number of fragments R_1, R_2, R_3…R_n. These fragments contain sufficient information to reconstruct the original relation R; this reconstruction can take place through the application of either the union operation or special type of join operation on the various fragments. To illustrate these ways clearly, consider we have the following relation.

Deposit-schema= (Visa Card branch, account-id, Person- name, Account).

The relation deposit (Deposit-Schema) is shown in Table (2.1).

Table (2.1): Sample Deposit Relation

Visa Card Branch	**Account-id.**	**Person-name**	**Account**
Baghdad	305	Ahmed	500
Baghdad	226	Ali	336
Mousel	177	Ali	205
Mousel	402	Rashid	10000
Baghdad	155	Rashid	62
Mousel	408	Rashid	1123
Mousel	639	Majid	750

The main advantages and disadvantages of data fragmentation are:

1. Usage: Applications work with views rather than entire relation.
2. Efficiency: Data is stored does to where it is most frequently used, and data that is not needed by local applications is not stored at that location.
3. Parallelism: With fragments as unit of distribution transaction can be divided into several sub queries that operate on fragments.

4. Security: Data not required by local application is not stored at that location and so not available to unauthorized users.

The various types for data fragmentation, can be disscussed [Hen 91]:

1. Horizontal Fragmentation

Horizontal fragmentation consists of partitioning the tuples of a global relation R into subsets R1,R2,R3,…,Rn ,each tuple of R must belong to one of the fragments ,so that the original relation can be reconstructed if needed .It can be defined by expressing each fragment as selection operation on the global relation R ,that is ,a predicate Pi is used to construct fragment R, as following format ***Ri = Q Pi(R)…..(2.1)***

The reconstruction of the relation R can be obtained by taking the union of all fragments, that is

R=R1 U R2 U R3 …U Rn or (R= Ui =1 to n Ri)…..(2.2)

To illustrate that if the banking system has only two branches: *Baghdad* and *Mosul*, then there are two different fragments,

Deposit1 = Qbranch-name="Baghdad" (Deposit)….(2.3)

Deposit2 = Qbranch-name="Mosul" (Deposit)….(2.4)

These two fragments are shown in Table (2.2).

Table (2.2): Horizontal fragmentations Tables

Visa-Branch	Account-id.	Person-name	Account
Mousel	177	Ali	205
Mousel	402	Rashid	10000
Mousel	408	Rashid	1123
Mousel	639	Majid	750

Visa-Branch	Account-id.	person-name	Account
Baghdad	305	Ahmed	500
Baghdad	226	Ali	336
Baghdad	155	Rashid	62

Fragment Deposit1 is stored in Baghdad site, and fragment Deposit2 is stored in Mosul site.

II. Data Replication

Data replication occurs if the system maintains several identical copies of a relation, R, with each copy being stored at a different site [Hen 91].

The advantages of replication then, are that

1. Improvement in the availability : when a copy is unavailable due to site failure , it should be possible to access another copy
2. Improvement in the performance: applications can operate on local replicas instead of having to communicate with remote sites, thus, reduced data transfer, and queries on R may be processed by several sites in parallel (concurrently).

But its disadvantages are:

The tread of is in the extra cost of the added storage and in the maintenance of mutual consistency among copies. The update of local copy imposes the added overhead of transmitting that update to all sites maintaining a copy of the data, thus it is increased complexity of concurrency control.

III. Replication Transparency

Replication transparency means that the user is unaware of which copy of data is being used. The need for replication transparency occurs when more than one copy of the data exist, since one copy must be updated when changes are made. This process can be a burden on users. Therefore, a DDBMS should handle all such requirements, freeing the user to concentrate on information needs.

IV. Allocation

Allocation means that each fragment is stored at site with optimal distribution. There are four alternatives regarding the placement of data:

a- Centralized: consists of a single database and DBMS stored at one site. With user distributed across the network.

b- Fragmented: database partitioned into disjoined fragment, each fragment is assigned to one site.

c- Full Replication: consists of maintaining complete copy of database at each site.

d- Selective Replication: is a combination of fragmented, replication, and centralized [Ram 00], [Nin 03].

2.8 Type of Distributed Database Systems

The increase in data volumes led to the categorization of DBs in two broad groups, homogeneous and Heterogeneous. The comparison is based on different levels in a distributed database: The hardware, operating system, and the local DBMSs. However, an important distinction for is at the level of local DBMSs, because differences at lower levels are managed by the communication software [Ozs 06].

2.8.1 Homogeneous Distributed Database System (DDBMS)

Homogeneous DDBS refers to a DDBMS with the same DBMS at each site that means, every site runs the same type of DBMS. In homogeneous DBMS, all sites have identical software and hardware (same computer and software), and appears to the user a single system.

2.8.2 Heterogeneous Distributed Database System (HDDBS)

Heterogeneous (HDDBS) refers to different sites run different DMBS. That mean, different site may use different schema and software. Difference in schema is a major problem for query processing, and the difference in software is a major problem for transaction processing. However, in a heterogeneous distributed database the sites may not be aware of each other and may provide only limited facilities for cooperate in transaction processing. Therefore Heterogeneous DDBMSs add the problem of translating between the different data models of the different local DBMSs to the complexity of homogeneous DBMSs [Ozs 06].

2.9 Characteristics of a HDDBS

A DDBMS relies on several database managed by several systems running on several machines [Ber 89]. In most of the systems implemented up to now, all components are homogeneous, that is to say that all the functions are realized in the same way on all the sites of the DDB. As soon as one of the functions is assumed in the whole DDBMS by components of different types, can said the system is heterogeneous for this function. A new problem is now merging: how to build heterogeneous DDBMSs. A heterogeneous DDBMS uses instead at least two different DBMSs.

Thus, many components can be heterogeneous in DDBMS, which means, they are different in [Ber 89]:

1. The hardware, computers.
2. The operating system, on which the DBMS are running.
3. The network that interconnect the different sites (the system can use several local networks connected on a WAN network).
4. The DBMSs, and each function they ensure: Protection, synchronization, resources allocation, transaction management… etc.

5. The data model (relational, network, hierarchical) at the three levels: external, conceptual, and internal.
6. Programming language, the data description languages (DDL) and the data manipulation languages (DML).
7. Implementation may be implemented by different developers.

2.10 The Aims of use Heterogeneous DDBS

Various heterogeneous systems can be imagined different computers with different operating systems running different DBMSs on identical computers, etc.

In practice, a heterogeneous system will be useful, if it proposes the following aims and facilities: [Ozs 06]

1. To allow the users to manipulate a database without having to worry about the distribution of data and the diversity of local systems.
2. To provide the facility to the entire users to manipulate the distributed database like a unique database with different languages they use to practice. So, each language can be more adopted for each usage.
3. To make possible the integration of an existing database in a DBMS without any reorganization of this new local database and without any modification in applications (Keeping data and Languages).
4. HDDBS allows a uniform view on the combination of data maintained by different autonomous database systems.
5. Capability of providing a view of the system which is transparent not only to data distribution, but also to heterogeneity of DBMSs, this is a very difficult goal.

2.11 Problems of Heterogeneous DDBS

In this section, summarize some of the problems of heterogeneous systems; [Ste 85]:

2.11.1 Selection of a Common Model and Data Manipulation Language

The most convenient way for allowing the communication of several heterogeneous DBMSs is by using a common data model and *data manipulation* language(DML); data representation and DML Primitive of each DBMS are mapped to equivalent , representations and primitive of common data model and DML . One advantage of having a common data model is the possibility of using it to describe the global schema of it reference architecture. Thus, also in a heterogeneous environment view of the DB. Another relevant advantage is that
mappings and translations need not be done between each pair of DBMSs. Wherever, the mappings and translations need to be done only between each DBMS and the common data model and DML, which requires a number of translators growing linearly with the number of DBMSs.

Thus the first and more general problem in the development of a heterogeneous system is to select a appropriate common data model and DML. The selected data model and DML should have the following properties:

1- They should allow a simple translation from the data models and DML of the DBMSs constituting the heterogeneous system.
2- They should be suited to represent data and processing of DDB conveniently, in particular, the data model should be capable of representing the global fragmentation, and allocation schemata, while the DML should process “set-oriented" primitives (i.e. the possibility of

representing layered schema architectures and set-oriented processing, such as the relational model, and algebra are good conditions).

2.11.2 Translation to the Common Data Model and DML

The translation between different DBMSs is not pearlier to distributed systems, Translation problems arise whenever two DBMSs are interfaced with in the same system, or whenever the DBMS is changed and applications written for the old DBMS are interfaced with the new DBMS (similar problem can also arise when a new version of the DBMS is also introduced) [Ste 85].

In a heterogeneous system on translation is required, distinguish global optimization, in which the distribution of the execution among sites determined from local optimization, in which each site determines the best method for evaluation its partition of execution.

In heterogeneous systems with a common global data model and DML, it appears responsible to perform global optimizations into the common DML. Thus

instead of building several versions of it , which assumes as input the description of the application in the common DML , a disadvantage of this solution is that an application must be translated also when it can be completely executed at its site of origin .

A different solution consists of the application to a local analyzer which is able to decompose it into local and remote portions. Only the remote portion is translated into the common DML .Thus if no remote site is involved in the application, no translation is required.

A simpler resolution consists of limiting the sophistication of the analysis by classifying applications into two sets:

Completely local application (those which do not need any remote resources), and all other applications .The former should use the local DBMS directly , the later should be translated and them optimized .In the absence of the analyzer ,the user should make this selection , and request the services of either the local DBMS or of the heterogeneous DDBMS.

2.11.3 Integration of Different Schemata

Heterogeneous DDBMSs are typically built by aggregating systems, using a Bottom–Up approach. Thus, it is possible that the same facts are already described in the schemata of two different DBMSs , and it is possible that the two descriptions do not agree .Different in data definition of the same facts are called conflicts . Considered the conflicts that can arise between two schemata which use the common data model. Conflicts are classified as follows:

1- Name conflicts, which typically involve homonyms (Different facts decoted with the same name) and synonyms (different names used for the same facts).
2- Structural conflicts, which arise when the same facts are described in two schemata using different elements of the data model. A typical structural conflict, with a common model including entities and attributes, consists of
3- regarding the same fact as an autonomous entity in one schema and as the attribute of different entity in the other schema.

Performing the integration of two or more schemata requires discovering and resolving the conflicts .In heterogeneous systems, solving conflicts by forcing one interpretation is not always request of possible, since typically data definitions of component DBMSs should not be altered.

Thus the integration process could leave the same conflicts unsolved but produce an auxiliary database, to be used during normal operation, typically for name conversions, and also mediating structural and abstraction conflicts.

2.11.4 Query Processing Problems

Most heterogeneous systems allow only distributed retrievals, while update applications must be done under the control each local DBMS, However, the heterogeneous DBs create additional (new query processing problems).

The distribution of a query should now take into account the difference in performance of the various sites. Some of the functions requested by an application either might be more expensive or might not be available at all remote DBMSs, and in this case it becomes necessary to transmit a superset of the data required by the application to the site of origin of the application, where those functions can be performed .Thus, query distribution should try to balance the amount of remote versus local processing very carefully.

2.12 Heterogeneous DDBS Architecture

Schema integration is a way of integrating the schemas of databases under consideration, or databases, which might be included in future. Schema integration has more relevance to the project as compared to the data warehouse concept. Schema integration results in a global schema, which is also sometimes regarded as the virtual view of the databases.

There is proposed architecture, proposed a four-layered architecture for schema integration but this architecture provides data integration from the structural point of view. The issues regarding the semantics of data were not resolved by the proposed architecture. The following Figure (2.8) shows the four layers architecture [Red 94].

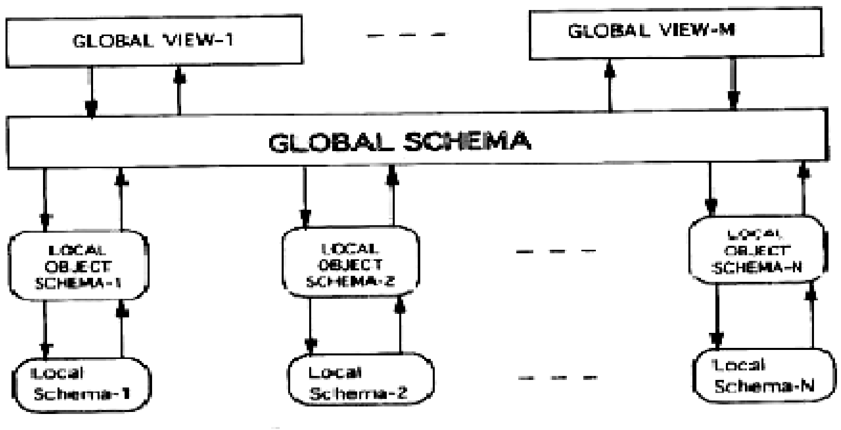


Figure (2.8): Four-Layered Architecture of HDDBS

A number of problems arise when different local schemas are put together. Because of the heterogeneous nature of the data models, each model tends to have different rules. Before the local schemas can be integrated, they need to be arranged in a uniform manner. Inconsistencies arising from issues like homonyms, synonyms, missing data and scaling conflicts, need to be handled at some intermediate level before actual integration can take place at the global level. This is done by defining the *local schema object*s; these objects are defined by the Database Administrators (DBA) who have the knowledge about the kind of information, which is required for integration. In above approach local schema
objects do introduce inconsistencies with respect to human interpretation of the data. It is a fact that the same information might be regarded useful by one person and not to be useful by another person.

Thus, a HDDBS is defined as, comprises a software layer (integration layer) and multiple databases and/or file systems to be integrated. Users can transparently access the integrated databases and/or file systems via the interface provided by the integration layer. The integrated, local databases are autonomous and can also be used as stand-alone systems. Local applications are unchanged and unknown to the HDDBS [Red 94].

There are important concepts in the *Integration Layer:*

1. Global data model
2. Global schema and meta data management
3. Distributed query processing and optimization
4. Distributed transaction management

The final representation of the architecture of Heterogeneous DDBS is shown in Figure (2.9) [Den 02].

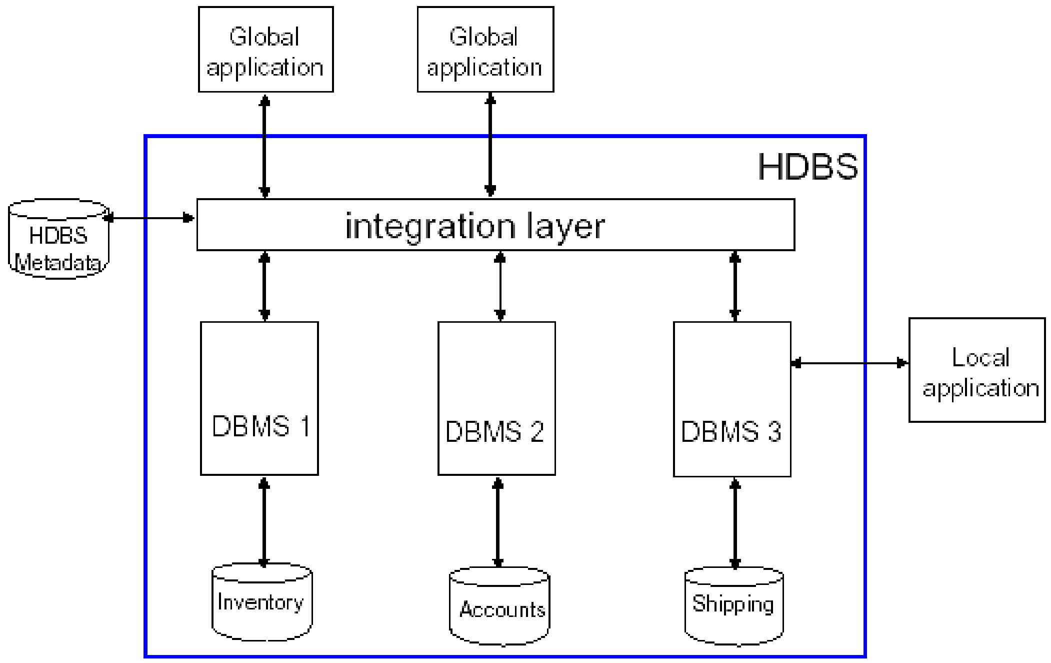


Figure (2.9): Heterogeneous DDBS Architecture

2.13 Strategies of Distributed DataBase Design

There are two alternative approaches to the design of data distribution, these are [Ste 85]:

2.13.1 Top-Down Approach

Top-Down approach is the most attractive for the systems which are designed from scratch, that means we start by designing the global schema and precede by designing the fragmentation of the database, and then by allocating the fragments to the site, creating the physical images. There is complete by performing, at each site, the physical design of data which are allocated to it. However, this approach is mostly used in homogeneous systems, as shown in Figure (2.10).

2.13.2 Bottom-Up Approach

Bottom-Up Appoach approach is based on the integration of existing schema into single, global schema. BY integration; we mean the merging of common data definition and the resolution of conflicts among different representations given to the same data. When existing DBs are aggregated into a distributed DB, it is possible that they use different DBMSs.

However, this approach which is mostly used in the prototypes of heterogeneous system is that to select a common data model, and then to translate it into this unique representation all the different schema of involved DBMSs, as shown in Figure (2.11).

However, the Bottom-Up design of a DDB requires:

1- The selection of a common Db model for describing the global schema of the database.

2- The *translatio*n of each local schema into the common data model.

3- The *integration* of the local schema into a common global schema.

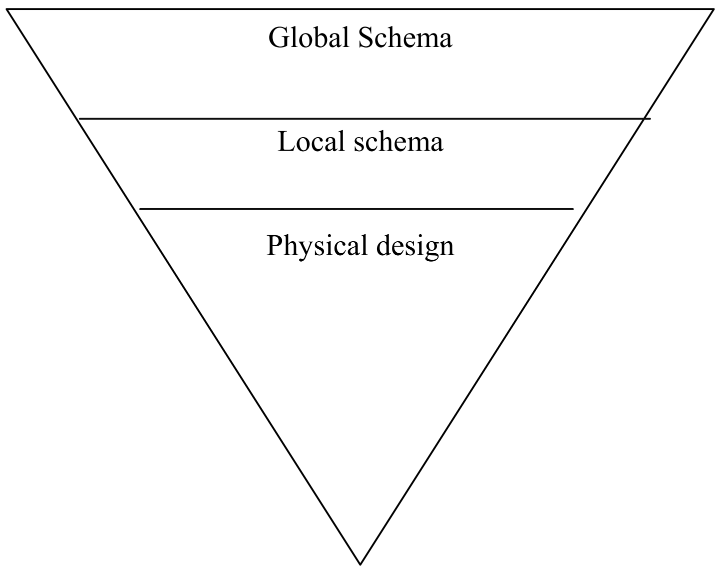


Figure (2.10): Top Down Approach

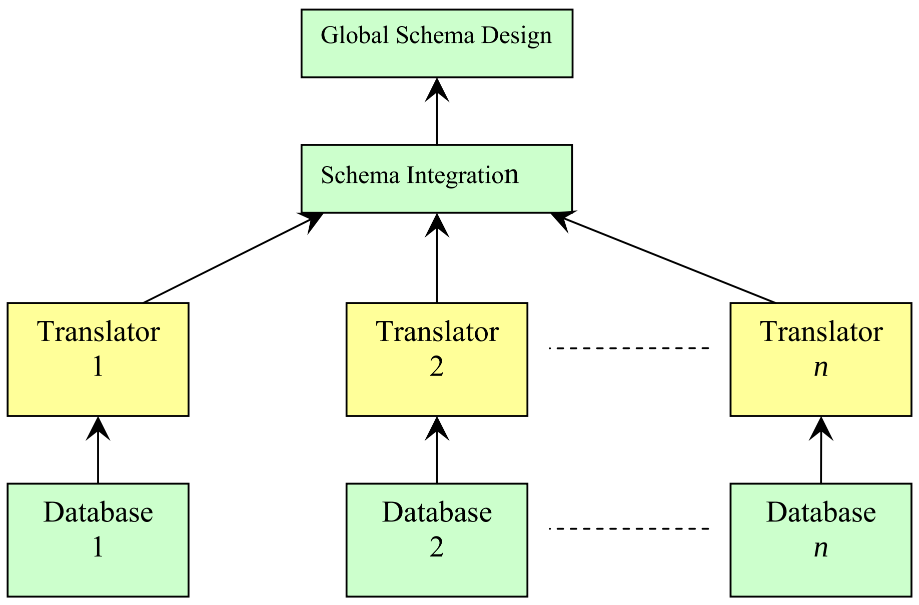


Figure (2.10) Bottom-up Design Approach

Chapter Three
Multi-Database Architectures

3.1 Introduction

The Multi-database system is heterogeneous; that is, each has its own strategy for implementing concurrency, deadlock resolution, and recovery. Additionally, each local system may be based on a different data model.

The multi-database will based on one of the multi-database model a multi-database management system (MDBMS) and the various layers that make up its query processing strategy. A critical component of the MDBMS is the global transaction manager.

3.2 Multi-Database Systems

A multi database system (MDBS) is a service that provides access to data stored in multiple autonomous and possibly heterogeneous. An MDBS integrates a set of autonomous and heterogeneous local DBSs[Ang95]
. In turn each local DBS consists of a local DBMS and a database. Users can access information from multiple sources through global transactions. Operations belonging to global transactions are executed by local DBMSs. Besides global transactions there exist local transactions in a multi database environment. Local transactions result from the execution of local applications. Such applications are typically pre-existing with regard to the integration realized by MDBSs. A computer system is subject o failures. Such failures may provoke loss of information. Hence MDBSs should be able to react in failure situations in order to restore the multi database to a consistent state without human intervention that is automatically. However ensuring reliability in MDBSs is a very complex task. ***First*** of all more types of

failures may occur in MDBSs (e.g. a communication failure which can isolate a local DBS from the MDBS) than in centralized DBMSs. ***Second*** in MDBSs there is a tradeoff between preserving local autonomy and providing an efficient global recovery mechanism.

Two classes of transactions are supported in a multi database environment

1. Local transactions which are transactions executed by an LDBMS outside the control of the MDBS.
2. Global transactions which comprise transactions submitted by the MDBS to LDBMSs. Global transactions may be executed in more than one local system. Thus define a global transaction G_i as a set of subsequences $\{SUB_{i,1}, SUB_{i,2}, SUB_{i,3}, \ldots.. SUB_{i,m}\}$ where each $SUB_{i,k}$ is executed at the local system $LDBS_k$ as an ordinary local transaction.

To client applications multi database systems provide an integrated view of the data stored at component database systems under their control. The abstraction gives applications the illusion that they are accessing a single databases system. This removes the complexity of distribution heterogeneity, integration, transaction, management and administration from the application to the multi database system. A centralized view of a single multi database server defeats the transparency aspect of the multi database abstraction. With a single centralized server, applications are exposed to the nuances of accessing data sources that cross multi database domains.

Better known as heterogeneous multi-database systems, this type of global DBMS is characterized by dissimilar data models, concurrency and optimization strategies, and access methods. The homogeneous distributed database, the data models that compose the global database could be based on the relational, hierarchical, or network model.

Access methods and concurrency control algorithms would utilize locking mechanisms at one site and time stamp strategies at another site. Unlike homogeneous distributed database systems, multi-database systems are heterogeneous; that is, their components and data models are dissimilar. Figure (3.1) shows an example of a multi-database architecture.

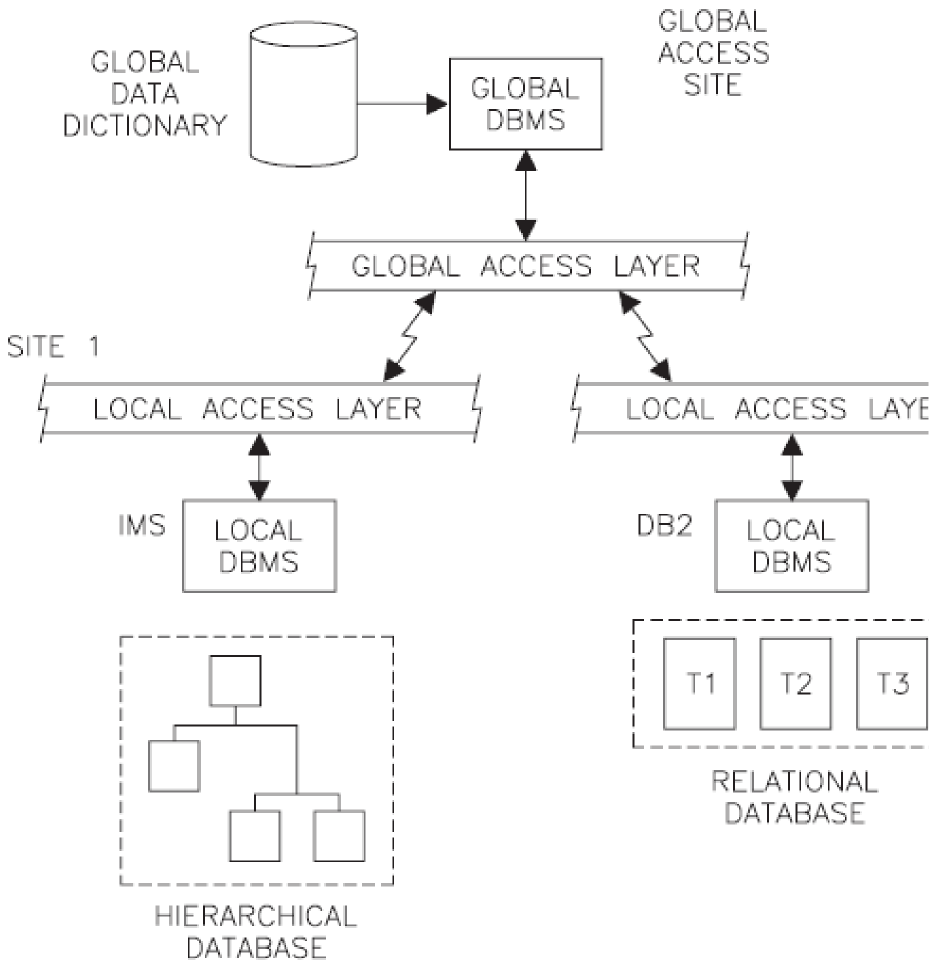


Figure (3.1): Multi-database architecture [Ang 95]

It is composed of a central global location and two distributed sites. The central global access site contains the global access layer and the global data dictionary. Another name for the global data dictionary is the global database schema. The local sites each contain a local access layer, a local database management system, and a database. The local access layers, together with the global access layer, make up the multi-database system.

The local components are based on two different database management systems and data models. The global data dictionary contains information that makes these two databases appear to the user as if they were one large database. When a user submits a distributed query to the global access layer, the query is decomposed and transformed to the appropriate data retrieval language for each site. To the user, it appears that the entire distributed database is based on the relational model (assuming the global data dictionary is based on the relational model).

3.3 Database Schemas

Figure (3.2) shows a simple relational database containing three relational tables. This model requires no pointers or special records to implement the relationships among the tables. These relationships are defined by primary key to primary or foreign key links. These links are called joins and are defined during table creation by specifying which columns act as primary and foreign keys. Table 1 linked to table 2 over the common column C1 in each table. Table 2 is linked to table 3 via the foreign key C2 belonging to each table; that is, column C1 is the primary key of table T1, and column C2 is a foreign key in tables T2 and T3. Atypical query joining these three tables.

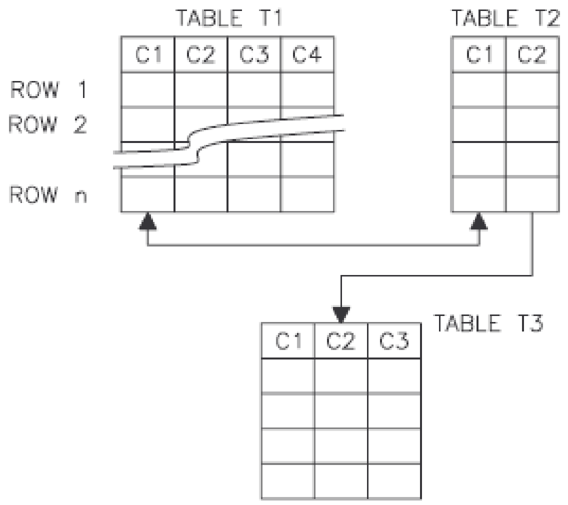


Figure (3.2): A Relational Database

3.3.1 The Hierarchical Model

Figure (3.3) shows a database based on the hierarchical data model. This database contains four files and implements the following relationships via special pointers embedded in the files' records:

File 1 is parent to file 2 and file 3. File 2 is parent to file 4. Notice that a child entity can have only one parent. A child entity can also be a parent to another child entity. Relationships are defined by including pointers between related records of different files.

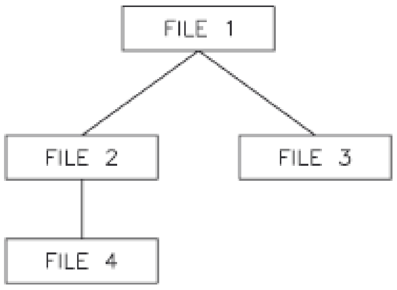


Figure (3.3): A hierarchical Data Model

3.3.2 The Network Model

Figure (3.4) is a block diagram of a database based on the CODASYL model. Relationships among network database files are implemented via special data objects called DBTG (database task group) sets. These records contain pointers that link related records in files, permitting one-way or two-way navigation. An application program reads these DBTG sets and navigates through the links. Unlike the hierarchical model, a child can have more than one parent, thereby increasing the modeling power by allowing designers to define complex relationships among files.

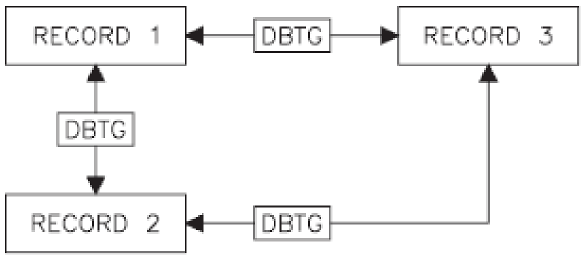


Figure (3.4): The network (CODASYL) model

3.4 Multi-DataBase Managemet Architectures

The central architecture is composed of a central global control layer situated at a dedicated control site [Ang 95]. Its main functions are to accept global user queries, fragment them according to the sites where the data reside, dispatch the query fragments, and coordinate all activities among sites involved in the query so as to generate a final result for the user. At each local MDBMS site there is a local layer that interacts with the global layer to process each query fragment that was routed to each site involved in the global query. Each site contains one or more local DBMS; it is the function of the local MDBMS to further decompose the global query fragments into sub fragments, perform any required translation from the global query language to the local query language, and route the translated fragment to the appropriate local DBMS.

Figure (3.5) shows Multi Database Architectures. The fragment component accepts queries submitted by the global user. It consults a directory for location information and fragments the query into global fragments to be routed to the sites where the desired data are located. The global query fragments are passed to an optimizer/router layer so that an execution strategy can be generated. This execution strategy contains information as to how intermediate result sets are to be generated, joined, and routed to sites that will perform the distributed joins.

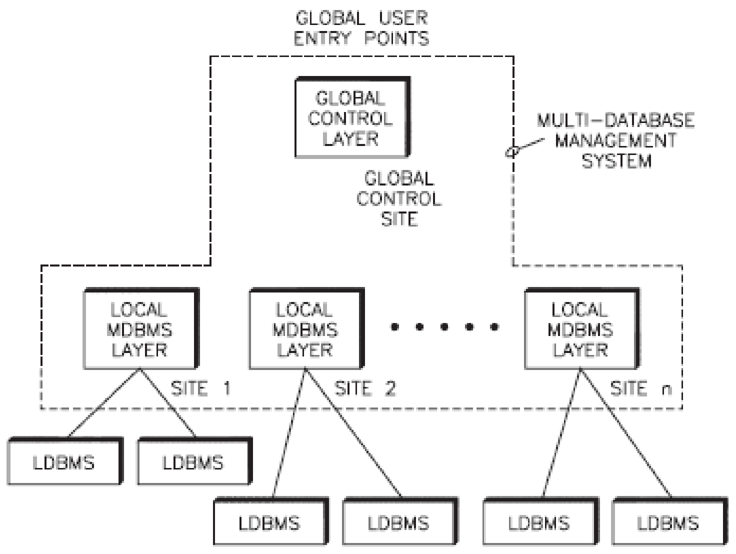


Figure (3.5): Multi-database Architectures[Ang 95]

3.5 Multi-Database Design Issues

The issues and methodology that must be familiar in order to implement a multi-database system. The design protocol is “bottom-up.”

A set of database schemas; possibly the data models are heterogeneous. Must somehow integrate them to generate a global schema that contains all the mappings, translations, and locality information for the components of each local database [Ang 95].

3.6 Simplified Design Steps

Figure 3.6 illustrates the principal steps together with the subtasks that make up the simplified design methodology. The first two steps, schema translation and schema integration, the third step generation, involve the creation of the templates for each relation that is a member of the global schema. Analyze the subtasks for each step. The schema translation step is where each of the local data models is translated to a common data model to facilitate the integration procedure. Usually, the following translation occurs:

1. Hierarchical-to-entity relationship
2. Network-to-entity relationship
3. Relational-to-entity relationship

This translation step has the following subtasks:

1. Select the global schema model.
2. Select the schema integration method.

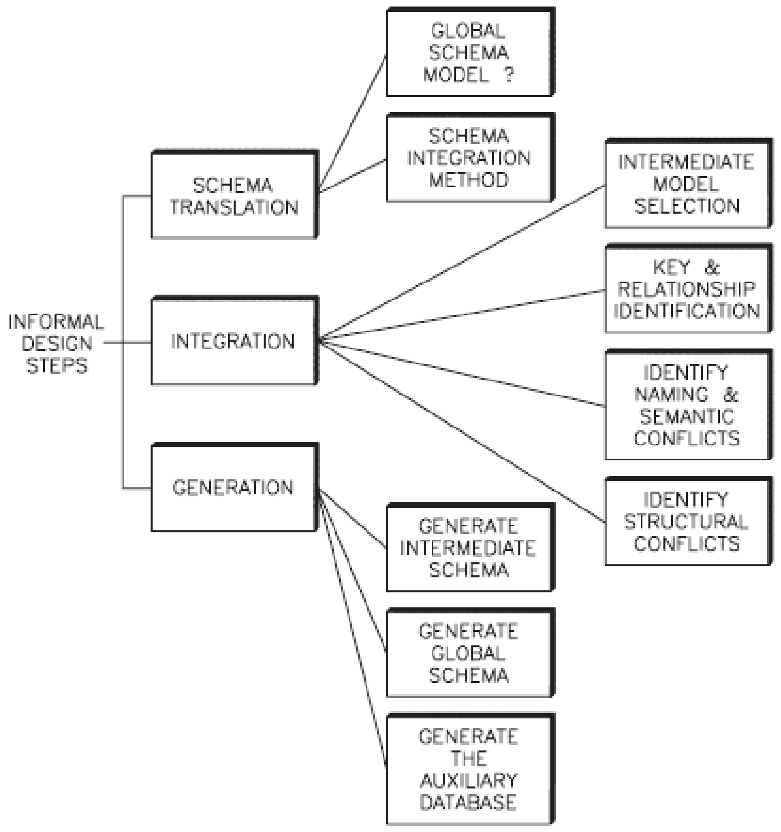

INFORMAL DESIGN STEPS
SCHEMA TRANSLATION
INTEGRATION
GENERATION
GLOBAL SCHEMA MODEL ?
SCHEMA INTEGRATION METHOD
INTERMEDIATE MODEL SELECTION
KEY & RELATIONSHIP IDENTIFICATION
IDENTIFY NAMING & SEMANTIC CONFLICTS
IDENTIFY STRUCTURAL CONFLICTS
GENERATE INTERMEDIATE SCHEMA
GENERATE GLOBAL SCHEMA
GENERATE THE AUXILIARY DATABASE

Figure (3.6): Informal intuitive design steps [Ang 95]

Select the data model that will be used to implement the global schema. Examine the various schema integration techniques. Select the relational model for the global schema because of its current popularity and versatility. Integration, involves the application of a set of techniques that integrate the local schemas into one large, common schema that contains the attributes and characteristics of all the local schemas. As the integration process progresses, the following must be performed:[Ang 95]

1. Intermediate model selection. This is when selects a data model that can be used to integrate all the local schemas. Translate the local schemas to the entity-relationship model, and retain that model as the intermediate global schema representation.
2. Key and relationship identification. As the design process progresses, the primary and foreign keys in each table are identified. The relationships between the tables are also identified. This will facilitate the selection of primary and foreign keys when the final global schema is created.
3. Identification of naming and semantic conflicts. In this phase the designer must isolate the data objects that have either different names but refer to the same object domain or objects with identical names that refer to completely different domains. Additionally, the designer must isolate objects that are semantically different.
4. Identifying structural conflicts. During this phase of the design process; the designer must isolate objects that are represented differently from one schema to the other. The intermediate schema is generated after all the information and documented. The designer then proceeds to implement the global schema by defining the relation definitions that will be used to create each relation in the global schema.

3.6.1 Schema Translation

Usually, database schemas are derived from an entity-relationship diagram. This step requires to perform the inverse of this traditional method. Given a database schema, must

1. Identify the target model.
2. Identify the keys.
3. Derive the relationship from the keys.

For this step must seek the assistance of the local administrator to find out the semantics of each relationship .

3.6.2 Schema Integration

To integrate local schemas to generate a global intermediate schema that will be located at the global control site of the global layer of a multi-database system.

Must have following anomalies:

1. Attribute or relation naming conflicts
2. Semantic conflicts
3. Structural conflicts
4. Relationship conflicts (i.e., many-to-one relationships in a local database schema represented as one to many in another local schema) In addition, identify the key combinations used to implement the relationships among the entities.

3.7 Integration Techniques

Now that the local schemas have been generated, must merge them to create one intermediate schema that will be used to create the global schema. The three methods that will discuss with the references to the researchers that discussed or authored the methods:

1. Pure Binary Schema Integration.

2. Ladder Binary Schema Integration.
3. Nary Integration Techniques.

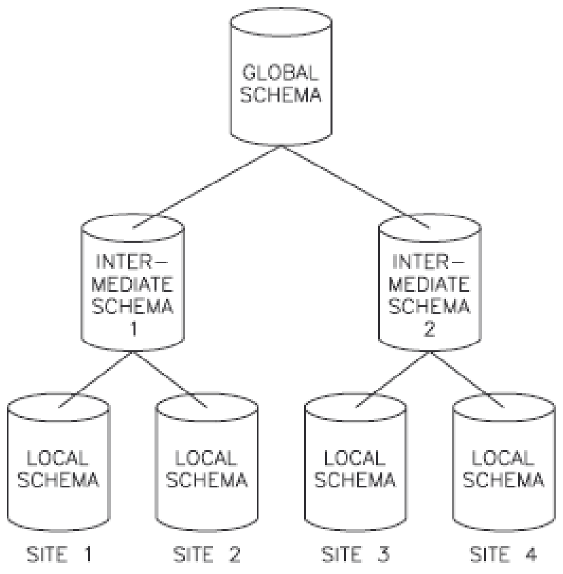


Figure (3.7): Pure Binary schema Integration

Figure (3.7)shows the pure binary schema integration technique. At the bottom of the figure, we see four local schemas. The schema at site 1 is integrated with the schema at site 2 to generate the intermediate schema 1. The schema at site 3 is integrated with the schema at site 4 to generate the intermediate schema 2. Finally, the two intermediate schemas are integrated to create the final globalschema. This technique has the benefit of simplicity. The designer can choose to select the two simplest schemas, integrate them, and then select the next two simplest schemas, integrate them, and so on until the final global schema is generated. By simplest, the two schemas that have the fewest naming, semantic, and structural conflicts.

Also wish to integrate schemas that are almost identical in appearance as to attribute count, type, and domain. Figure (3.8) is a block diagram depicting the binary ladder schema integration method. This method is similar to the pure

binary technique in that two schemas are integrated at a time. This retains the simplicity benefit. Figure (3.8) shows *n* local schemas that require integration.
The schema at site 1 is integrated with the schema at site 2 to generate the first intermediate schema.

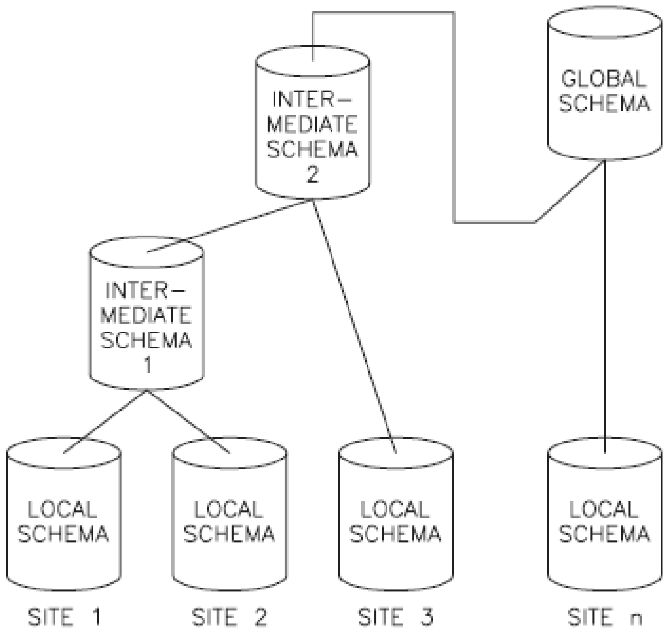


Figure (3.8): Ladder Binary Schema Integration

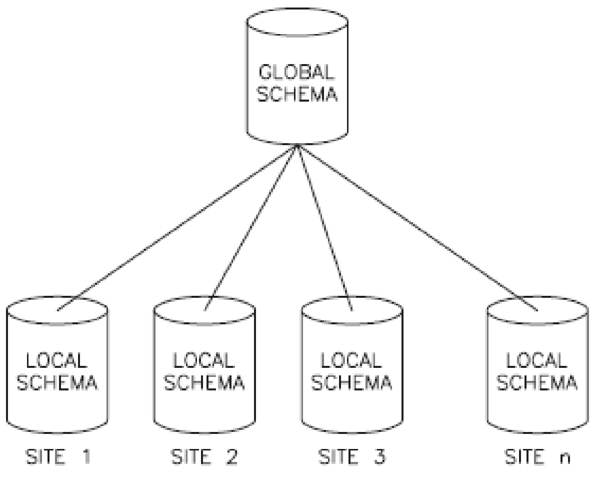


Figure (3.9): Nary schema Integration

The first intermediate schema is integrated with the third local schema to generate intermediate schema 2. Each intermediate schema is then integrated with the next local schema in line until the last local schema is absorbed. The resulting intermediate schema represents the intermediate global schema that can be transformed to the selected target model for the global schema. Figure (3.9) is a block diagram for the Nary integration technique. This is a one-shot strategy. The information for all sites is collected and all the local schemas are integrated at one time. Although this is a faster technique, it is much more complicated to implement[Ang 95].

3.8 Query processing in Multidatabase System

Many of the distributed query processing and optimization techniques carry over to Multidatabase systems but there are important differences [Ozs 06]. The characterized distributed query processing in four steps: query decomposition data localization global optimization and local optimization. This is a generalization of the local query processing steps in centralized DBMSs which include decomposition optimization and execution. The nature of Multidatabase systems requires slightly different steps.

The nature of the multi-DBMS is a layer of software that runs on top of component DBMSs. Each DBMS has its own query processors which execute queries.

Query processing in a Multidatabase system is more complex than in a distributed DBMS for the following reasons

1. The capability of component DBMSs may be different which prevents uniform treatment of queries across multiple DBMSs and sites.
2. Similarly the cost of processing queries may be different on different DBMSs. This increases the complexity of the cost functions that need to be evaluated.

3. There may be difficulties in moving data between DBMSs since they may differ in their ability to read moved data.
4. The local optimization capability of each DBMS may be quite different.

In addition the autonomy of these systems poses problems communication autonomy means that a component DBMS may terminate its services at any time. This requires query processing techniques that are tolerant to system unavailability.

The question is how the system answer queries where the component system is either unavailable from the beginning or shuts down in the middle of query execution. There has not been much work in this area. Design autonomy may restrict the availability and accuracy of statistical information that is needed for query optimization. The execution autonomy of Multidatabase systems makes it difficult to apply some the query optimization strategies. For example semi join –based optimization of distributed joins may be difficult if the source and target relation reside in different component DBMSs since in this case the semi join execution of a join translates into three queries: one to retrieve the join attribute values of the target relation and to ship it to the source relations DBMS the second to perform the join at source relation and the third to perform the join at the target relations DBMS. The problem arises because communication with component DBMSs occurs at a high level of the DBMS.

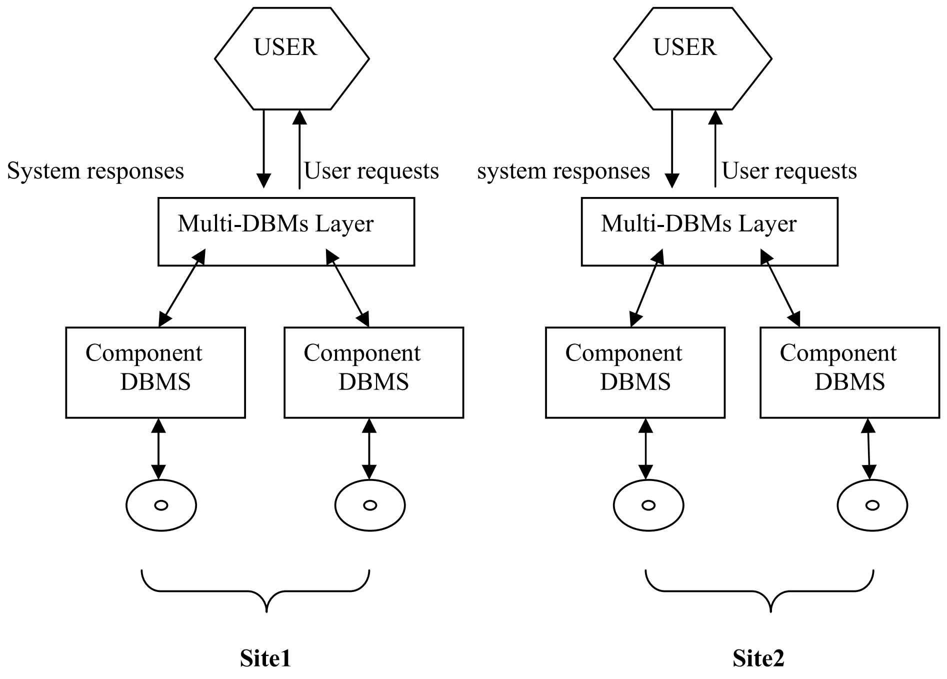


Figure (3.10) Structure of a Distributed Multi-DBMS

3.9 Query processing Layers in Distributed Multi DBMSs

When a query is received at a site their first thing that needs to be done is to split it into subqueries based on data distribution across multiple sites. Rather than its stage across various databases. Therefore the only the only information that is required is the typical data allocation information stored in a global directory. The site that receives the query and performs the splitting called the control site is ultimately responsible for successful completion of the task.

Each sub query is then sent to the site where it is to be processed. The multi DBMS layer at each site further fragments the query for each DBMS that it controls.

At this stage the information within the directory is used. Each subquery is then translated into the language of respective DBMS. Extensive information about the global query language and the individual languages used by the DBMSs needs to be maintained to facilitate translation. Even though this information can be kept within the directory it is common to store it as an auxiliary database.

The queries submitted to the component DBMSs are processed following decomposition optimization and execution steps. The decomposition step involves the simplification of a user query that is specified in some relational calculus and its translation to an equivalent relational algebra query over the conceptual schema. The optimization step involves the reordering of relational algebra operations as well as determination of the best access paths to data. The resulting schedule is then executed by the run-time support processor [Ozs 06].

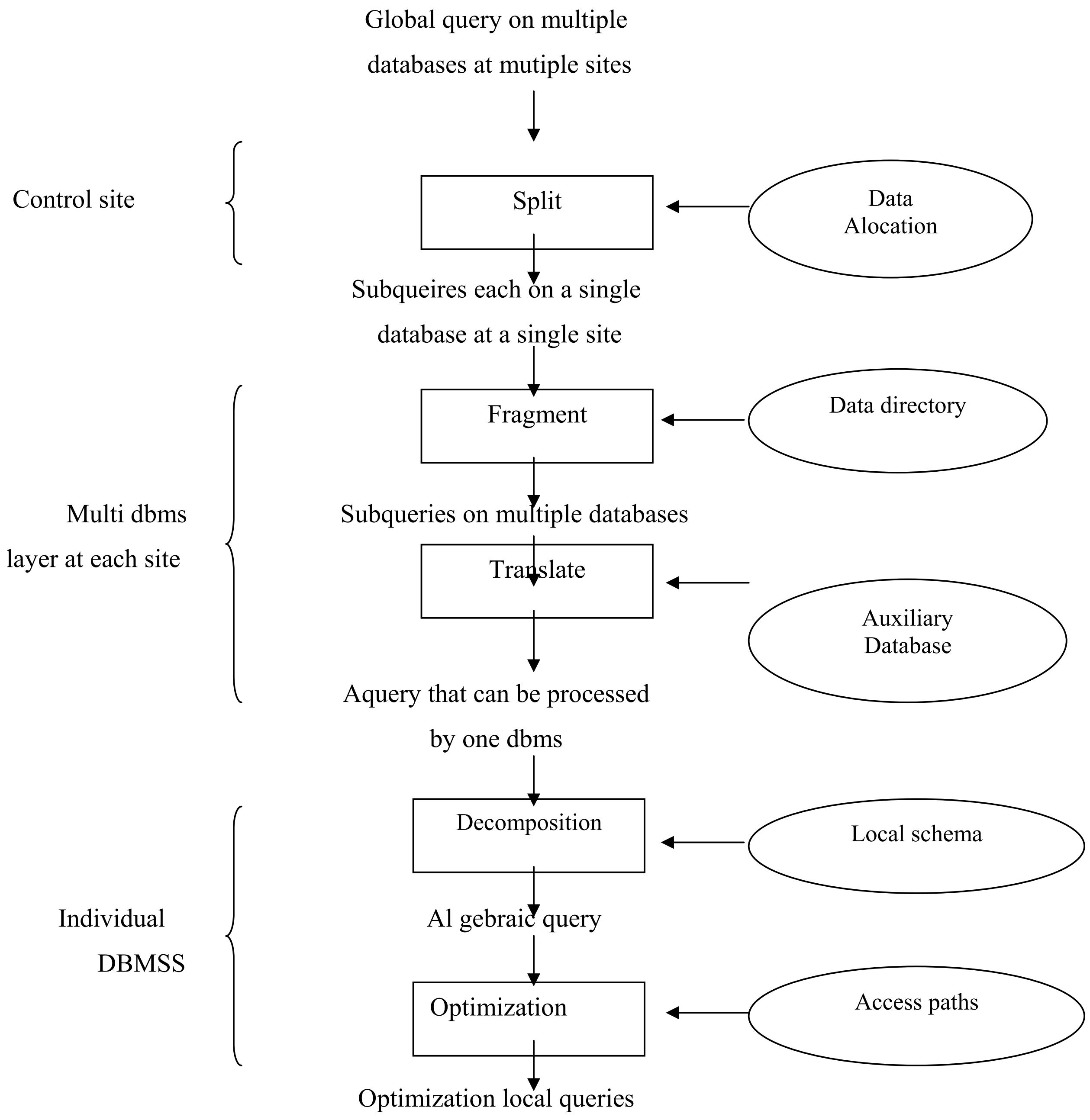


Figure (3.11): Query Processing Steps in Multidatabase Systems[Ozs 06]

In Figure (3.11) translation information is stored in a separate auxiliary database. There is no overriding principle that dictates separation of the global directory from the auxiliary database .In fact there are prototype heterogeneous systems. The auxiliary database contains information describing how mappings from to participating schemas and global schema can be performed. It enables conversions between components of the database in different ways. For example if the global schema represents temperatures in Fahrenheit degrees but a participation database uses Celsius degrees the auxiliary database must contain a conversion formula to provide the proper presentation to the global user and the local databases. If the conversion is across types and simple formulas cannot perform the translation complete mapping tables could be located in the auxiliary as illustrated in the age category relation.

3.10 Transaction Management

Among all database interoperability problems transaction management has probably the most extensively. The challenge is to permit concurrent global updates to the component databases without violating their autonomy. In general it is not possible to provide the same semantics homogeneous distributed DBMSs without violating some autonomy.

Execution autonomy implies that the global transaction management functions are performed independent of the component transaction execution functions. In other word the individual component DBMSs more specifically their transaction managers are not modified to accommodate global updates. Design autonomy has the additional implication that the transaction managers of each DBMS may employ different concurrency control and commit protocols.

3.10.1 Transaction and Computation Model

The MDBS architecture involves a number of DBMSs each with its own transaction manager called local transaction managers or (LTMs) and a multi-DBMS layer on top. The transaction manager of the multi-DBMS layer is called the global transaction manager or GTM since it manages the execution of global transactions[Ozs06].

In a Multidatabase system there are two types of transaction: local transaction which are submitted to each DBMS and global transactions which are submitted to the multi-DBMS layer as shown in figure (3.12).

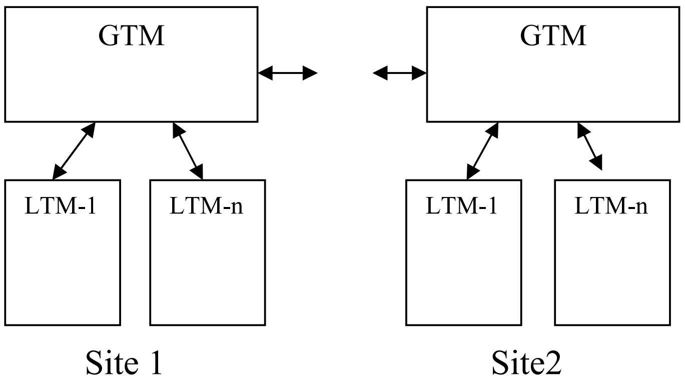


Figure (3.12): Distributed Multi-DBMS Transaction Management

3.10.2 Multidatabase Concurrency Control

There have been many proposals for ensuring consistency of concurrency executing transactions in a Multidatabase environment. The concurrency control algorithms maintain the consistency and isolation properties of transactions. Given the autonomy of the component DBMSs it is not easy to maintain these properties.

Concurrency control algorithms synchronized concurrent transactions by ordering their conflicting operations such that a serialization order can be maintained among transactions[Ozs06].

3.11 Multi Database Integration

Database integration involves the process by which information from participating databases can be conceptually integrated to form a single cohesive definition of a multi database[Ozs 06]. In other word it is the process of designing the global conceptual schema. In multi database, a number of databases already exist, and the design task involves integrating them into one database (server side). The bottom-up approach is suitable of this type of environment. The starting point of bottom up design is the individual local conceptual schemes. The process consists of integration local schemes in to global conceptual schemes. This type of environment exists primarily in the context of heterogeneous data base. Database integration can occur in two steps (figure 3.13) schema translation (or simply translation) and integration

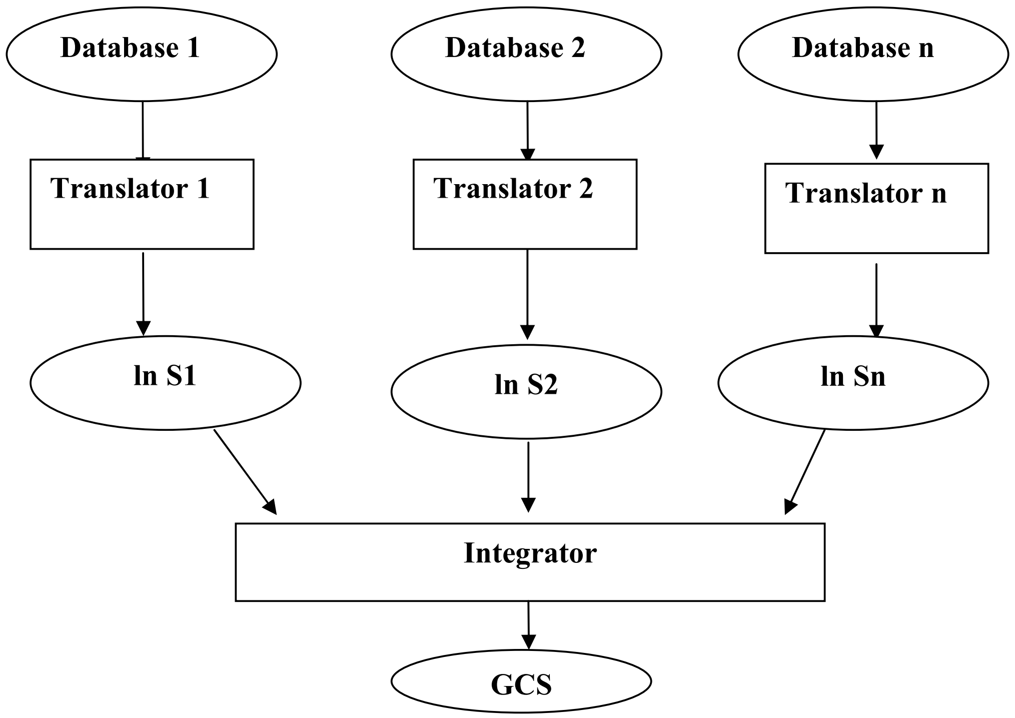


Figure (3.13): Database Integration Process [Ozs 06]

The design process in multi database systems is bottom-up. In other words the individual databases actually exist and designing the global conceptual schema
involves integrating these component databases into a multi databases. In the first step the component database schemas are translated to a common intermediate (InS_1, InS_2 …InSn) representation[Ozs06]. As a principle it should be one that is sufficiently expressive to incorporate the concepts available in all the databases that will later be integrated. Clearly the translation step is necessary only if the component databases are heterogeneous and each local schema may be defined using a different data model. In recent years commercial interest has been on the integration of multiple relational databases where this translation step can be by past. There is some recent work on the development of system federation in which systems with similar data models are integrated together (e.g. relational systems are integrated into one conceptual schema and perhaps object databases are integrated to another schema) and these integrated schemas are combined at a later stage. In this case the translation step is delayed providing increased flexibility for applications to access underlying data sources in a manner that is suitable for their needs. In the second step each intermediate schema is integrated into a global conceptual schema. In some methodologies local external schemas are considered for integration rather than local conceptual schemas since it may not be desirable to incorporate the entire local conceptual schema in the multi database.

Chapter Four

The Proposed System Design

4.1 The System Components

The proposed system based on multi database system which consists of a three sites, connected together via a communication network, as shown in Figure (4.1).

The main components of proposed system in this figure are:-

1. Master Database-FoxPro / Server: The backend of Multi-Database where the entire database is manipulated, managed, and stored.

2. Database Access / Client: The front-end of Multi-Database where the access requests are issued, which is executed under Microsoft Access environment.

3. Database SQL / Client: The front-end of Multi-Database where the access requests are issued, which is executed under Microsoft Visual FoxPro environment.

4. Communication System (CS): It enables the communication between the two clients and server.

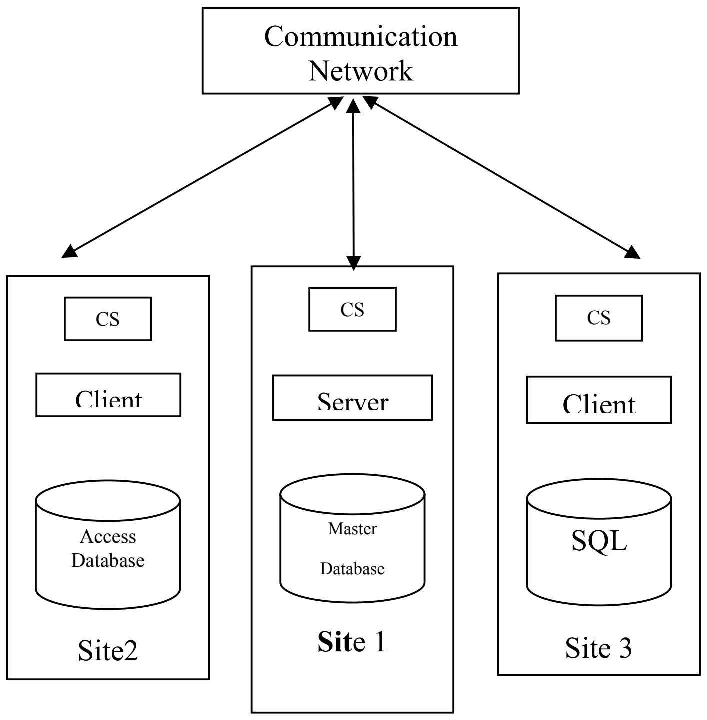


Figure (4.1): the Components of the Proposed System

4.2 Heterogeneous or Multi-database System (MDBS)

This is a layer on top of an existing DB system that integrates different software environments and machine architectures with different DBMSs which are all heterogeneous.

The DBMSs employ different data model (or no models at all) - i.e. hierarchical relational network and different DDLs SMLs and may have different concurrency control mechanisms and transaction processing environments.

A multi-database system creates the illusion of logical DB integration without requiring physical DB integration.

Advantages

1. Extended user capabilities: Users access and share data without the added burden of learning the intricacies of different DBMSs.
2. Pre-existing programs and procedures are still operational in the integrated MDBMS.
3. The common data model used to employ this layer is the relational model and in most cases the DMS is SQL.
4. A complex provision of a common Conceptual Schema: is an integration of all conceptual schemata of the heterogeneous DBMSs.

4.3 Transaction Management and Concurrency and Serializability

Two types of transactions Management

Local transactions executed by each local DBMS and outside of the MDBMS system control

Global transactions executed under the MDBMS system control.

The MDBMS has NO control over the execution of local transactions.

Each local DBMS must ensure some concurrency control such as two phase locking or timestamps and to ensure that its schedule is serializable and avoidance of local deadlocks.

4.4 The System Design

The design of this system requires working on three application environments:-

4.4.1 Microsoft Visual FoxPro Environment

This application is used to build *FoxPro* database and allocated at (FoxPro) , this site contain Phone system which database consist of one database table such as in Table (4.1).

Table (4.1) FoxPro-Phone Database Table

Field Name	**Field Type**
Name	Text
Mobile	Long
Mail	Text
Address	Text

4.4.2 Microsoft Access Environment

This application is used to build Microsoft Access database and allocated at *Access* Client, this site contain information research system of university, which consists of two database tables at this site. Table (4.2) illustrates these tables.

Table (4.2) Access Database-Research Tables

Field Name	**Field Type**
Project Name	Text
Department	Text
Abstract	Text
Supervision	Text
Year	Date
Keywords	No.
Document of Project	Text

Field Name	**Field Type**
Project Name	Text
Student Name	Text

4.4.3 Microsoft SQL Server 2000

This DataBase is called bank (bank).It contains information about people accounts and visa card numbers. It contains one table.

Table (4.3) SQL Database-Research Table

Field Name	**Description**
P- Name	Person Name
P-account-id	Account number of person
P-password	Account password person
P-visa	Visa card number of person
P-account	Account Value (amount of money) of person

4.4.4 Microsoft Visual Basic Environment.

The three sites interfaces built in Visual Basic Environment; Visual Basic environment provides a wealth of ***tools*** for creating and accessing database on both individual and networks. The major these tools it listed below:-

I. Data controls

The Visual Basic *data control* is the control used to gain access to database tables. You can have more than one data control in your program and more than one data control on a single form. Bound data controls are the same as any other Visual Basic control objects, except that they have been given additional properties, *events*, and methods that allow you to "bind" them directly

to one or more data tables. This binding makes it easy to create data-aware input and display
objects that you can use to perform data input and display with very little program code.
The data control may be gives access to database without any programming; you can set a few properties of the control and use regular controls such as textboxes to display the values of the fields in the database. Using bound controls simplifies your programming chores a great deal. Most bound controls automatically handle the various chores related to processing data entry and display for databases. The bound controls make it easy to write Visual Basic programs that handle all (or nearly all) of the following processes:

1. Loading data from the database into a Visual Basic data object.
2. Selecting the data record(s) requested by the user.
3. Loading form controls with values in the requested record(s).
4. Trapping simple user input errors.
5. Updating the data object with modified data from the form controls.

II. DataEnvironment

DataEnvironment component lets you design a connection to database and retrieve desired records. With DataEnvironment visual Basic application sees three objects:

1. A Connection Object, which establishes a connection to the database, but it a local file or a remote SQL server.
2. A Command Object, which executes commands against the database.
3. A RecordSet Object, which holds the records retrieved from the database or records to be updated on the database.

III. Data Access Objects (DAO)

The Data control provides a means of quickly developing database applications with little or no code, but it limits your access to the underlying database. *DAO* is a structure of objects for accessing databases through your code.

All the functionality of the data control is also available to your code, through the *DAO*; thus, using *DAO* requires more coding than using the Data control, it offers complete programmatic access to everything in the database, as well as significantly greater flexibility.

IV. Crystal Reports Pro

Crystal Reports Pro is a complete program that helps you define reports, save their definitions to disk, and then run these reports against databases to create final printouts. Crystal Reports Pro has an added feature that lets you run the final reports from within your Visual Basic 6 application using the Crystal Reports Pro control, which ships with Visual Basic 6.

Crystal Reports Pro is a banded report writer. A banded report writer treats all output as "bands" of data. Each band has its own processes (such as functions it performs) and settings (properties) that you can manipulate in order to create the report layout and behaviors you need. Here are the main bands in Crystal Reports Pro.

1. The header and footer bands.
2. The detail band.

4.5 The Main Block Diagram of Proposed System

This section provides important flowcharts that illustrate the work structure and the internal design of the proposed system. The system is software program that

connect and manage more than data source in one time, in our proposed system it connect to three databases (Microsoft Access, Microsoft Visual FoxPro database and Microsoft SQL Server 2000 database). The managements represent the main database operations which are (insert, delete, update, search, filtering and reports)

Figure (4.2) and (4.3) illustrate the main block diagram of the proposed system.

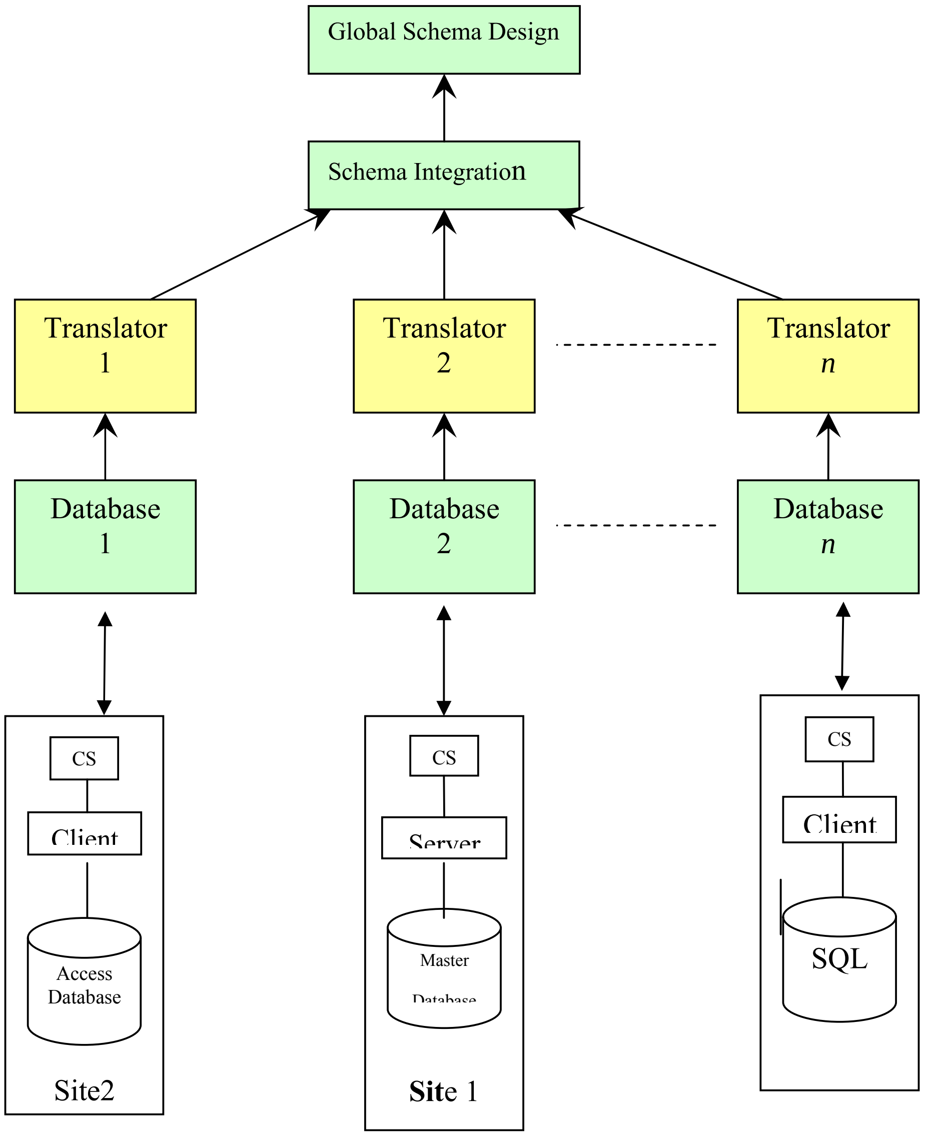


Figure (4.2) Integration and Translation Schema

Schema Integration

1. Preintegration: identify the keys and defines the ordering of the binary processing approach.
2. Comparison: Identification of naming and structural conflicts.
3. Conformation: Resolution of the naming and structural conflicts.
4. Restructuration and Merging of the different intermediate schema to the global conceptual scheme (GCS)

Schema translation means the translation of the participating local schemes into a common intermediate canonical representation

1. Schema translation is the task of mapping one schema to another.

2. Requires the specification of the target data model for the global conceptual schema
3. Some rare approaches did merge the translation and integration phase, but increases the complexity of the whole process.

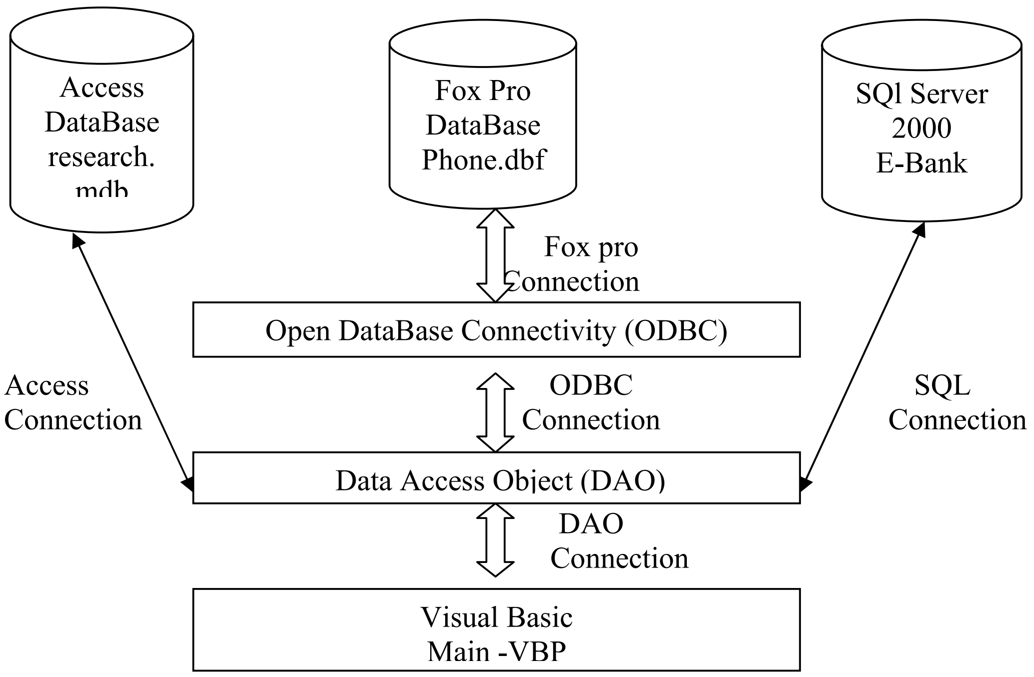


Figure (4.3) proposed system Architecture

4.6.1 Master Database Flowchart

The main operations and real work of this project are manipulated at this site. The ODBC used to open the database from the master. The DAO used to design new database (Mastering database), this new database can be saved as binary file that contain all the required information, to retrieving from both sites. The Figure (4.4) shows the main flowchart for the Master database design. This site contains main menu of the proposed system, the user can interactive with this main menu by followed these steps:

1. The user must open different types of databases then selects the required tables.

2. The system will be analysis this database tables. The fields' names, Description of those tables are determined and displayed.

3. The user can verify connection between this different types database master.

4. If there are duplicated exist in the databases, the system will ignore it.

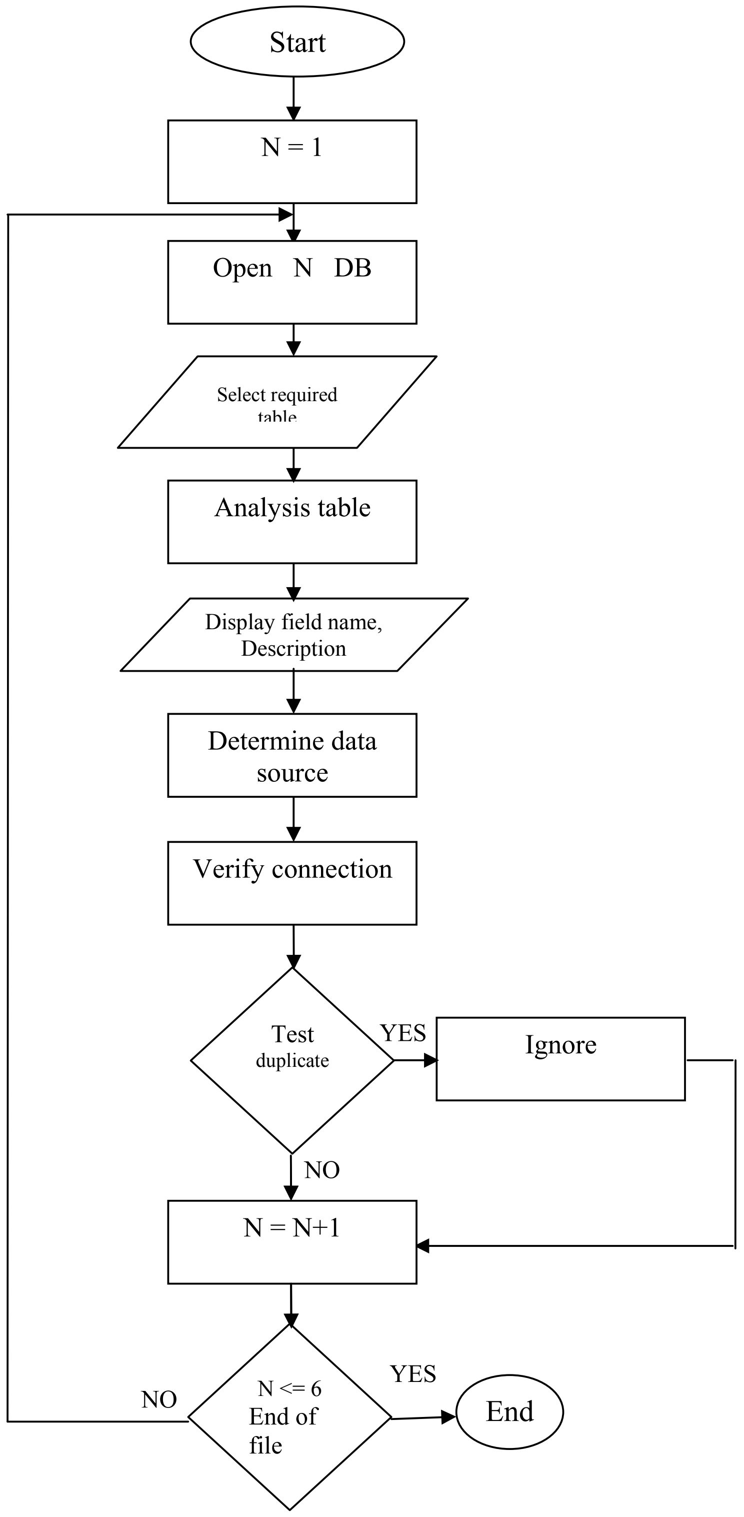


Figure (4.4) Master DB Server Flowchart

4.6.2 Access Client flowchart

ActiveX data control is used to open Access database and to bind with it tables. In this section, dedicated to explain the flowchart of Access Client design, as shown in Figure (4.5), that illustrate execute and perform most important operations of Research proposed system, step by step:

1. Select Access database.
2. The required database table must be determined.
3. The user can perform the necessary operations on the opening database (Adding New Record, Delete Record, Searching Record, and the Exit of the System).
5. If the user selects Add operation, the user can add a new record to the database table.
6. If the user selects Delete operation, the user can delete the old record from the database table.
7. If the user selects Query operation, the user can search on this database table; by selecting field name, certain operator, he can enter the compared value and make search to show required records.
8. Finally, when the user selects Exit, he can quit from this system.

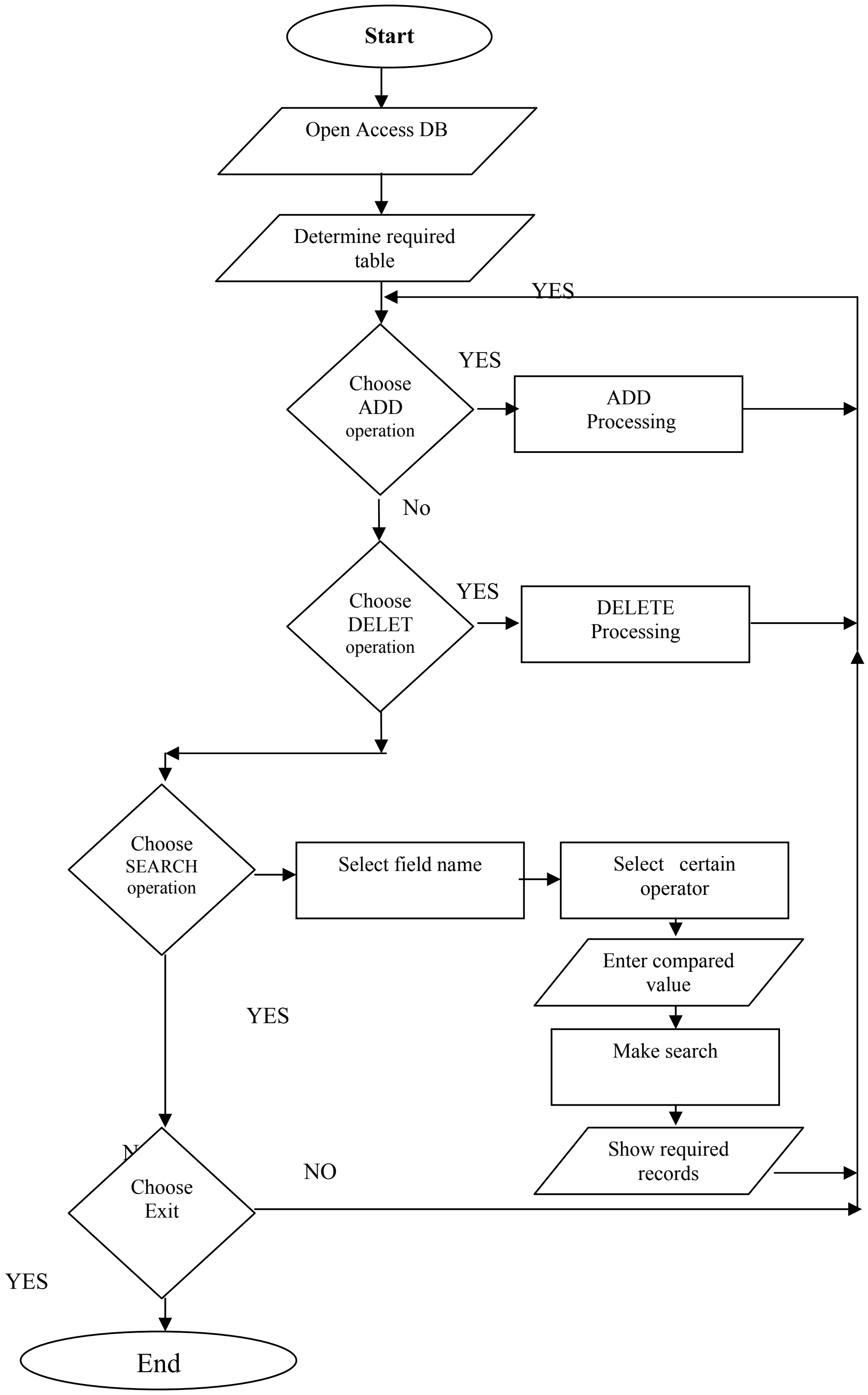


Figure (4.5) Access Client flowchart

4.6.3 FoxPro Client Flowchart

Data Environment control is used to open FoxPro database and to manipulate the operations that performed on the table of FoxPro database. Figure (4.6) shows that flowchart of FoxPro Client design. The database operations are: Add Database, Delete Database, Update Database, Perform Report, Searching Record, Exit of the System. To explain the idea of the proposed university salary system at this site, and to execute and perform most important operations see, notice following steps:

1. The user must determine the database source, and the required table.

After the user is completing successful all above steps, the main menu screen of this system will appear.

2. If the user selects Add new record, a blank record will display, the user must fill this blank with the required information.

3. If the user selects Delete record, the user can delete the old record from this database table.

4. If the user selects Update record, the users can update any existing record in this database table.

5. If the user select Report database, the user can see final report

6. If the user selects Search records, the user can search on this database table; by SQL search.

7. If the user selects Exit, the user can quit from this system, else the user is still in main menu screen.

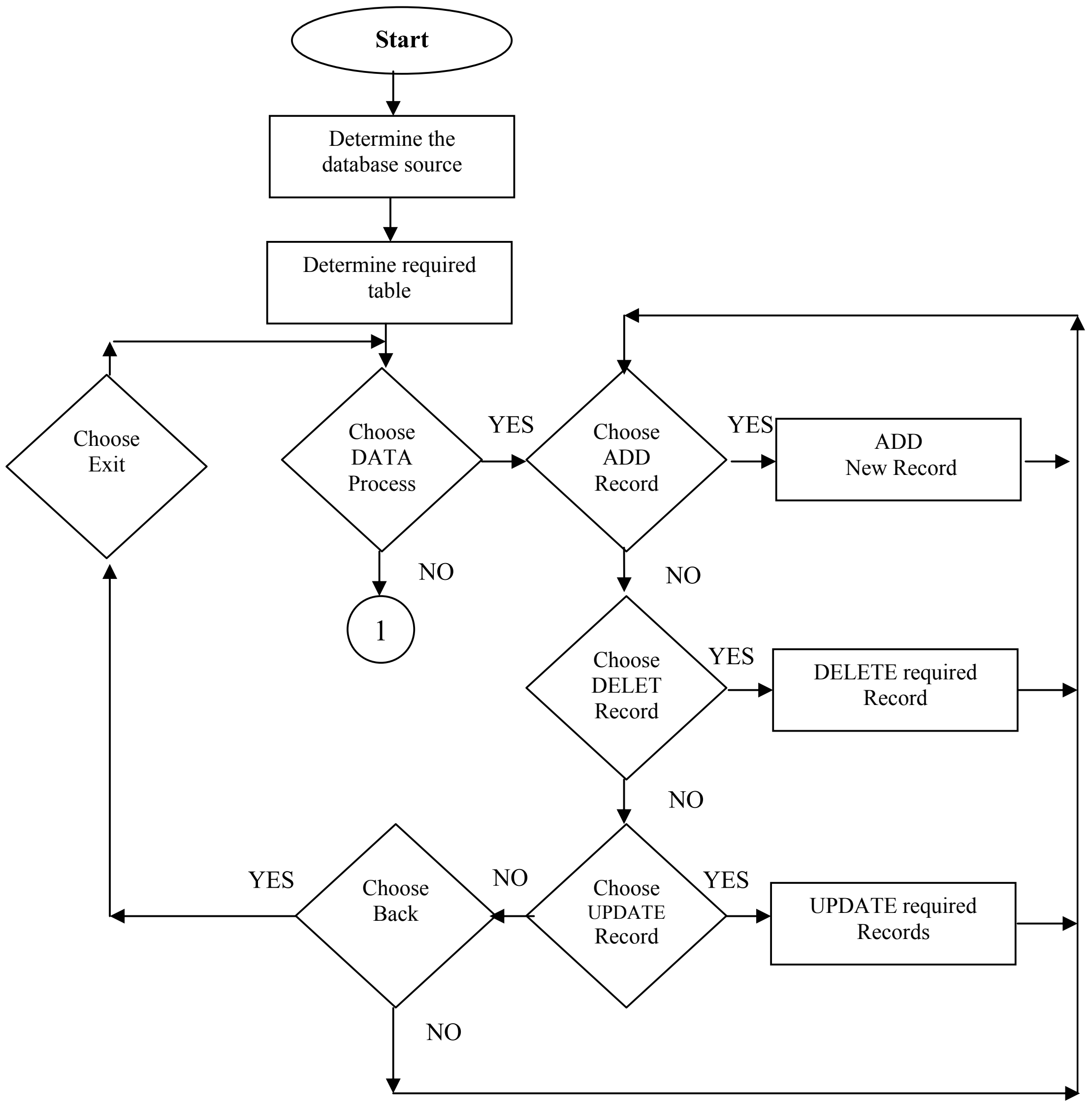


Figure (4.6) FoxPro Client Flowchart

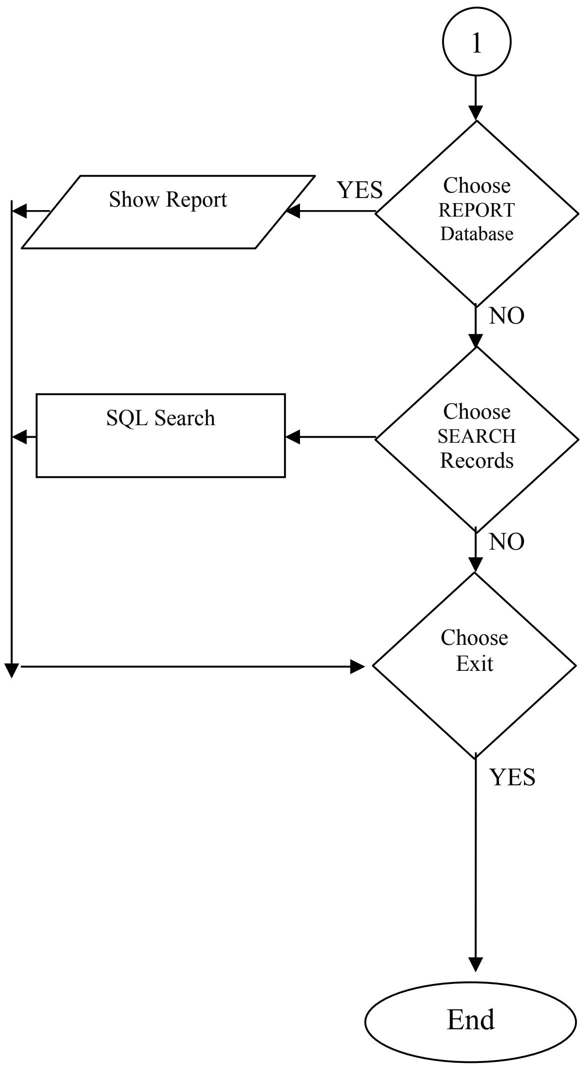


Figure (4.6) Continued FoxPro Client Flowchart

Chapter Five
The System Implementation

5.1 Introduction

The aim of the proposed system is to implement an application interface to connect to three types of database management systems (SQL Server, Microsoft Access and Microsoft Visual FoxPro). The main program can manage these database through the main operations (Add, Delete, Update and Search) on records. This chapter explains in details the interfaces of the proposed system.

5.2 System Requirements

In order to make the proposed system working properly, the following requirements should be provided. These requirements are Hardware and Software as shown :

5.2.1 Hardware Resources

The hardware resources required for this project are as follows:

1. One computer machine used as server. The server computer must have minimum a large main memory (to support buffer management) and a high capacity disk 85GB (for storing the entire master database).
2. Two computers used as clients 60GB minimum for each disk. These computers dedicated to the client must be suitable for interaction with the user and supported with two applications (Bank/SQL system and Research Access system). Figure (5.1) shows these required resources.

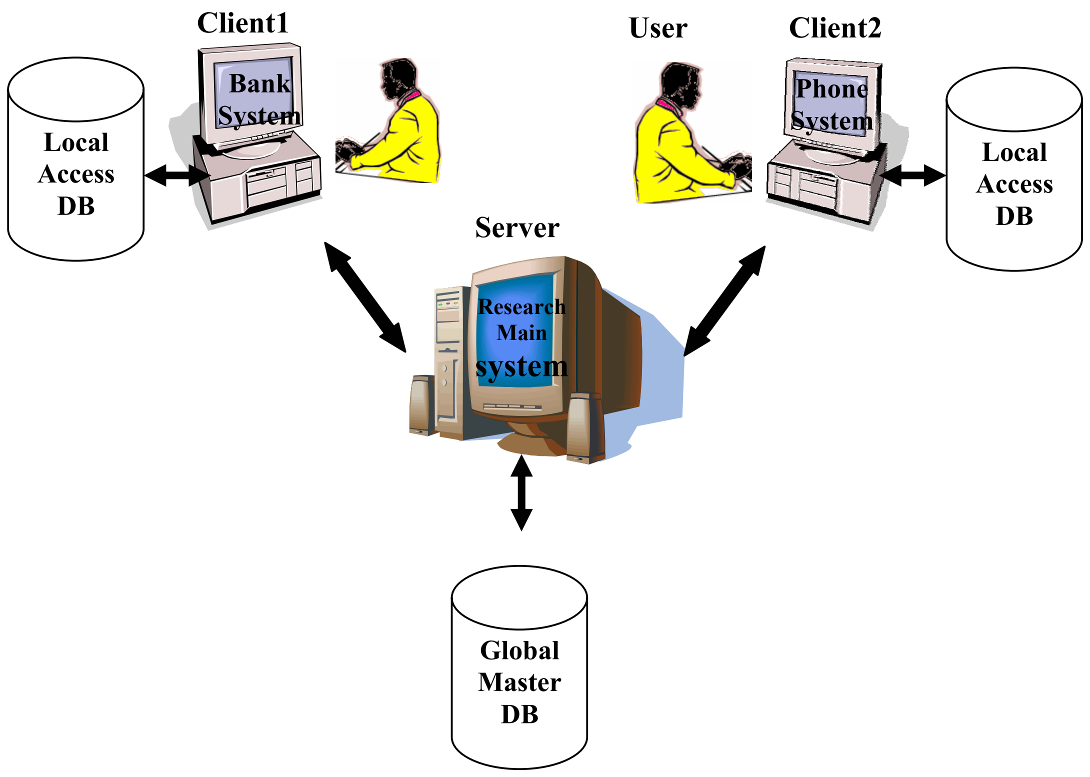


Figure (5.1) Required Resources of the proposed system

5.2.2 Software Resources

The software resources required for this project are :

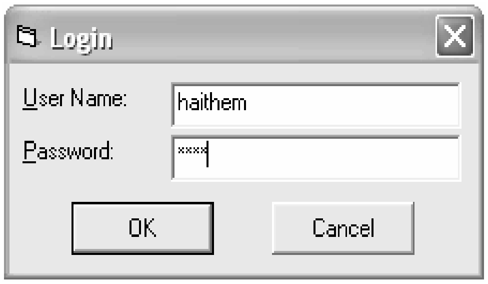
1.Relational Databases (Microsoft Access, Microsoft Visual FoxPro environments, SQL Server 2000).

2. Programming Language development environment (Visual Basic 6).

3. A Windows based PCs with administrative rights would be required for the successful implementation.

5.3 The Proposed System Implementation

The proposed system can be explained in details by describing each window interface in details. The proposed system is programmed by using Visual Basic 6.0. When the program is running the main Login Window interface is appears as shown in figure (5.2).



User Name:	haithem
Password:	****

Figure (5.2):Login Form

This window used to allow accessing to program by only valid user. When the user is valid the main window will appears otherwise an error message box will appears to indicate invalid user.

The main window of the proposed system contains all the buttons that make the connections with other database management systems. These buttons are (Access Database – Electronic Library, SQL Server 2000 – Electronic Bank, FoxPro – Phone Book, About and Exit) as shown in figure (5.3).

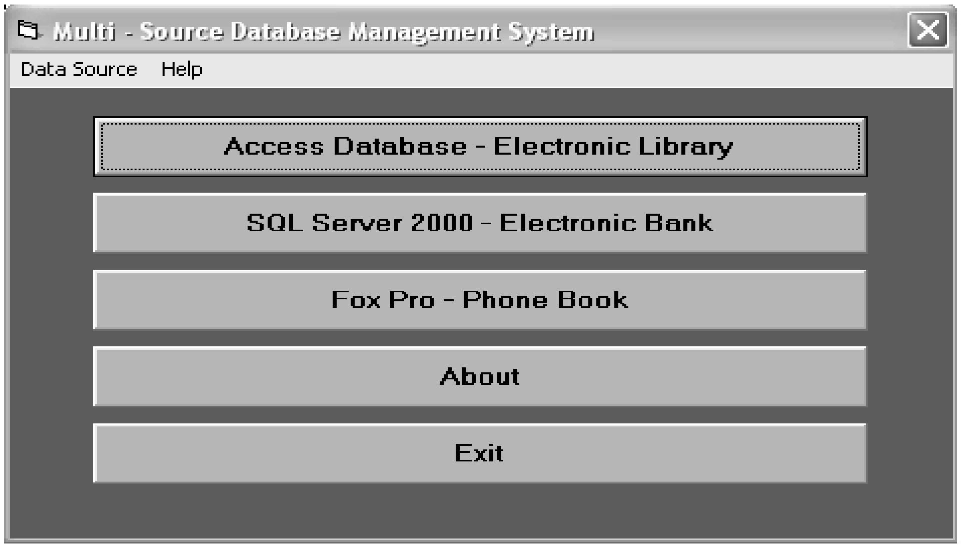
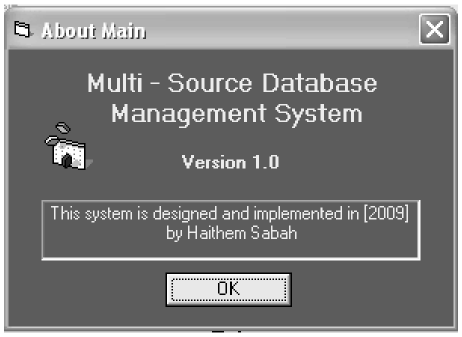

Multi - Source Database Management System
Data Source
Help
Access Database - Electronic Library
SQL Server 2000 - Electronic Bank
Fox Pro - Phone Book
About
Exit

Figure (5.3): The proposed system main window.

The Exit button is used for terminating program. The About button is used for displaying information window about the proposed system. This window is as shown in figure (5.4).



Multi - Source Database Management System

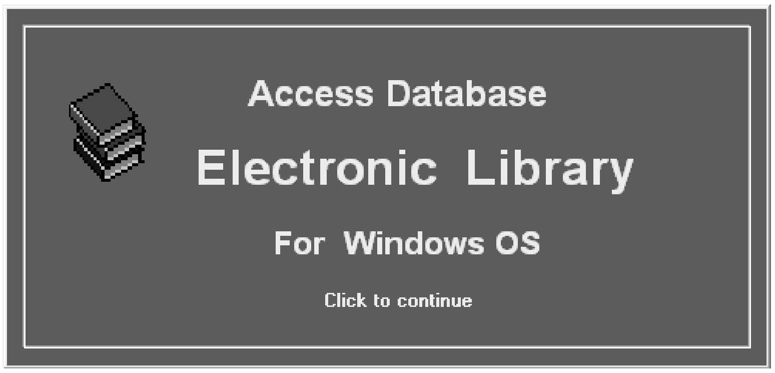
Version 1.0

This system is designed and implemented in [2009] by Haithem Sabah

Figure (5.4):About window

5.3.1 Connecting to Access Database

Connecting to Access database can be done through press (Access database – Electronic Library) button from the main window of the proposed system. The window of this system is as shown in figure (5.5). This window is called Splash screen it display information about this sub system. When click any key the main window of this sub system will appears as shown in figure (5.5).



Access Database

Electronic Library

For Windows OS

Click to continue

Figure (5.5):Electronic Library Splash Screen

The Electronic Library main windows contains four buttons (Projects, Queries, About and Exit). The Project button used to manage adding and deleting projects from database. While Queries button used to manage searching and updating projects in database. This window is appears as shown in the figure (5.6).

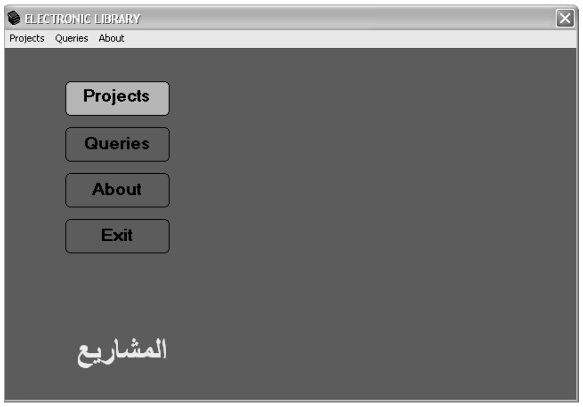


Figure (5.6):Electronic Library Window

A – Adding New Project

This is done by clicking projects button then Add New. This will make project display ADD window like the following windows. It contains all fields of the database.

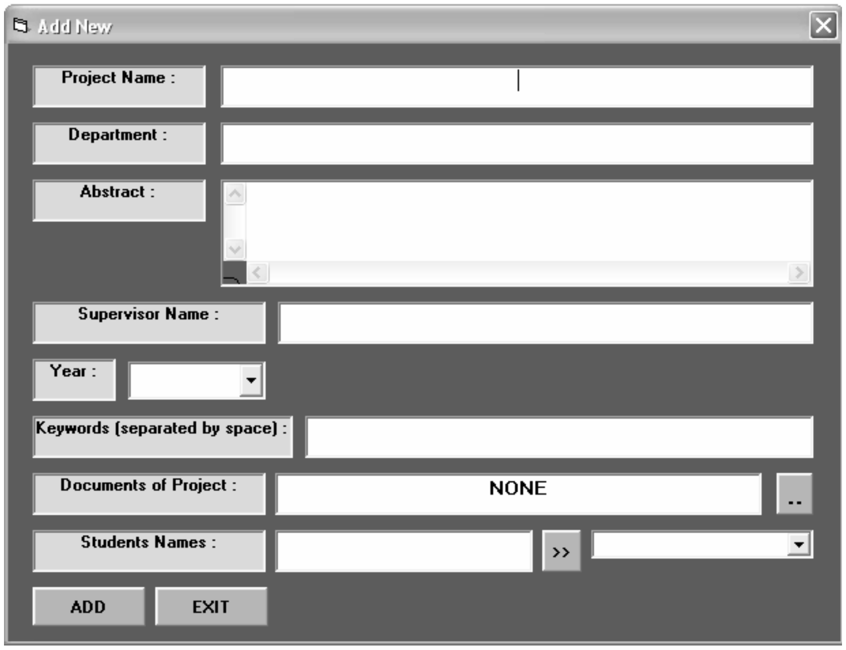


Figure (5.7):Add New Window

B – Deleting Project

This is done by clicking projects button then delete. This will make project display Delete window like the following windows .

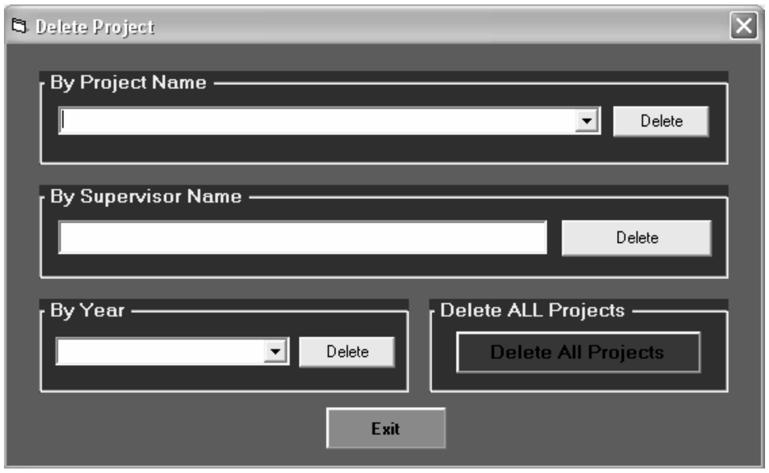
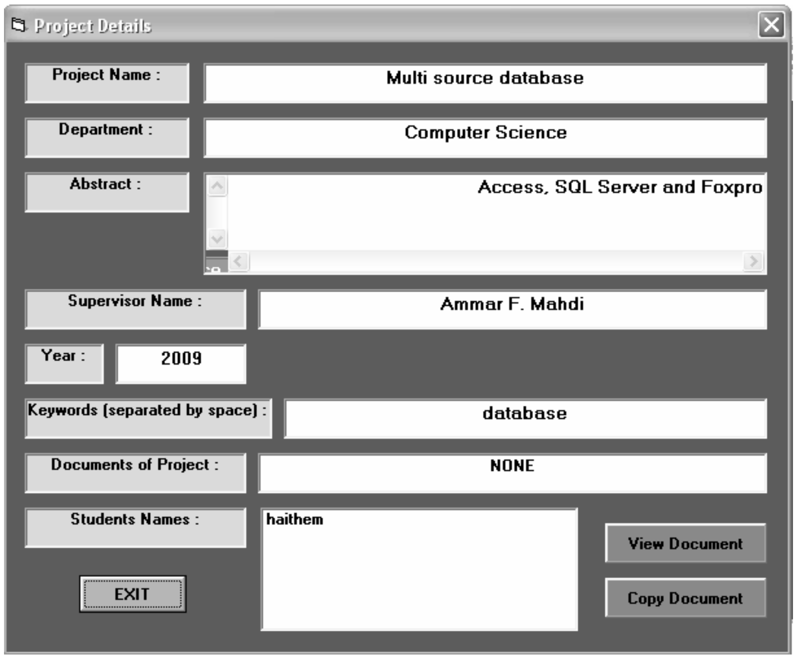


Figure (5.8):Delete Window

C – Searching and Updating

This is done by clicking Queries button then select one method of search (by name, student or other fields). This will make project display Details window like the following windows in figure (5.9).



Project Name : Multi source database
Department : Computer Science
Abstract : Access, SQL Server and Foxpro
Supervisor Name : Ammar F. Mahdi
Year : 2009
Keywords (separated by space) : database
Documents of Project : NONE
Students Names : haithem

Figure (5.9):Details Window

5.3.2 Connecting to SQL Server 2000 Database

Connecting to SQL Server database can be done through press (SQL Server 2000 database – Electronic Bank) button from the main window of the proposed system. The window of this system is as shown in figure (5.10). This window is called Splash screen it display information about this sub system. When click any key the main window of this sub system will appears as shown in figure (5.10).

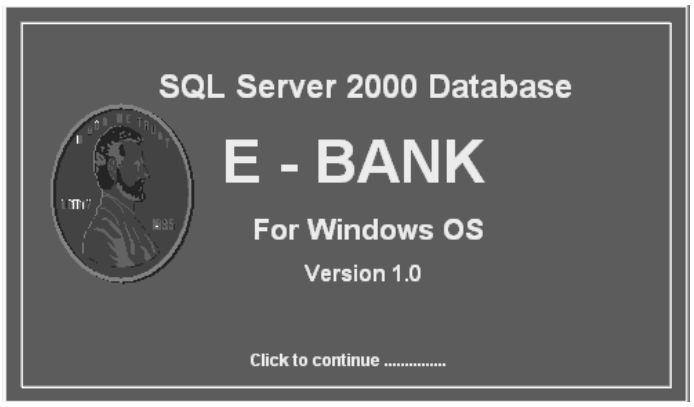


Figure (5.10): E-Bank Splash Screen

The Electronic Bank main window contains many buttons (Add new account, Delete account, Update account, Search for account, About and Exit). These buttons used for managing user accounts in the electronic bank. This window is appears as shown in the following figure.

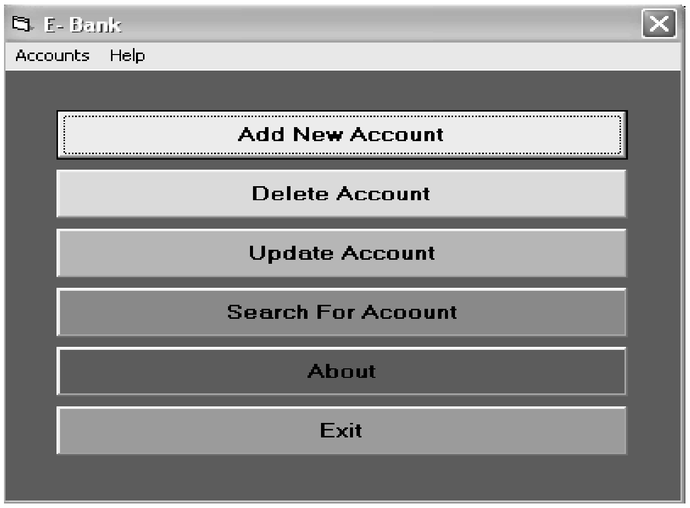


Figure (5.11):E-Bank main window Window

A- Add New Account

This button used for adding new record account. It display a window contains all information required for registering the user account. The window of the adding new record is as shown in the figure (5.12).

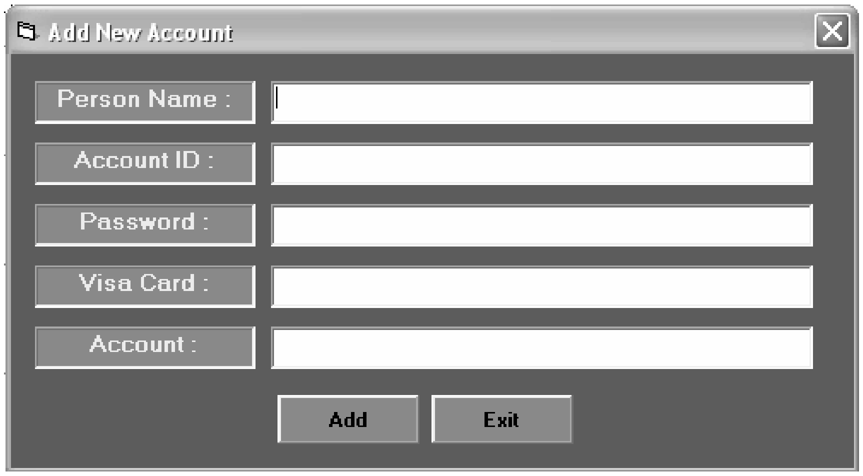


Figure (5.12):Add new account Window

B- Delete Account

This button used for delete record account. It display a window contains selecting name of the user account. The window of the deleting record is as shown in the figure (5.13).

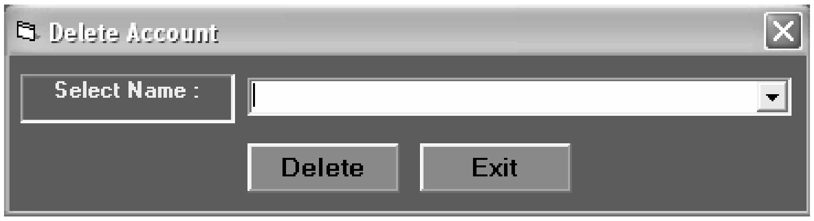


Figure (5.13):Deleting account Window

C- Update Account

This button used for update record account. It display a window contains selecting name of the user account for update. The window of the update record is as shown in the figure (5.14). This window also contains all database fields that can be changed if required to update them.

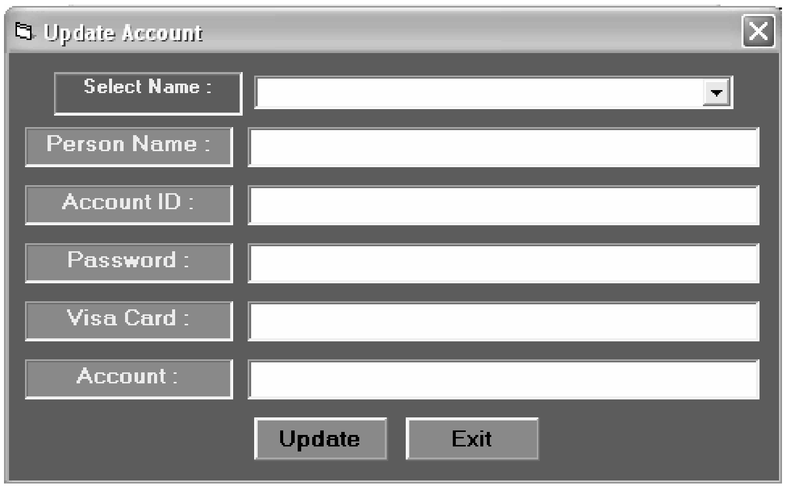


Figure (5.14):Update Window

D- Search for Account

This button used for Search for account. It display a window contains selecting name of the user account. The window of the searching record is as shown in the figure (5.15).

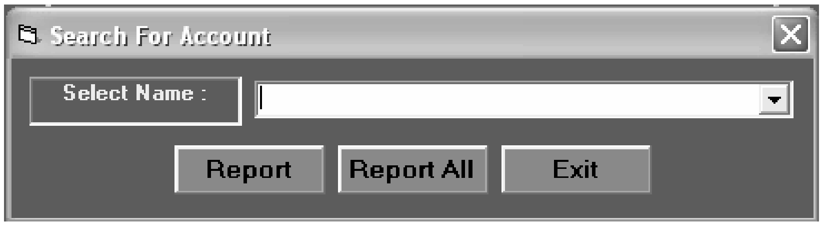
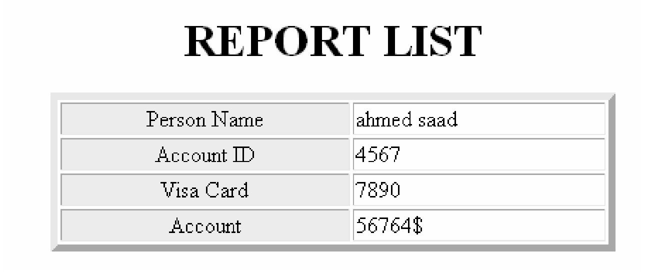


Figure (5.15):Search Window

The result of search can be displayed in details as HTML report and can be printer on papers directly. The report is as shown in figure (5.16).



REPORT LIST

Person Name	ahmed saad
Account ID	4567
Visa Card	7890
Account	56764$

Figure (5.16):Report Window

The other keys are for displaying information about this sub system and exit from program.

5.3.3 Connecting To FoxPro Database

Connecting to FoxPro database can be done through press (FoxPro – Phone Book) button from the main window of the proposed system. The window of this system is as shown . This window is called Splash screen it display information about this sub system. When click
any key the main window of this sub system will appears as shown in figure (5.17).

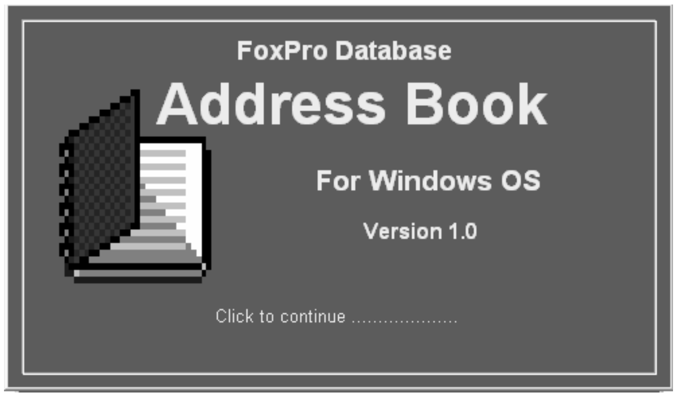


Figure (5.17):Phone Book Splash Screen

The Phone book main window contains many buttons. These buttons used for managing address book in the phone book. This window is appears as shown in figure (5.18).

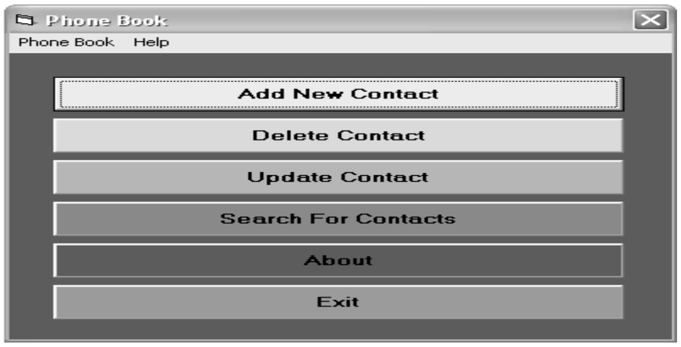


Figure (5.18): Phone Window

A- Add New Contact

This button used for adding new record contact. It display a window contains all information required for registering the user contact. The window of the adding new contact is as shown in the figure (5.19).

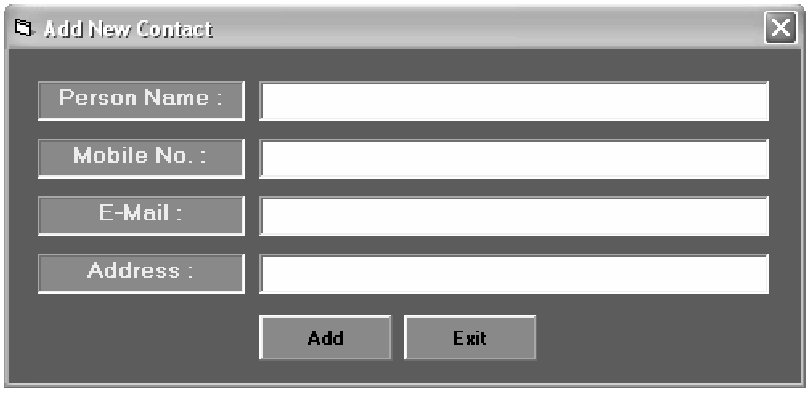


Figure (5.19):ADD new contact Window

B- Delete Contact

This button used for delete record account. It display a window contains selecting name of the user account. The window of the deleting record is as shown in the figure (5.20).

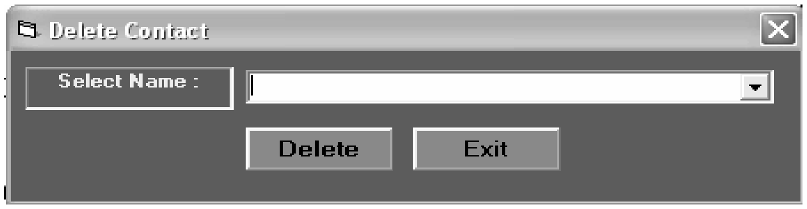


Figure (5.20):Deleting contact Window

C- Update Contact

This button used for update record contact. It display a window contains selecting name of the user contact for update. The window of the update record is as shown in the figure (5.21). This window also contains all database fields that can be changed if required to update them.

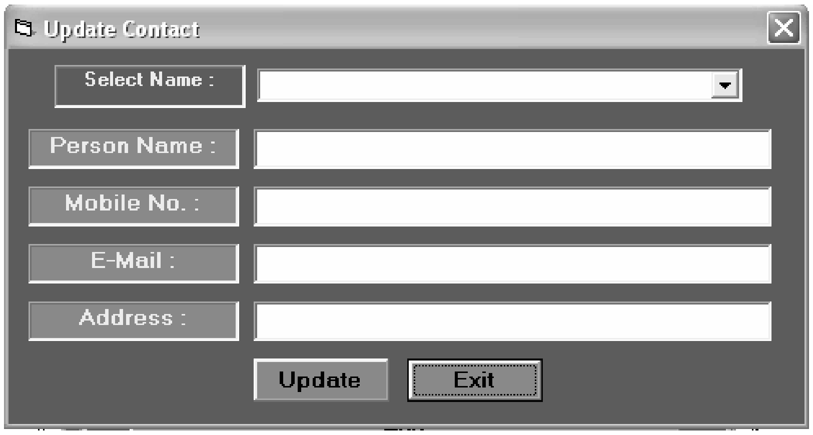


Figure (5.21):Update Window

D- Search for Account

This button used for Search for account. It display a window contains selecting name of the user account. The window of the searching record is as shown in the figure (5.22).

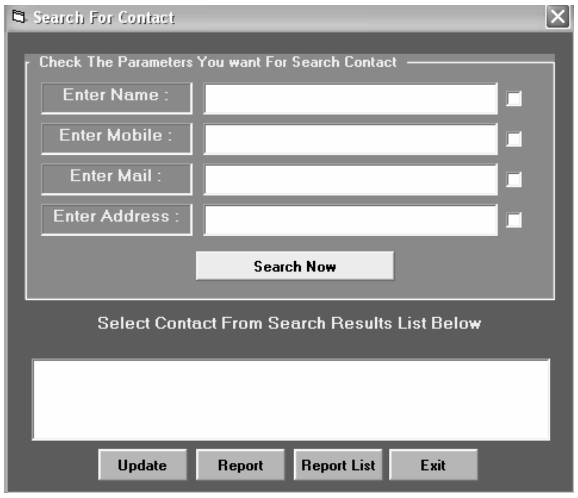
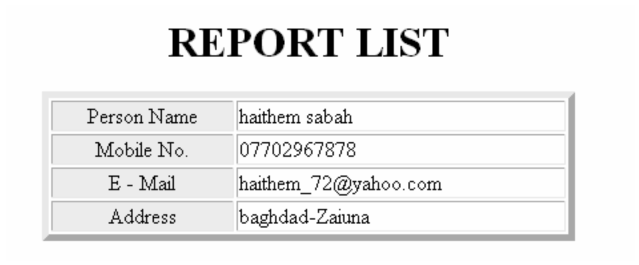


Figure (5.22):Search Window

The result of search can be displayed in details as HTML report and can be printer on papers directly. The report is as shown in figure(5.23).



REPORT LIST

Person Name	haithem sabah
Mobile No.	07702967878
E - Mail	haithem_72@yahoo.com
Address	baghdad-Zaiuna

Figure (5.6):Report Window

The other keys are for displaying information about this sub system and exit from program.

Chapter Six
Conclusions and Suggestions for Future Work

6.1Conclusions

The following conclusions are drawn from the design, implementation, and experimental results of the proposed system.

1. Retrieve distributed queries from client workstations.
2. Decompose the global queries to multiple local sub queries to be executed at each of the two sites.
3. Generate distribute the execution strategies for each site (this includes the global site that has to process and route the final result rows to the requesting client).
4. The system is providing almost advantages of DBMS, such as availability, reliability, distribution transparency…etc.
5. The proposed system build in bottom-up approach using different active X data control, related with two different databases (Access, FoxPro); the data control that uses in the Access Client design, as well as the Data environment is uses in the FoxPro Client design.
6. It is possible to apply the proposed system on the Wide Area Network (WAN) rather than Local Area Network (LAN).

6.2 Suggestions for Future Works

The following suggestions made for future work to enhance the proposed system:

1. The project design would have been even far better if somehow the Artificial Intelligence used in the project could be updated dynamically. The concept of having intelligent database or World Wide Web (WWW)

databases is growing in application with self-learning capacity, as data warehouse. Whenever a new source of information is detected; the intelligent database come into action and gather the new information.

REFERENCES

[Ami 04] Amir A.M. Al-Sameraee, "***Proposed Watermarking Technique for Distributed Database Transmission***", University Of Technology/ Institute for Postgraduate Studies in Informatics, Baghdad 2004.

[And 01] Andrei Lopatenko UM, Anne Asserson, UiB, Keith G Jeffery CLRC"*CERIF* " ***Information Retrieval of Research Information in a Distributed Heterogeneous Environment"*** CERIF TG, 2001.

[Ang 95] Angelo R.Bobak "***Distributed and Multidatabase System***",London,1995

[Aru 97] Arunk Majunder, "***Database Management System***", 1997.

[Den 02] Denise Ecklund, and Vera Goebel," ***Heterogeneous / Federated /Multi-Database Systems*"**, 9 October 2002.

[Ber 89] Bertion.E, G. pelagatti, and L.Sbattella"***An Object-Oriented Approach to the Interconnection of Heterogeneous Database"*** In Processing of the Workshop on Heterogeneous Database. NSF, December 1989.

[Gar 92] Gary W.Hansen, James V.Hansen, "***Database Management and Design"**,* Prentice Hall, Inc.Englewood Cliffs, New Jersey 07632,Book, USA (1992).

[Ham 80] M. Hammer and D McLeod.*"**On Database Mnagement***

***System Archiecture*”**. In Infoted State of the Art Report: Data

Design. Pergamon Infotech Limited, Maidenhead, United Kingdom, 1980.[M.Ta 2000] M.Tamer Özsu,”*Distributed*

Database”,www.pmg.lcs.mit.edu/~chmoh/pubs/DDB.pdf.2000, University of Alberta, 2000.

[Hen 91] Henry F.Korth, and Abraham Silberschatz, “***Database System Concept***”,University of Texas at Austin, McGraw-Hill, Inc, 1991.

[Hul 97] Hull, R. “***Managing Semantic Heterogeneity in Databases”:*** Theoretical Perspective, Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. 1997

[Joa 96] Joachim hammer, and dennis mcleod,” ***An Approach to Resolving Semantic Heterogeneous in a Federation of Autonomous,Heterogeneous Database Systems***”, Computer Science Department, University of Southern California. Los Angeles. CA 90089-0781, USA, (213)740-4504.1996

[Kam 01] Kamal Karlapalem,” ***Distributed Database System***”, Introduction IIIT (Indian Institute of Information Technology),Gachibowli, Hyderabad 500019, INDIA, 2001.

[Khu 04] Khurram Allah Ditta” ***Integrating Information from Heterogeneous Distributed Databases”,*** MSc Information Systems,School of Computing, University of Leeds MSC Project

Objectives and Deliverables, (2004).

[Nin 03] Nina Dzamashvili, Jellte Jansons, "***Physical modeling, distributed of data***". BLEKINGE TEKNIKA HÖGSKLA (BHT), May 2003.

[Nor 00] Norio Katayama, Masanori Sugimoto, and Jun Adachi "***A Universal Query Interface for Heterogeneous Distributed Digital Libraries***" NACSIS (National Center for Science Information Systems) 2000.

[Orf 96] Orfali.R, Harkey.D, and J. Edwards, "***Essential Client/Server Survival Guide", 2nd edition***. Wiley, 1996.

[Ozs 06] Tamer Özsu .M & Patrick Valduriez "***Principles of Distributed DataBase System"***,2nd Edition,India,2006

[Pao 99] Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, and Riccardo Torlone, "***Database Systems Concepts, Languages and Architectures"***, McGraw-Hill, 1999.

[Ram 00] Ramez Elmasri, Shamkant B. Navaathe," ***Fundamentals of Database Systems***", Addison-Wesley, 2nd edition, USA 2003.

[Rag 00] Raghu Ramakrishnan "***Introduction to Database Systems***", Um- Madison,(2000).

[Red 94] Reddy, M. et al.. "***A Methodology for Integration of Heterogeneous Databases",*** IEEE Transactions on Knowledge and Data Engineering, Vol. 6, No 6, Available at: http://portal.acm.org/portal.cfm, (1994).

[Rob 88] Robert Ascher Maum, "***Network Essentials For***

Dummies “, Los Angeles 1988.

[Sil 02] Silberschatz Abraham, Korth Henry, F. and S”***Database system concept***", 4th Edition, Mc Graw Hill, 2002.

[Sey 98] Seydim A. Y., "***An Overview of Distributed Database Management***", www.pmg.lcs.mit.edu/papers /DDBM.pdf,1998 .

[Ste 85] Stefano. Ceri, “***Distributed Database Principles and System***”, McGraw-Hill Book Company, 3rd Printing, 1985.

[Ste 00] Stefan Leue, “***Distributed Systems***”,
tele Research Group for Computer Networks and
Telecommunications,
http://www.informatik.uni- freiburg.de/~leue Winter Term
2000.

[Tap 03] Tapio Niemi, Marko Niinim¨aki, and Vesa Sivunen,” ***Integrating
Distributed Heterogeneous Databases and Distributed Grid
Computing”*** Helsinki Institute of Physics, CERN Offices CH-
1211 Gen`eve, Switzerland 2003.

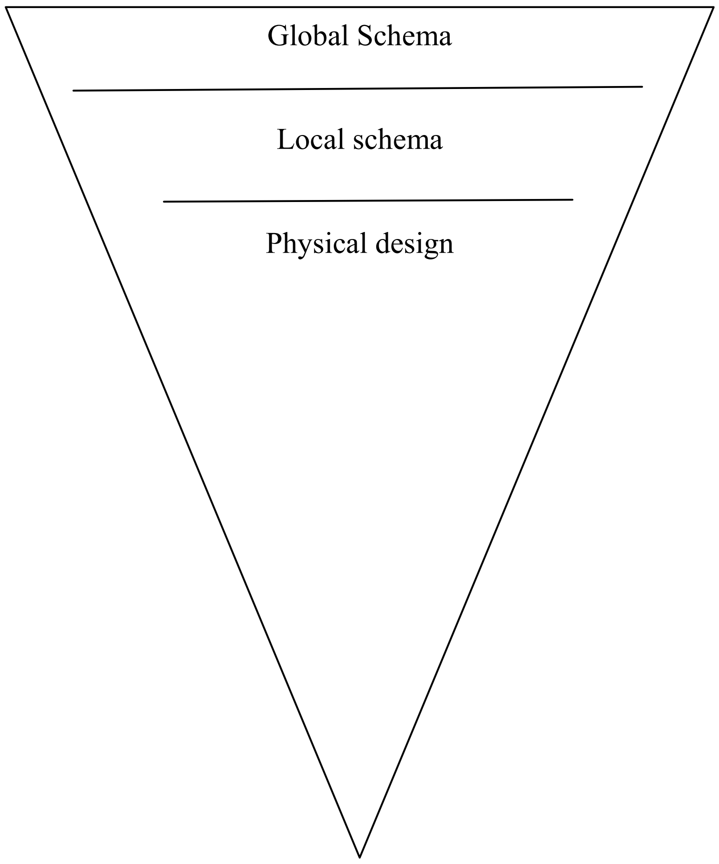
[Tam 98] Tamer Özsu & Patrick Valduriez, “***Distributed DBMS***”,
University of Alberta, 1998 .

[Xue 04] Xuequn. Wu ,and Schmelzer R. “***A CORBA-Based Architecture
for Integrating Distributed and Heterogeneous Databases”***.
RDF/XML Syntax Specification (Revised) (2004).

[Zho 01] Zhool W.,”***Distributed Database System***”,

www.cs.adfa.oz.au/teaching/studinfo/dsd/DistributedDBS.pdf,
2001.

a. Dependability - when the server goes down, operations cease
b. Lack of mature tools - it is a relatively new technology and needed tools are lacking
    i. e.g.. Automated client software distribution
c. Lack of scalability - network operating systems (e.g.. Novell Netware, Windows NT Server) are not very scalable.
d. Higher than anticipated costs
e. Can cause network congestion



ملخص البحث

أن قواعد البيانات المعتمدة تتكون من مجموعة مستقلة من قواعد البيانات
أن الانظمة التي تستخدم لتدير قواعد البيانات المعتمدة تسمى انظمة قواعد البيانات المتعددة في هذه الانظمة المعالجات تتعامل وتنفذ تحت سيطرة نظام قاعدة البيانات المتعدد. ان المعالجات المحلية غير المعتمدة ترسل بشكل مباشر ومستقل الى انظمة قواعد بيانات محلية من خلال تطبيقات محلية لذلك ان انظمة قواعد البيانات المتعدد يجب ان توفر ميكانيكية للمعالجات العامة وعليه مثل هكذا تعاملات تتطلب حياة طويلة لقواعد بيانات محلية مستقلة علاوة على ذلك ان انظمة قواعد البيانات المتعددة لا تحوي اي معلومات حول وجود وتنفيذ اوامر التعاملات المحلية .

الهدف من هذا البحث تقديم مقترح للتصميم الذي يقوم لدمج ومعالجة ثلاثة مصادر من قواعد البيانات غير المتجانسة من نفس النموذج (العلائقي) تحت بيئة برمجية مختلفة . هذا بالاضافة الى دراسة موضوع المشروع الذي يهدف الى :

- تطوير فهم كيفية الاستفسار عن المصدر المعلومات الغيرمتجانسة المنفذة في بيئة موزعة
- تحصيل النتائج من خلال مترجم متخصص لفلترت ودمج المعلومات وارجاع الاجوبة النهائية للمستخدم او تطبيقات الزبون .
- تتم عمليات (حذف وتحديث وبحث) عن طريق الحاسبه الخادم (server)

الاخرى فتتولى كل منهم عمليات التحديث والاضافه (client)اما حاسبات الزبائن والبحث الخاص بها.

لقد نفذهذا المشروع على حاسبه (PENTIUM4) ذات سرعه(1.8) MHZ وذاكره (128 MB)خزن بنظام تشغيل XP حيث تم تصميم النظام باستخدام لغه البرمجه فيجوال بيسك (VISUAL BASIC 6.0)وقواعد البيانات تم بنائها باستخدام (MICROSOFT FOXPRO)و (MICROSOFT ACCESS 2000) و (SQLSERVER) من خلال شبكه من (LAN)