Query Processing for Data Retrieval from Distributed Database Management System

Hayder Mosa Merza Al-Rubaiy

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
February 2014
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Prof. Dr. Nazim Mahmudov
Chair, Department of Mathematics

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Assoc. Prof. Dr. Rashad Aliyev
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Rashad Aliyev
2. Asst. Prof. Dr. Ersin Kuset Bodur
3. Asst. Prof. Dr. Mehmet Ali Tut

ABSTRACT

The goal of this thesis is to analyze an importance of a distributed database management system for data retrieval process. The characteristics of distributed database management systems are defined. The homogeneous and heterogeneous distributed database management systems are presented. The main objectives of data replication and data allocation are highlighted, and data fragmentation by query processing is implemented. The relational algebra operations and Structured Query Language (SQL) statements are applied to determine different types of data fragmentation.

Keywords: Distributed Database Management System, Homogeneous and heteregeneous distributed database management systems, Data replication, Data allocation, Data fragmentation, Query processing, Strucured Query Language (SQL)

ÖZ

Bu tezin amacı veri erişimi işlemi için dağıtımlı veritabanı yönetim sisteminin önemini araştırmaktır. Dağıtımlı veritabanı yönetim sisteminin özellikleri tanımlanır. Homojen ve heterojen dağıtımlı veritabanı yönetim sistemleri gösterilir. Veri çoğaltma ve veri tahsisi vurgulanır, ve sorgu işlemleri kullanmakla veri parçalanması hayata geçirilir. İlişkisel cebir operatörleri ve Yapısal Sorgulama Dili (SQL) komutları uygulamakla veri parçalanmasının farklı türleri belirlenir.

Anahtar Kelimeler: Dağıtımlı Veritabanı Yönetim Sistemi, Homojen ve Heterojen Dağıtımlı Veritabanı Yönetim Sistemleri, Veri çoğaltma, Veri tahsisi, Veri parçalanması, Veri sorgulama, SQL Yapısal Sorgulama Dili

ACKNOWLEDGMENTS

I would like to express my special appreciation and thanks to my supervisor Assoc. Prof. Dr. Rashad Aliyev for his continuous support at all the stages of this thesis. I would also like to thank him for being an open person to ideas and for helping and encouraging me.

Special thanks to my government represented by Iraqi Council of Representative and especially to Presidency Commission, the Expert Mr. Sarchel Lawrani, Chief of Staff Mr. Iyad Al-Hajj Namak, members of Subject Committee represented by Dr. Firas Al-Hussainy and my manager Mr. Majid Khedir Ahmed.

I would like to express appreciation to the members of my family - my beloved wife Zina who spent sleepless nights and supported me whenever there was nobody to answer my questions, and my children Narjis and Mohammed Baqer. My sincere thanks also go to my mother, and it is very difficult to express verbally how grateful I am to her, and her prayers have been always giving me power in all the areas of my life. I would like to thank the rest of my family, my brothers, sisters, all my relatives and my closest friends.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

In recent times, databases reached a significant increase in the proportion of the volume of data, and one of the big reasons for this increase is due to many factors in a wide range of businesses around the world, in addition to the use of electronic trading which will be very fast and reliable.

The large volume of business dealings led to the necessity of providing a reliable source to be concentrated in one location. The broad scope of the size of these transactions led to the distribution of the data on several different sites which displayed database management systems to many of the problems, especially in interoperability and data duplication, and so the attention must focus on the performance of database systems by distributing more lively database technology with careful attention to the user through the performance evaluation models of the factors that adversely affect the performance of database systems.

Management of distributed databases is of great significance in the audit and provides enough time for users that are posing a major concern for developers of database systems as well as in the implementation of database management systems in terms of the architectural side of these regimes.

Database management systems provide a multi environment for storing and distributing data on several sites instead of the original environment which is controlled by centralization of these systems that store data as a form with one copy.

Most problems in the database systems are shared in the management of transactions and access control. Therefore it should be easy to find solutions for these problems by using the distributing system such that the retrieved data will be updated and be available for users. In other words, the multiple rules must be available for the detection and updating data within the systems.

Distributed DBMS (DDBMS) has some advantages over classical DBMS. The advantages are:

- DDBMS permits a lot of users to access and process a big number of data at the same time, i.e. the data retrieval and manipulation process in DDBMS are executed in parallel form;

- The organizational structure of distributed database systems is more proportional than central databases systems that are geographically dispersed;

- When any failure in the system occurs, it does not lead to a stop work order or invalidity, on the contrary - makes it more reliable;

- It is possible to obtain data and to enter the networks taking a copy of data in each site by stopping one of servers the system;

- It is easy to expand the distributed database management systems through the addition of a new location without affecting the list.

In the process of using DDBMS some disadvantages should be overcome. They are given below:

- Distributed database systems are more expensive and complicated than the regulations of the central character;

- To confirm the forcefulness of the system, the authenticity of the software and hardware should be provided through development of system security and controlling the user input;

- Most processes that happen through the data must be clear for user like detection and retrieval of data over the sites;

- Maintaining the system requires availability of experts and technicians at each location, and it leads to an increase in cost;

- The security aspect of most complex aspects plagues distributed database systems in all the locations. Moreover, the communication that takes place between the sites can be exploited to disclose.

There are two types of distributed database management systems: homogeneous DDBMS and heterogeneous DDBMS. The homogenous DDBMS is easily designed and managed, and all the sites use the same components of database management systems. In heterogeneous DDBMS there is a possibility of running of most system applications over the sites. These sites have some rules which are programmed by administrators.

Chapter 2

REVIEW OF EXISTING LITERATURE ON DISTRIBUTED DATABASE MANAGEMENT SYSTEMS AND THEIR APPLICATIONS

One of the essential components we are facing in our daily life is database management system to be used to get the relevant data that everyone needs, and the benefit from the services and resources are provided from the libraries that play the important role in our modern society. To have more effect from the using of available database management systems, and taking into account that information is managed by many systems, distribution of data in these systems becomes very actual problem. [1] considers the technology that can be successfully used to access and to manage the available distributed information.

In [2] grid and distributed database approaches for data replication are considered. The main purpose of the research is to find the common properties of both approaches in order to get the most efficient outcome from their combination. The proposed object-oriented database management system is very important for some High Energy Physics experiments. The characteristics and needs of data grid are obtained.

In [3] the importance of additional coordination of diverse computerized operation for different organizations is discussed. It must contain database system that can

work over an allocation network and can include multiple computers operating system, communication relation and local database management systems. The approaches for description of characteristics and architectures of some heterogeneous distributed database management systems are presented.

The problem of a coordination of the sites of a communication network and data allocation of a database is discussed in [4]. The benefits of proposed two allocation management problems are considered. The cost of an allocation is computed by applying optimal heuristic solutions.

There are some uncertainties occurring in organization of the information infrastructure of modern organizations. In order to reduce the uncertainties, the DDBMS are applied for business applications, and their feasibility is determined [5]. It is mentioned that the distributed databases provide advantages for organizations in terms of obtaining the necessary meaningful information.

Increasing needs of current University environment make it inevitable to apply distributed database systems for accessing reliable and scalable information. In [6] proposed relational database system is designed in a way that each department of the University includes its own database for specific needs. The Structured Query Language (SQL) is used to design a client server distributed database to process the student records effectively.

There are many optimization algorithms used in distributed queries, and as a result of the research conducted for the optimization of the query processing, a new algorithm is developed that enables it to reduce the amount of intermediate data and the cost of the network communication which significantly optimizes the efficiency of the distributed query [7].

[8] focuses on methodological approaches of the design process of distributed database. It is noted that the design can be either from top to bottom or vice versa, and the first of these approaches is typical to distributed system developed from the scratch, and the second one is typical to multi-database as the aggregation of existing databases. The case study is considered to resolve the design problems for both approaches.

Object sharing in distributed DBMS is an actual issue, and the Distributed Object Database Model (DODM) proposed in [9] provides some operations to define, to manipulate, and to retrieve objects. A small set of operations is introduced to develop complex database systems. The provided example illustrates the importance of DODM in the design process of the distributed database system.

[10] discusses multilevel security for a distributed database management system functioning in a heterogeneous environment. The structure of the system and query processing techniques are investigated.

In [11] a multilevel secure distributed database management system is described, and the users have access to the system at different levels. The security levels are assigned to the data. The techniques intended for security constraints processing in a centralized multilevel database are important for querying, and updating of information.

Military applications require an efficient processing of the distributed system, and the system should operate with a high level of security. [12] focuses on the security of query processing in a distributed DBMS. The algorithm for secure processing of queries is implemented.

Since many factors may affect the results of simulating distributed database management system, it is a difficult process to manage such system. In [13] proposed DBsim simulator architecture of a distributed system can be extended that enables to change the parameter and configuration of the system. The offered two concurrency control algorithms are compared in terms of performance and response times. The results show that using long transactions because of their high abort rates leads to non-identical performance of two schedulers.

To maintain the integrity of the distributed database management systems, in [14] a new module CDAI (Consistent Data Access with Integrity) is presented. It provides users to access data in a reasonable time period. The proposed algorithms show effective functional architecture and high level description.

In the process of early design of database management system such activities as documentation and training associated with administering the system must be specified. The database management system performing administrative tasks and functioning in a distributed environment is discussed in [15]. A general framework for administering a distributed database management system and a list of administrative functions are discussed. The advantage of the system is an ability of being extended to other types of distributed database management system.

One of the main principles of most systems applied to distributed database management system is site autonomy. The organizations using distributed database technology consider administrative issues as very important. In [16] the compatibility between replication transparency and site autonomy in distributed database management systems is analyzed. It is suggested that the decision of system administrator should be up to the degree of site autonomy. A mechanism intended for extension of the ANSI SQL authorization model is implemented for both centralized and decentralized administration policies of the users.

In [17] a formal model and a modeling approach for distributed database management system are offered. A general purpose system simulation (GPSS) for modeling is discussed. Two - phase locking in distributed database system is executed for data replication. The centralized, primary copy, distributed, and voting two-phase locking are suggested. The simulation of the distributed two-phase locking is carried out.

[18] illustrates the query execution strategy for distributed database management systems to enable users to execute queries in databases which are series of operations performed on different nodes of the network and transmission. The evaluation of this strategy depends on the validity of the results and their conformity system using multi-relations algebraic within the framework of a unified mode.

A decentralized control of user accesses to a distributed database is suggested in [19]. The distributed copy of a single database is considered. It is mentioned that the predicate/transition nets are very appropriate for representation of decentralized systems. This distributed database system is deadlock-free, and provides effective service to users.

In [20] the basic concepts of distributed database system including transaction management process are given. While accessing distributed database several problems occur. In order to perform transaction management and to monitor data different mechanisms of distributed DBMS are required. The transaction management process is explained. Two-phase and three-phase commit protocols are executed.

[21] describes distributed transaction model for multi-database management systems. The differences between distributed transaction and other distributed processing systems are presented. The failures of site, network and time in distribution of transactions are highlighted. The proposed transaction model is used to process

transaction queries in multi-database management system in order to show that a complete database can be designed at any stage.

In [22] the features of the distributed database architecture are considered. The design of such system improves consistency, accessibility and flexibility in the process of accessing different types of data. The security of the system is reached by access control and integrity of the system. The confidentiality and reliability of a distributed database system are discussed. It is underlined that efficient communicating and reasonable price of such system is especially effective in both military and commercial applications. The partial security of a distributed database management system is mentioned.

While dealing with distributed database management system the security features should be taken into account. In [23] the security features of both object oriented and relational data model are discussed. The single and multilevel access controls play an important role in choosing one of the models. The strengths and weaknesses of both models are represented.

The design process of a distributed relational database system is suggested in [24]. The experiments for obtaining the performance of the system are carried out by executing short and long commands. The query processing in distributed data base system is performed to minimize metrics.

In [25] the distributed database optimization problem is considered which is very important and difficult problem in designing of client/server distributed system. The technique called ARRQ is proposed to process queries in optimal form, and namely, this technique help the users to find out which fragments should be partitioned into fragments. The advantages and efficiency of the proposed technique over other techniques are presented that lead to reducing of communication and local processing costs by the improvement of the response time of the queries.

Data or fragments allocation in distributed systems requires a lot of efforts. The quality, performance and operational efficiency of the distributed system are strictly up to the allocation process. All these factors stipulate the application of an appropriate technique for data allocation. A new dynamic data allocation algorithm for non-replicated distributed database system is offered in [26]. The algorithm is very suitable for distributed systems with low bandwidth and frequent requests.

Chapter 3

CHARACTERISTICS OF DATABASE MANAGEMENT SYSTEMS AND DISTRIBUTED DATABASE MANAGEMENT SYSTEMS

3.1 Differences between file based system and database management system

The file system is defined as a method for organizing data in the system computers. In general, the file system contains folders, files or special types of files. The file system has a limitation in storing and organizing files in the computer logically. Actually, the computers store data as strings of bits, so data do not have any structure just being collection of 0’s and 1’s.

Usually the file systems are using tables to store the information in a limited area on the device, and every file is placed in this table with its length, location and other facilities. The operating system should access and hold some functions in this table such as editing, deleting, renaming, or moving any of attributes.

As mentioned above, file systems have different types. Every file has different logic and structure. While dealing with organizing and management of data in the file system, there are some causes making us feeling problems with this system. Below some of these problems are given:

1. Data redundancy: The duplication of the same data in multi files;

2. Data inconsistency: The contradiction of the data, in other words, number of different copies of the same data but not matching. It happens as a result of updating process of data in one file, and not updating the same information in another file;

3. Difficulty in accessing data: It is not easy and sometimes impossible recovering data by processing traditional file system;

4. Data isolation: Writing processing of a new program to advice new application is very difficult, because the data are dispersed in different files and may be written in different formats;

5. Integrity problems: There are some constrains with the value of data, and it is necessary to accept the integrity of these constrains;

6. Atomicity problems: File processing system is not strong with atomicity, especially with the transformation operation between two or more places; it may be satisfied in one place and not in another one;

7. Concurrent access anomalies: This problem explains the difficulty of updating same information by many of users;

8. Security problems: This problem is very sensitive in file systems, because file systems are protected very weakly from the attacks of hackers.

File systems use files for saving information, whereas the databases are applied for saving data in DBMS. Despite the file system and DBMS are two methods for working with data it must be noted that DBMS has more advantages over the file system. In file system, most of operations like retrieval, storage and searching are done manually which is very boring process, while DBMS makes these tasks by supporting atomic ways for completing. Taking above into consideration it is to say that such problems as data integration, data contradiction and data security can be overcome applying DBMS.

Database management system is a number of network programs of computers which concern the management of data and holding a processing like retrieval, storage and organization. There are many different kinds of database management systems, and some of them are designed for special needs and some for commercial business. Any DBMS has four important elements:

1. Modeling language: It defines the language of every database environment in DBMS;

2. Data structure: It helps organizing data like single records, files, fields and the objects with their definitions;

3. Query of languages: It gives the maintaining and more security to data, and allows many users to access the system, to add or to modify data. Structured Query

Language (SQL) is considered the most famous type for this purpose that is used in relational database system;

4. Mechanism for transactions: It ensures integrity of data.

Primarily, DBMS reveals extraordinarily complicated relationships between bound items of information. Several DBMSs will be able to place information in an exceedingly method that reveals patterns to be troublesome or not possible to identify exploitation.

One of the benefits of DBMS is that it reduces the danger of human error and may even be programmed so that human error is just about not possible. DBMS will mechanically make a copy of data so the information will never be lost or destroyed.

In addition, DBMS gives all the facilities of data with backup that keeping the database from lost. DBMS is a package of applications that set up to managing data and uses many of functions to make the processing operation over the data, whereas the file system is a group of clean data that are stored in any device, such as CD or hard-drive. Both of these systems are provided with the permissions to the user for working with data in a same way. File system is closer for managing data, but it has some reduction for especially electronic data.

3.2 Distributed Processing System. Characteristics of Distributed Database Management Systems

Actually there is no precise definition of a distributed processing. The elements of distributed processing are interconnected by a computer network and performing their tasks, and intended to cope the large-scale complicated data management problems [27].

Distributed processing of data is very popular, and has many examples such as networks of telecommunication, cellular and telephone networks, internet networks, and wireless networks.

There are many advantages of the distributed processing. Some of them are: organization and easy communion; supporting availability and reliability; high level performance and fill of self location; improved time responsibility speed that means the system response is given just after the users entry queries; low cost and long communication with lower rate; distributed process can minimize volume of information that should be transmitted along the area; integration of data which is improved by giving the users controlling over the data with good degrees of correctness and accuracy; sharing the resources between multi computers.

Some disadvantages of distributed processing should be also taken into account. They are given below:

- Complexity of the system: Different extra software is needed to be installed in distributed systems;

- Network failure: Because the distributed system is connected over the network, system may not work after network failures;

- Side of security: Because the information crosses over the network, it might be displayed to unauthorized users or hackers to be used illegal.

Distributed systems are mainly characterized by three properties: concurrency of components, lack of a global clock, and independent failure of components.

In distributed system data is distributed over a multi place on the contrary with centralized database in which data is placed in a single physical location, i.e. in distributed system all data are held on multiple computers interconnected over a computer network, but in centralized database all data are held in a central computer that can be easily accessed and bucked up.

What kind of advantages does a distributed database over a centralized database have? The major characteristics of a distributed database are high reliability, possibility of modular growth, and ability of fast response time for most queries of users.

The figures 1 and 2 depict the structure of a centralized database and a distributed database, respectively.

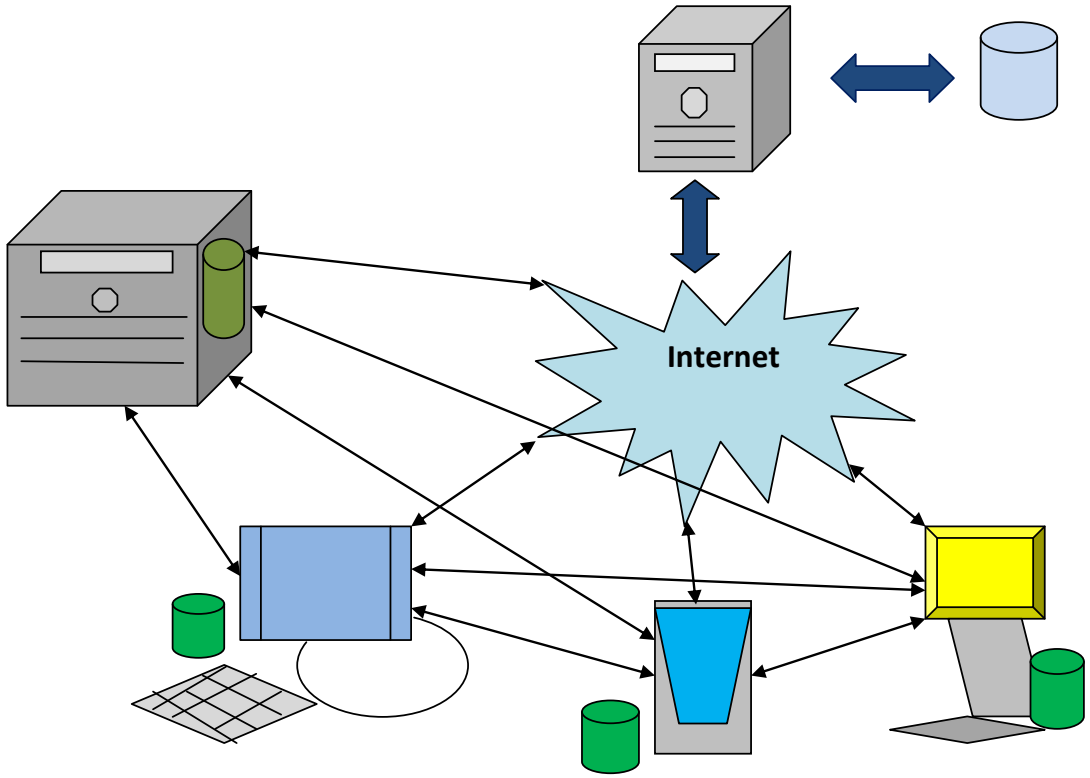

Figure 1. Structure of centralized database

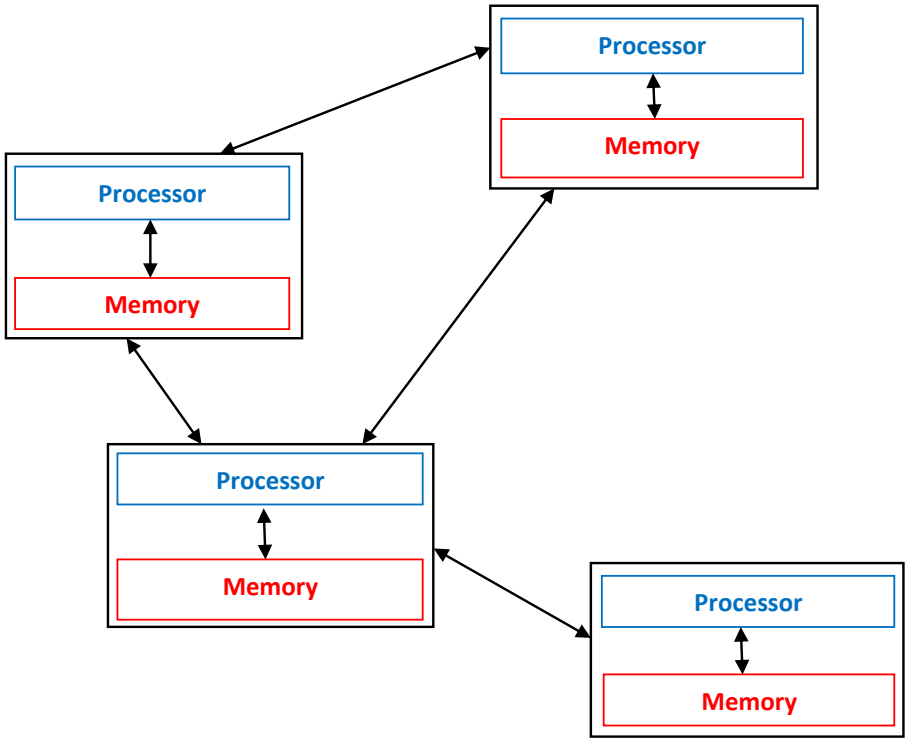

Figure 2. Structure of distributed database

The distributed system refers to a network of computers which are allocated physically independent with each other at geographical places. The users are solving the problems with individual way by receiving the messages that pass through the service of communication.

DDBMS has number of characteristics represented below:

- A common variety of logically related data;

- The data is divided into number of fragments;

- The possibility of working replicas of the fragment;

- The fragments replicated are allocated to the sites;

- Linking sites by the communications network;

- DDBMS is dealing with local application at each site.

In DDBMS it is not necessary that each site of the system will have its own database. The topology of distributed database management system is depicted in Figure 3 [28].

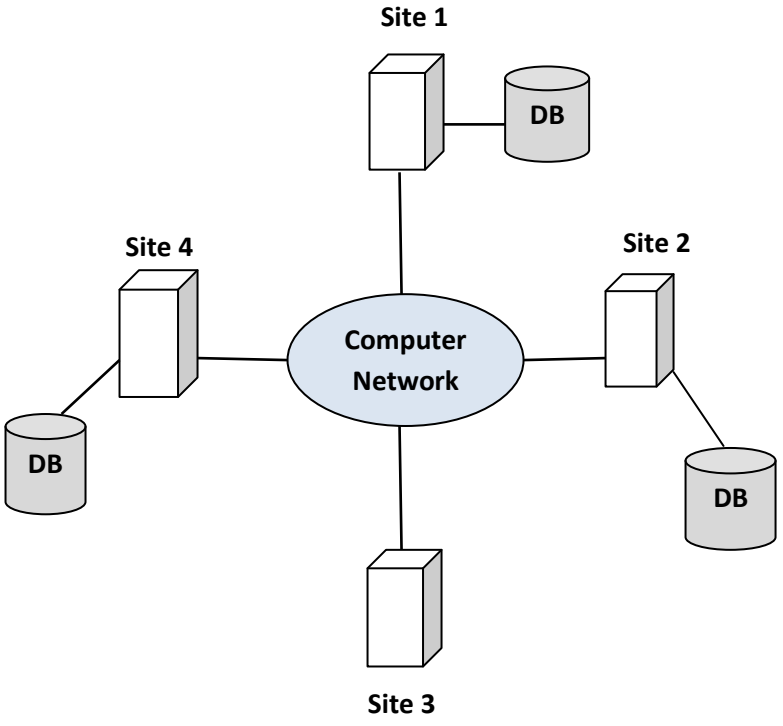

Figure 3. Topology of distributed database management system

3.3 Homogeneous and Heterogeneous Distributed Database Management Systems

The frequent use of database systems in a wide range of areas and size increasing of data in these systems led to the division of the distributed databases into two broad groups classified as homogeneous and heterogeneous distributed database systems. The main differences between these two types of distributed database management systems are that homogenous database system is considered as a single entire system, all the sites use the same software and agree about cooperation with each other in order to optimally process user requests, whereas in heterogeneous database system the different sites have different software, sites may not cooperate with each other, and query processing is a complex problem that leads to the difficulty of managing.

3.3.1 Homogeneous Database Management System

The homogeneous database management systems distinguishes from the heterogeneous type by simplicity and easiness of design, and handling making the process of adding new sites to the distribution system more effectively that lead to increasing of the proportion of savings, as well as taking advantages of these systems capabilities of parallel processing of several sites which improve the performance of systems, and each site is aware of working of other sites in the system.

A network of two or more databases is homogeneous that are available on a single device or multiple devices. For example, the distribution system linking some databases can be applied to access the system at the same time and then conducts operations to edit or to add data to a number of databases within one distribution environment.

Employing a homogeneous distributed database environment and phases of the system in which the implementation and maintenance of order, where they can create synonyms for objects in the system in order to make users easily to process and access.

The operating systems, data structures and database application which are used at each location of the homogeneous distributed database management system must be compatible [29].

Homogeneous distributed database systems can be classified as autonomous and non-autonomous systems. In autonomous systems all DBMSs are independent, but in non-autonomous systems all DBMSs are coordinated. In autonomous homogeneous distributes system each site has some degree of autonomy.

3.3.2 Heterogeneous Database Management System

Structure of a heterogeneous distributed database management system contains four levels: the local schema, the local object schema, the global schema and the global view schema, and all four layers are included in integration model.

Before discussing about heterogeneous DDBMS we need to explain the meaning of the term “object”. An object is represented by its three important components which are identifier, current state, and interface. Interface is intended to define object’s behavior.

3.3.2.1 Integration Model

Each layer in a heterogeneous distributed database management system represents the integration view:

- Local Schema.

This layer composes collection of local information system schema, and these schemata supply the case of data that is stored in the models, and data stored can be recovered by using query languages.

- Local Object Schema.

Every local schema is constructing only one local object schema, and this construction implies the identification which explains the different kind for every object in the schema. An object of database is unique and each object consists of a set of unique properties, and these properties are associated with each object.

- Global Schema.

The global schema can't exist unless local schemata are available. Objects in the component schema are brought together and then decomposed into objects of equivalence classes. Every object in the group gives only one type of global object.

- View Object Schema.

Each global object is generalized for the representation of the global view called the component of the view object. Computation of the union of the properties of the component object leads to a global view object.

3.3.2.2 Requirements for Heterogeneous Database Management System

There are some requirements to heterogeneous distributed database systems, and they are given below:

- Modeling of recovery, transactions and dist transparency;

- Stored data and heterogeneous integration;

- Heterogeneous collection of data through queries;

- Constructing of heterogeneous facts;

- Outrun of data repeat;

- Intromission transparency of language;

- "Open system" provides support for the integration of database systems models, in addition to the schemes;

- Constraints: maintaining the integrity of databases systems independently; moving away from a modification of effective applications; provides universal model in favor of the data compared with global applications.

It must be noticed that it is very important in heterogeneous systems that the physical distribution and heterogeneity of data are independent between themselves. The

heterogeneity can occur at each level of the database. For example, different sites can be used to write the applications of database with different languages, queries, models, DBMSs, file systems etc.

In terms of the practical side, if the following facilities are available, heterogeneity can be used fully:

- Manipulation of the database for users with no concern about the distribution of data and variety of local regulation;

- Supporting distributed databases by providing the system easily dealing with different languages;

- Possibility of integration the database in the distribution database management system without any editing or reorganization of the applications.

3.3.2.3 Schema Conflicts

There are some conflicts that may happen with schema during working with processing of data on database:

- Name: Writing or using different names make the system bad especially while dealing with equal entities, such as relationship, attributes etc;

- Structure: Losing some of attributes or tacit attributes cause problems to the system;

- Relationship: Multiplicity of such relationships in the database as one-to-one, many-to-many, one-to-many, and many-to-one makes conflicts in the system;

- Entity versus attributes: Some attributes depend on the key of a database, and the attribute plays a role of a key in a database.

- Behavior: Database system contains many different constraints that are used by users, for example, automatic system updating, deleting process etc.

In order to claim that a system is heterogeneous, the following steps should be followed:

- Connecting to the national network by using the system to the local networks;

- The Data Manipulation Languages (DML) and the Data Description Languages (DDL);

- Every function and DBMS guarantee: allocation of resources, protection, transaction management, and synchronization.

It is necessary to consider some constraints characteristic to heterogeneous distributed database system:

- Maintaining the integrity of databases systems independently;

- Moving away from a modification of effective applications;

- Provide universal model in favor of the data compared with global applications.

3.3.3 Multi-database Heterogeneous System

This system contains three major problems related with the optimization of the query for these systems. These problems are: heterogeneous cost modeling, heterogeneous query optimization and adaptive query processing.

3.3.3.1 Heterogeneous Cost Modeling

In this problem it is necessary to limit the cost of levels in the execution process of a query. In order to limit the executing cost of queries at DBMSs, three approaches should be used:

- Black Box Approach: This approach is working with the components of DBMSs as black box, and by activating some queries the cost of information is limited;

- Customized Approach: This approach uses some characteristics as external methods after using the first acknowledge for the component of DBMS;

- Dynamic Approach: Controlling process of a run time of a query conducted in the component DBMS is used.

3.3.3.2 Heterogeneous Query Optimization

It's very important to deal with the relationship between heterogeneous query optimization and the heterogeneous capabilities of the components of DBMS. For example, it may be one of the DBMS components supporting just one simple kind of

selected operation, whereas another one will support the complex of the same selection.

Dealing with two major approaches depends on the type of monitoring between the pander and the wrapper: query based and operator based.

- Query based: This approach is about the same language or query of capability, such as a set of SQL (Structured Query Language) which is transferred to the component of DBMS;

- Operator based: The wrappers use the relational operator to send the capability of DBMS's component via the composition of this relational operator.

3.3.3.3 Adaptive Query Processing

There is a very strong relationship between multi-database systems and the principles of processing of a query. The main relationship assumes that the optimizer of query multi-database systems has sufficient information about the condition of runtime of the query, but this assumption is accepted for systems with a little data in a small environment. On the other hand, this type of approach is inappropriate to the system which has a big size of data and wide environment.

Chapter 4

DATA REPLICATION. DATA ALLOCATION. DATA FRAGMENTATION

4.1 Data Replication

In a distributed database management systems users are geographically distributed far from each other, and the data replication becomes very effective for the access to a database. Database replication is a popular technique used in a lot of database systems, and users share identical level of information so that the benefit of using of a data replication consists in the fact that each user can access his/her own copy of a database without interfering works of other users, and the change made in one database is reflected in all others.

Data replication is realized in three different ways [30]:

- Snapshot replication. Data in the original database is updated and copied to another database located on the same server;

- Merging replication. There are two or more databases which are combined into one database;

- Transactional replication. All the users are provided with full version of the initial copy of a database and can get changes if any update in the database is done.

In a data replication any additions and deletions of data in any reversal site data are automatically stored in the rest of the other sites, and therefore, all users see the same data.

There are some benefits of a data replication. Let's consider the benefits of a data replication in a detailed form [31].

- Availability: One of the most effective ways to avoid the problem related with database availability is the process of maintaining multiple copies of data, and if any problem occurs with data, it is possible to access same data in other sites of a database;

- Cross-site database operation: There are many applications that use open multiple databases on different sites, and it is always possible to send replication updates and copies of the transfer process to a central location;

- Scaling: It is possible to distribute a real traffic of replicated copies, and reports made on a replica will not affect other copies;

- Upgrades: Users are allowed to carry out the upgrade process to copy the replica and then a main copy is to be switched. This is one of the classic styles to provide

optimal downtime, in addition to providing appropriate back-out whether there may any problem happen;

- Heterogeneous database integration: Data can be entered in one database type, and used in another database type, and the proper conversion of transformation should be provided;

- Data warehouse loading: Updates in data replication are performed in a real time. Data warehouse process is executed easily by transforming or copying updates to a central location;

- Geographic distribution: In order to avoid the process of site failure the users are allowed to put two or more clusters in different places located far from each other.

4.1.1. Normalization

Data replication is an effective way to eliminate the incontinency or ambiguity of data among users, known as normalization. In normalization process some tables are created, the relationships depending on some rules are established between these tables, and the most important property is that making any changes in one of these tables will not affect other table(s).

Normalization of a database system is an appropriate method to effectively store data, to eliminate redundancy, to minimize redundancy and functional dependency, and to update, insert, and delete anomalies.

Each of the rules mentioned above is called “normal form”. If the first one of these rules is applied, it is called a “first normal form”. If the first three of these rules are possible to be used, the database will be in “third normal form”.

There are three main normalization processes defined as following:

- First normal form: It should be made sure that there is a primary key in the table, and there are no repeating groups of data, and all the values of the attributes are atomic (not multiple);

- Second normal form: A database is in the second normal form, if it is already in the first normal form, and all the non-primary key attributes are entirely functionally depending on the primary key;

- Third normal form: A database is in the third normal form, if it is already in the second normal form so that all the non-primary key attributes are depending on a primary key, and also no transitive functional dependency exists.

4.2 Data Allocation

One of the important issues that the performance of DDBMS dependent on is the allocation of data between the multiple sites on the network, and it has problem with the allocating of data and this problem is related with some of concepts such as the elements to be allocated are not known, and using schedules to access these elements. These schedules contain sharing process between elements to have the result. So it is very important to choose the convenient technique for allocation in the

DDBMS, because it provides just the restricted reply to the modification in workload.

In distributed database system it is necessary to provide the database servers and then install the application strategy.

Using the optimum and line analysis to determine the dissemination of data in DDBMS is very important especially while using fixed patterns to access the database.

The aim of data allocation algorithm is to determine the work of fragments by establishing locations to minimize the total cost of data transfer in working of fragments in executing some queries with the consideration of the constraints in database. The largest number of fragmentation processing allocated at every site should be defined, and the query execution time must be reduced.

How does the system find an optimal distribution of fragments on the sites? The definition of the optimality can be given with two measures:

- Minimal cost. This measure consists of a cost of data storing, cost of query processing, cost of data update over all the sites and cost of communication. The optimal schema for the minimization of the cost is used;

- Performance. The strategy of allocation designed to lead the performance is required. There are two ways: one of them is to minimize the response time of the query and the other one is to maximize locating of the system at every site.

4.3 Data Fragmentation

In a design process of the distributed database management system the entire relation can be subdivided into some distributed relations called fragments. One of the main ideas of using fragmentation is that transactions can be executed concurrently, because it is possible to access different portions of relations at the same time.

A fragmentation is a feature of a database server which enables the controlling over data stored at any table. It makes the user defining a set of row groups or index keys for a table depending on some of scheme or algorithms, and it can be stored in any of these groups at individual database space. The user can use relational algebra operations or Structured Query Language (SQL) to build fragments and focus them at a database space. The processing of using scheme for row groups or index keys is called a distribution scheme, and the database spaces and distribution scheme to be placed together in a fragmentation makes the fragmentation strategy.

After the creation of the fragments, the server of the database starts to store the location for every table fragment and index with information which are related to the system, and it is called a system fragment, and this table is used to enter data about fragmented tables and indices. In some distribution schemes a database server contains most of information about the fragments and the data inside these

fragments, so the data server will be drawing the data request of user to the convenient fragment and it is not allowed to enter the unrelated fragments.

4.3.1 Purpose of using Fragmentation

There are many goals for using fragmentation, and these goals are shown below:

- Single user response time;

- Concurrency;

- Availability;

- Backup and restore characteristics;

- Loading of data.

Each goal represented above has the implication about the strategy of fragmentation. The first goal of fragments is a determination of the execution process of the fragmentation strategy. The activity of monitoring and adding some administration for the fragmentation requirements are important.

4.3.2 Responsibility of Fragmentation Process

There is a big responsibility for an administrator of a database server to create the scheme which includes the table of fragmentation. On the other hand, responsibility of administrator of the database server is an allocation of the fragments in the table over disk size. The execution of the fragmentation requires mutual efforts between administrator of database server and database administrator.

4.3.3 Storage Fragmentation terms

For the index fragment or fragments in the table, the following issues are useful to understand to develop a strategy of the fragmentation:

- Fragment key. The table or index fragmented contain a column or set of columns, depending on the level of fragment strategy, and a fragment key may be is a column or expression of unique or multi columns;

- Fragment list. There must be a list of fragmentations to be ordered. The position of every fragment in the list shows which fragments should be created sequentially. The value will be updated automatically by database server to be returned to the list of fragmentation;

- Fragment expression. The particular fragment will be defined by the expression. For instance, consider fragment key is Nama with the data type SMALLINT, and it can define the fragment by expression Nama<=7 OR Nama IN (8, 9, 23, 24) in the expression depending on the fragment strategy;

- NULL fragment. The value with NULL means that is either there is some range of fragment or the list of fragment is expressed was NULL, or because it uses data type constraint of NULL in the database as expression;

- REMAINDER fragment. It is saving any row of fragment which is the value of fragment key not similar with any expression of fragment. This fragment is always located in the end of the fragment list;

- Transition fragment. A transition value of the table is the limit of upper transition of the fragment, and it is possible to increase transition value of the table.

Through the data fragmentation, anyone can allow to fracture one object to many fragments or segments. The object can be in several types, such as table, database of user or system the database, and can be used in communication network over the computers to place each fragment at any site.

4.3.4 Strategies of Data Fragmentation

The data fragmentation has three types of strategies: a horizontal fragmentation, a vertical fragmentation, and a mixed fragmentation.

The horizontal fragmentation indicates that it can split the relation into subsets of fragments of the rows. This type of fragmentation has a property that all rows of the relation should satisfy a selection condition. The selection condition can be connected by the logical statements AND or OR. The union operation of a relational algebra is used to reconstruct the original table decomposed into horizontal fragments. In the table 1 the customer relation is represented, and the attributes of this relation are customer number (CUS_NO), customer name (CUS_NAME), customer city (CUS_CITY), customer state (CUS_STATE), customer limit

(CUS_LIMIT), customer balance (CUS_BAL), and customer rating (CUS_RATING).

Table 1. Customer relation

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RAT
1	Bell	Dallas	Texas	4700	3800	3
2	Bryson	Miami	Florida	7300	3600	1
3	Davis	Dallas	Texas	5400	4200	3
4	Smith	Miami	Florida	7500	6800	3
5	Niles	Miami	Florida	2700	860	1
6	Robinson	Detroit	Michigan	3800	580	2

Suppose we want some information about the customers of three states. In the table 2, a horizontal fragmentation can be determined by the distribution of data according to the states Texas, Florida, and Michigan. The queries can be written in a relational algebra as

$$\Pi_{CUS\_NO,\ CUS\_NAME,CUS\_CITY,CUS\_STATE,\ CUS\_LIMIT,\ CUS\_BAL,\ CUS\_RATING}(\sigma_{CUS\_STATE\ =\ Texas}$$

(Customer))

$\Pi_{CUS\_NO, CUS\_NAME, CUS\_CITY, CUS\_STATE, CUS\_LIMIT, CUS\_BAL, CUS\_RATING}(\sigma_{CUS\_STATE = Florida}$

(Customer))

$\Pi_{CUS\_NO, CUS\_NAME, CUS\_CITY, CUS\_STATE, CUS\_LIMIT, CUS\_BAL, CUS\_RATING}(\sigma_{CUS\_STATE = Michigan}$

(Customer))

SQL codes of the same queries are represented as

```
SELECT CUS_NO, CUS_NAME, CUS_CITY, CUS_STATE, CUS_LIMIT,
CUS_BAL, CUS_RATING
FROM Customer
WHERE CUS_STATE = 'Texas';
```

```
SELECT CUS_NO, CUS_NAME, CUS_CITY, CUS_STATE, CUS_LIMIT,
CUS_BAL, CUS_RATING
FROM Customer
WHERE CUS_STATE = 'Florida';
```

```
SELECT CUS_NO, CUS_NAME, CUS_CITY, CUS_STATE, CUS_LIMIT,
CUS_BAL, CUS_RATING
FROM Customer
WHERE CUS_STATE = 'Michigan';
```

Table 2. Horizontal fragmentation by using the attribute CUS_STATE

CUS_STATE Texas

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RAT
1	Bell	Dallas	Texas	4700	3800	3
3	Davis	Dallas	Texas	5400	4200	3

CUS_STATE Florida

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RAT
2	Bryson	Miami	Florida	7300	3600	1
4	Smith	Miami	Florida	7500	6800	3
5	Niles	Miami	Florida	2700	860	1

CUS_STATE Michigan

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RAT
6	Robinson	Detroit	Michigan	3800	580	2

From the table Customer it is to see that a horizontal fragmentation can be also determined by the distribution of data according to the customer rating represented in the table 3. The queries can be written in a relational algebra as

$\Pi_{CUS\_NO, CUS\_NAME, CUS\_CITY, CUS\_STATE, CUS\_LIMIT, CUS\_BAL, CUS\_RATING}(\sigma_{CUS\_RATING = 1}$

(Customer))

$\Pi_{CUS\_NO, CUS\_NAME, CUS\_CITY, CUS\_STATE, CUS\_LIMIT, CUS\_BAL, CUS\_RATING}(\sigma_{CUS\_RATING = 2}$

(Customer))

$\Pi_{CUS\_NO, CUS\_NAME, CUS\_CITY, CUS\_STATE, CUS\_LIMIT, CUS\_BAL, CUS\_RATING}(\sigma_{CUS\_RATING = 3}$

(Customer))

SQL codes of the same queries are represented as

```
SELECT CUS_NO, CUS_NAME, CUS_CITY, CUS_STATE, CUS_LIMIT,
CUS_BAL, CUS_RATING
FROM Customer
WHERE CUS_RATING = 1;
```

```
SELECT CUS_NO, CUS_NAME, CUS_CITY, CUS_STATE, CUS_LIMIT,
CUS_BAL, CUS_RATING
FROM Customer
WHERE CUS_RATING = 2;
```

```
SELECT CUS_NO, CUS_NAME, CUS_CITY, CUS_STATE, CUS_LIMIT,
CUS_BAL, CUS_RATING
FROM Customer
```

WHERE CUS_RATING = 3;

Table 3. Horizontal fragmentation by using the attribute CUS_RATING
CUS_RATING 1

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RAT
2	Bryson	Miami	Florida	7300	3600	1
5	Niles	Miami	Florida	2700	860	1

CUS_RATING 2

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RAT
6	Robinson	Detroit	Michigan	3800	580	2

CUS_RATING 3

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RAT
1	Bell	Dallas	Texas	4700	3800	3
3	Davis	Dallas	Texas	5400	4200	3
4	Smith	Miami	Florida	7500	6800	3

Let’s consider the derived horizontal fragmentation of the Customer relation. We will use the CUS_LIMIT attribute to make derived horizontal fragmentation.

In the first part of the table data about customers with limitations less than or equal to 5000 are stored, and the second part contains data about customers with limitations more than 5000 (Table 4). The queries in relational algebra can be written as

$$\Pi_{CUS\_NO,\ CUS\_NAME,CUS\_CITY,CUS\_STATE,\ CUS\_LIMIT,\ CUS\_BAL,\ CUS\_RATING}(\sigma_{CUS\_LIMIT<=5000}$$

(Customer))

$$\Pi_{CUS\_NO,\ CUS\_NAME,CUS\_CITY,CUS\_STATE,\ CUS\_LIMIT,\ CUS\_BAL,\ CUS\_RATING}(\sigma_{CUS\_LIMIT>5000}$$

(Customer))

The SQL codes of above queries are

```
SELECT CUS_NO, CUS_NAME, CUS_CITY, CUS_STATE, CUS_LIMIT,
CUS_BAL, CUS_RATING
FROM CUSTOMER
WHERE CUS_LIMIT < = 5000;
```

```
SELECT CUS_NO, CUS_NAME, CUS_CITY, CUS_STATE, CUS_LIMIT,
CUS_BAL, CUS_RATING
FROM CUSTOMER
WHERE CUS_LIMIT > 5000;
```

Table 4. Derived horizontal fragmentation by using the attribute CUS_LIMIT
CUS_LIMIT 1

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RAT
1	Bell	Dallas	Texas	4700	3800	3
5	Niles	Miami	Florida	2700	860	1
6	Robinson	Detroit	Michigan	3800	580	2

CUS_LIMIT 2

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RAT
2	Bryson	Miami	Florida	7300	3600	1
3	Davis	Dallas	Texas	5400	4200	3
4	Smith	Miami	Florida	7500	6800	3

The vertical fragmentation indicates that it can divide the relation into subsets of attributes (columns), and every node contains a subset of fragment that has exactly one uniquely column, except the key of column that will be general to every fragment. The join operation of a relational algebra is used to reconstruct the original table decomposed into vertical fragments. Compare to the horizontal fragmentation, the vertical fragmentation is more complicated because of having more alternatives.

The Customer relation represented in table 1 can be described in the form of vertical fragmentation which is given in the table 5, and the queries in relational algebra are as

$\prod_{CUS\_NO,\ CUS\_NAME,CUS\_CITY,CUS\_STATE}$ (Customer)

$\prod_{CUS\_NO,\ CUS\_LIMIT,\ CUS\_BAL,\ CUS\_RATING}$ (Customer)

The queries in SQL form are represented as

```
SELECT CUS_NO, CUS_NAME, CUS_CITY, CUS_STATE
FROM CUSTOMER;
```

```
SELECT CUS_NO, CUS_LIMIT, CUS_BAL, CUS_RATING
FROM CUSTOMER;
```

Mixed fragmentation (or alternatively called hybrid fragmentation) indicates the involvement of composition of horizontal and vertical partitioning; a table or structure may be divided into some subsets of horizontal rows. Two possible mixed fragmentations are either a horizontal fragment subsequently represented by vertical fragment, or vertical fragment subsequently represented by horizontal fragment. Both selection and projection operations are used in mixed fragmentation.

Table 5. Vertical fragmentation of the Customer relation

V1

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE
1	Bell	Dallas	Texas
2	Bryson	Miami	Florida
3	Davis	Dallas	Texas
4	Smith	Miami	Florida
5	Niles	Miami	Florida
6	Robinson	Detroit	Michigan

V2

CUS_NO	CUS_LIMIT	CUS_BAL	CUS_RAT
1	4700	3800	3
2	7300	3600	1
3	5400	4200	3
4	7500	6800	3
5	2700	860	1
6	3800	580	2

In the table 6 the first mixed fragmentation of the Customer relation is represented. The horizontal fragmentation in which the relation divided into some horizontal

fragments according to the attribute CUS_STATE is followed by the vertical fragmentation.

Table 6. First mixed fragmentation of the Customer relation

H1

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RAT
1	Bell	Dallas	Texas	4700	3800	3
3	Davis	Dallas	Texas	5400	4200	3

V1_1

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE
1	Bell	Dallas	Texas
3	Davis	Dallas	Texas

V1_2

CUS_NO	CUS_LIMIT	CUS_BAL	CUS_RAT
1	4700	3800	3
3	5400	4200	3

H2

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RAT
2	Bryson	Miami	Florida	7300	3600	1
4	Smith	Miami	Florida	7500	6800	3
5	Niles	Miami	Florida	2700	860	1

V2_1

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE
2	Bryson	Miami	Florida
4	Smith	Miami	Florida
5	Niles	Miami	Florida

V2_2

CUS_NO	CUS_LIMIT	CUS_BAL	CUS_RAT
2	7300	3600	1
4	7500	6800	3
5	2700	860	1

H3

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RAT
6	Robinson	Detroit	Michigan	3800	580	2

V3_1

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE
6	Robinson	Detroit	Michigan

V3_2

CUS_NO	CUS_LIMIT	CUS_BAL	CUS_RAT
6	3800	580	2

In the second mixed fragmentation of the Customer relation we begin with the consideration of the vertical fragmentation, and then the horizontal fragmentation is applied. The first vertical fragmentation is described in the form of horizontal fragmentation according to the CUS_STATE attribute, and the second vertical fragmentation is described in the form of horizontal fragmentation according to the CUS_RATING attribute.

Table 7. Second mixed fragmentation of the Customer relation

V1

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE
1	Bell	Dallas	Texas
2	Bryson	Miami	Florida
3	Davis	Dallas	Texas
4	Smith	Miami	Florida
5	Niles	Miami	Florida
6	Robinson	Detroit	Michigan

H1_1

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE
1	Bell	Dallas	Texas
3	Davis	Dallas	Texas

H1_2

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE
2	Bryson	Miami	Florida
4	Smith	Miami	Florida
5	Niles	Miami	Florida

H1_3

CUS_NO	CUS_NAME	CUS_CITY	CUS_STATE
6	Robinson	Detroit	Michigan

V2

CUS_NO	CUS_LIMIT	CUS_BAL	CUS_RAT
1	4700	3800	3
2	7300	3600	1
3	5400	4200	3
4	7500	6800	3
5	2700	860	1
6	3800	580	2

H2_1

CUS_NO	CUS_LIMIT	CUS_BAL	CUS_RAT
2	7300	3600	1
5	2700	860	1

H2_2

CUS_NO	CUS_LIMIT	CUS_BAL	CUS_RAT
6	3800	580	2

H2_3

CUS_NO	CUS_LIMIT	CUS_BAL	CUS_RAT
1	4700	3800	3
3	5400	4200	3
4	7500	6800	3

Chapter 5

CONCLUSION

Advances in database management technology have led to the development of the distributed database management systems enabling the process of sharing the information stored in many computers by using one computer, and making the information transparent to all the users that is important to improve the performance of the database system. The design technologies and implementation process of the distributed database management systems have been growing so rapidly that it is expected the distributed database management systems are very promising in terms of efficiency of the application instead of centralized database systems in the future.

The analysis of a distributed database management system for data retrieval is a main goal of this thesis. The characteristics of the distributed database management systems are given. Homogeneous and heterogeneous classifications of the distributed database management systems and their differences are discussed. The data replication, data allocation, and data fragmentation in distributed database environment are performed. The operations of relational algebra and SQL statements are used for the fragmentation process.

REFERENCES

[1] N.Geetha. (2004). Distributed Database Management Systems for Information Management and Access. *2nd International CALIBER-2004, New Delhi, 11-13 February,* pp. 464-469.

[2] Heinz Stockinger. (2001). Distributed Database Management Systems and the Data Grid. *Proceedings of the Eighteenth IEEE Symposium on Mass Storage Systems and Technologies, April 17-21,* pp. 1-12.

[3] Gomer Thomas, Glenn R. Thompson, Chin-Wan Chung, Edward Barkmeyer, Fred Carter, Marjorie Templeton, Stephen Fox, Berl Hartman. (1990). Heterogeneous distributed database systems for production use. *ACM Computing Surveys (CSUR) - Special issue on heterogeneous databases. Volume 22, Issue 3*, pp. 237-266.

[4] Peter M.G.Apers. (1988). Data allocation in distributed database systems. *ACM Transactions on Database Systems (TODS), Volume 13, Issue 3,* pp. 263-304.

[5] Anand K. Tripathi, Monika Tripathi. (2012). A Framework of Distributed Database Management Systems in the Modern Enterprise and the Uncertainties removal. *International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 4*, pp. 60-66.

[6] Djam Xaveria Youh. (2010). Design and Implementation of a Client Server Distributed Database for Student Results Processing. *The Pacific Journal of Science and Technology, Volume 11, Number 2,* pp. 288-295.

[7] Fan Yuanyuan, Mi Xifeng. (2010). Distributed Database System Query Optimization Algorithm Research. *IEEE International Conference on Computer Science and Information Technology (ICCSIT), Volume 8,* pp. 657-660.

[8] Stefano Ceri, Barbara Pernici, Gio Wiederhold. (1987). Distributed Database Design Methodologies. *Proceedings of the IEEE, 1987, Volume 75, Issue 5,* pp. 533-546.

[9] Peter Lyngbaek, Dennis McLeod. (1983). An Approach to Object Sharing in Distributed Database Systems. *Proceedings of the 9th International Conference on Very Large Data Bases, VLDB'83,* pp. 364-375.

[10] Bhavani Thuraisingham and Harvey H. Rubinovitz. (1992). Multilevel Security Issues in Distributed Database Management Systems-III. *Computers & Security, 11,* pp. 661-674.

[11] Bhavani Thuraisingham. (1995). Security constraint processing in a multilevel secure distributed database management system. *IEEE Transactions on Knowledge and Data Engineering, Vol.7, No.2,* pp. 274-293.

[12] Thuraisingham, B. (1990). Secure query processing in distributed database management systems-design and performance studies. *Proceedings of the Sixth Annual Computer Security Applications Conference, December 3-7,* pp. 88-102.

[13] Kjetil Nørvag, Olav Sandsta, and Kjell Bratbergsengen. (1997). Concurrency Control in Distributed Object-Oriented Database Systems. *Proceedings of the First East-European Symposium on Advances in Databases and Information Systems (ADBIS'97), Volume 1,* pp. 1-14.

[14] Dana L. Small, Deborah Goldsmith, Bhavani M. Thuraisingham. (1994). Consistent Data Access in a Distributed Database Management System for Command and Control Applications. *Proceedings of the Conference on High Performance Computing'94,* pp. 342-347.

[15] Henry M. Walker. (1982). Administering a distributed Data Base Management System. *ACM SIGMOD Record, Volume 12, Issue 3,* pp. 86-99.

[16] Kenneth R. Abbott, Dennis R. McCarthy. (1988). Administration and Autonomy in a Replication-Transparent Distributed DBMS. *Proceedings of the 14th International Conference on Very Large Data Bases, VLDB'88,* pp. 195-205.

[17] Svetlana Vasileva, Petar Milev, Borislav Stoyanov. (2007). Some Models of a Distributed Database Management System with Data Replication. *Proceedings of the*

2007 International Conference on Computer Systems and Technologies CompSysTech 2007, pp. II.12-1 - II.12-6.

[18] S. Ceri and G. Pelagatti. (1983). Correctness of Query Execution Strategies in Distributed Databases. *ACM Transactions on Database Systems, Vol. 8, No. 4, December 1983,* pp. 577 - 607.

[19] Voss, K. (1980). Using Predicate/Transition-Nets to Model and Analyze Distributed Database Systems. *IEEE Transactions on Software Engineering, Vol.6, No.6,* pp. 539-544.

[20] Khake Asha, Gojamgunde Ashwini, Shastri Ashlesha and Biradar Usha. (2011). Transaction Management in Distributed Database. *BIOINFO Transactions on Database Systems, Volume 1, Issue 1,* pp. 1-6.

[21] Omar Baakeel and Abdulaziz Alrashidi. (2012). Distributed Transaction Model for a Multi-database Management System. *International Journal of Scientific & Engineering Research, Volume 3, Issue 3,* pp. 1-4.

[22] Ranjana Kumari, Arun Kumar Singh. (2012). Partial Security Violation Model in Non-Blocking Distributed Databases. *International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 12,* pp. 283-289.

[23] Sunil Kumar, J.Seetha, S.R.Vinotha. (2012). Security Implications of Distributed Database Management System Models. *International Journal of Soft Computing and Software Engineering (JSCSE), Vol. 2,No.11,* pp. 20-28.

[24] Michael Stonebraker, John Woodfill, Jeff Ranstrom, Joseph Kalash, Kenneth Arnold, Erika Andersen. (1983). Performance analysis of distributed data base systems. *In Proceedings of the Third Symposium on Reliability in Distributed Software and Database Systems*, pp. 135-138.

[25] B.M. Monjurul Alom, Frans Henskens and Michael Hannaford. (2009). Query Processing and Optimization in Distributed Database Systems. *IJCSNS International Journal of Computer Science and Network Security, Vol.9, No.9,* pp. 143-152.

[26] Dejan Chandra Gope (2012). Dynamic Data Allocation Methods in Distributed Database System. *American Academic & Scholarly Research Journal AASRJ, Volume 4, Number 6,* pp. 1-8.

[27] M. Tamer Özsu, Patrick Valduriez. (2011). Principles of Distributed Database Systems. *Springer, Third Edition.*

[28] http://ecomputernotes.com/database-system/adv-database/distributed-database

[29] http://en.wikipedia.org/wiki/Distributed_database.

[30] http://searchsqlserver.techtarget.com/definition/database-replication.

[31]https://s3.amazonaws.com/releases.continuent.com/doc/tungsten-1.3.2/html/Tungsten-Replicator-Guide/content/ch01s01.html.