Aspects of the design of distributed databases

Burlacu Irina-Andreea, Titu Maiorescu University, Romania

Abstract

Distributed data - data, processed by a system, can be distributed among several computers, but it is accessible from any of them.

A distributed database design problem is presented that involves the development of a global model, a fragmentation, and a data allocation. The student is given a conceptual entity-relationship model for the database and a description of the transactions and a generic network environment. A stepwise solution approach to this problem is shown, based on mean value assumptions about workload and service. A management system of a distributed database (SGBDD) is a software system that enables management and distributing BDD transparent to the user. A SGBDD consists of a single database which is decomposed into fragments, poassibly some fragments are multiplied, and each fragment or copy kept on one or more sites under the control of a local DBMS. Each site is capable of processing user queries in the local system, independently of the rest of the network, or is able to participate in the processing of data in other sites in the network. To say that a DBMS is distributed CLE should be less global demand.

Keywords: databases, DBMS, SGBDD, distributed databases, design

Introduction

Distributed database systems (SBDD) are two approaches to meeting the data processing which may seem diametrically opposed: technology systems and the database of computer networks. Database systems have evolved from data processing in which each application to define and maintain their own data, to one in which the data are defined and managed centrally. This new orientation leads to independent data, such applications become immune to changes in physical or logical data organization and vice versa. A major motivation in using database systems is the integration of data and provide a centralized and controlled access to data. On the other hand, the technology of computer networks promotes a thing that is against all efforts of centralization. It can be so difficult to understand how these two contrasting approaches can be summarized in a technology that is stronger and more promising than both. The key to understanding is the realization that the most important objective of database technology is not centralized, but integration. It is important to note that none of these terms do not implicate the other. Perhaps no integration without centralization, it is the very purpose of distributed databases.

Distributed Data Processing

Distributed Processing is one of the most abused terms in computer science in recent years. It was used to refer to various systems such as multiprocessor systems, distributed data processing and computer networks. Distributed processing is a concept that is difficult to give a rigorous definition, so we give a definition in terms of distributed database systems. A distributed computing system consists a number of autonomous processing elements (not necessarily homogeneous) that are interconnected by a network of computers and cooperate in performing tasks (task) them. By "processing element" means a computer that can run their own programs.

In this context, the question: What is distributed? Processing logic is something to be distributed. The definition of a computer system given above implies that the logic processing or processing elements are distributed. Another may be the distribution functions. The various functions of a computer system can be performed by different parts of hardware or software. Another may be that according to data distribution. The data used by a number of applications can be distributed to multiple processing nodes. Finally, and control can be distributed. Control execution of different tasks can be distributed instead of being executed by a single computer. From the point of view of distributed databases, these methods of distribution are necessary and important.

Another question that can be asked is: Why distribute? Classical answers to this question said that better reflects the distributed processing of large enterprise organizational structure today, and that such a system is safer and better. From a global perspective, however, can be said that the main reason for distributed processing is large and complicated problems we face today, using variations of the well known rules divide et impera. If the required software distributed processing can be created, then it is possible to solve these complicated problems simply by sharing their smaller parts that can be solved by different software groups, located on different computers, producing a system that runs on more many processing elements, but can effectively perform a common task.

What is a distributed database system?

We define a distributed database as a collection of logically interrelated databases distributed over a computer network. A distributed database management is defined as a software system that enables management of distributed databases and makes the distribution transparent to users. There are two important terms in this definition: logically related in a distributed computer network.

Distributed database is a virtual character. Its components are stored on separate databases, located on separate nodes of a network. Each node is a database system, with its own database, its users and the local DBMS. Distributed system can be viewed as a partnership between the local DBMS and a new component in each node. It provides the functions necessary for pairing.

A distributed database system (SBDD) is not only "a collection of files" that can be stored individually on each node of a network of computers.

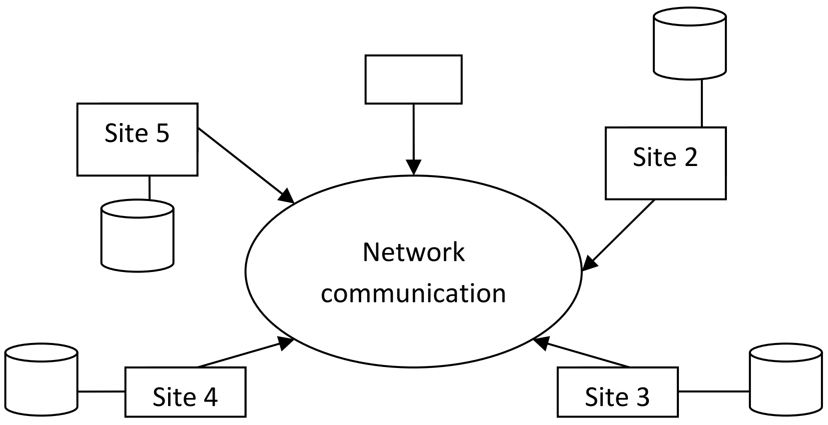

Figure 1.1 Environment SBDD

Advantages and disadvantages of distributed database

Advantages:

Local autonomy. Users of the database is on a certain station in the network have local control of data, due to the decentralized organization. Local data is kept locally, where they belong logically in most cases is in executing their processing node are stored.

Performance Improvement. The parallel execution of multiple tasks in different stations, it can reduce data access conflicts and can improve both speed of execution of operations on the database and speed access to stored information.

Improving safety and availability. If the data is replicated on multiple nodes, the unavailability of one of them or a network problem is not that the data can not be accessed.

Economic. If you are geographically dispersed databases and applications that are running in place and data, improves communication costs.

Expandability. In a distributed environment is much easier to increase the database size.

Partajabilitate. Organizations that have geographically distributed operations and data stores normally in the same manne.

Disavantages:

Complexity. Problems in the SBDD are more complex than centralized ones, including problems they shared, and the still unresolved in the centralized environment. The problem is the complexity of the programmer and not the user.

Cost: Distributed systems require additional hardware and software, which increases costs.

Distribution control. This point, which is also an advantage, cause problems of timing and coordination.

Security. It is well known difficulty in maintaining adequate control of network security, so in distributed databases.

Adoption of technology difficult. Many companies have invested heavily in their database systems that are not distributed. Currently there are no tools or technologies that help users to convert the centralized databases distributed databases.

Distributed database design

Designing a distributed computing system involves taking decisions on the placement of data and programs in a computer network nodes, and network design itself. In the case of distributed databases, assuming that the network has been designed already and there is a copy of the DBMS software on each node in the network where data are stored, it remains to focus our attention on the distribution of data.

Alternativ design strategia

There are two major strategies for the design of distributed databases: top down design and bottom-up design.

As the name indicates, these strategies are very different approaches to the design process. But most applications are not so simple that it fits completely in one of these strategies, so it is important to know that these strategies should be used together as a complement to each other.

Top down design

The work begins with the analysis of requirements that define the system environment. The document is the entrance requirements for two parallel activities: conceptual design and design views. Visual design activity defines interfaces for end users. Conceptual design is the process by which the system is examined to determine its component types of entities and relationships between them. Conceptual design can be interpreted as the integration of user views. This is very important because the conceptual model must not only support existing application , but also future onest Global Conceptual Schema and information about access patterns collected as a result of design views are distributed design step inputs. The objective is now to design conceptual schemes through local distribution entities across nodes of the distributed system. In a relational model, entities corresponding relationships. Rather than distributing their relations division subrelatii is used, called fragments, and they are distributed. So distributed design activity consists of two steps: fragmentation and allocation.
The last step in the design process is the physical design, which makes connections between conceptual schemes and local physical storage devices on the nodes corresponding data. Entries in this process are the local conceptual schemes and patterns of access to information.

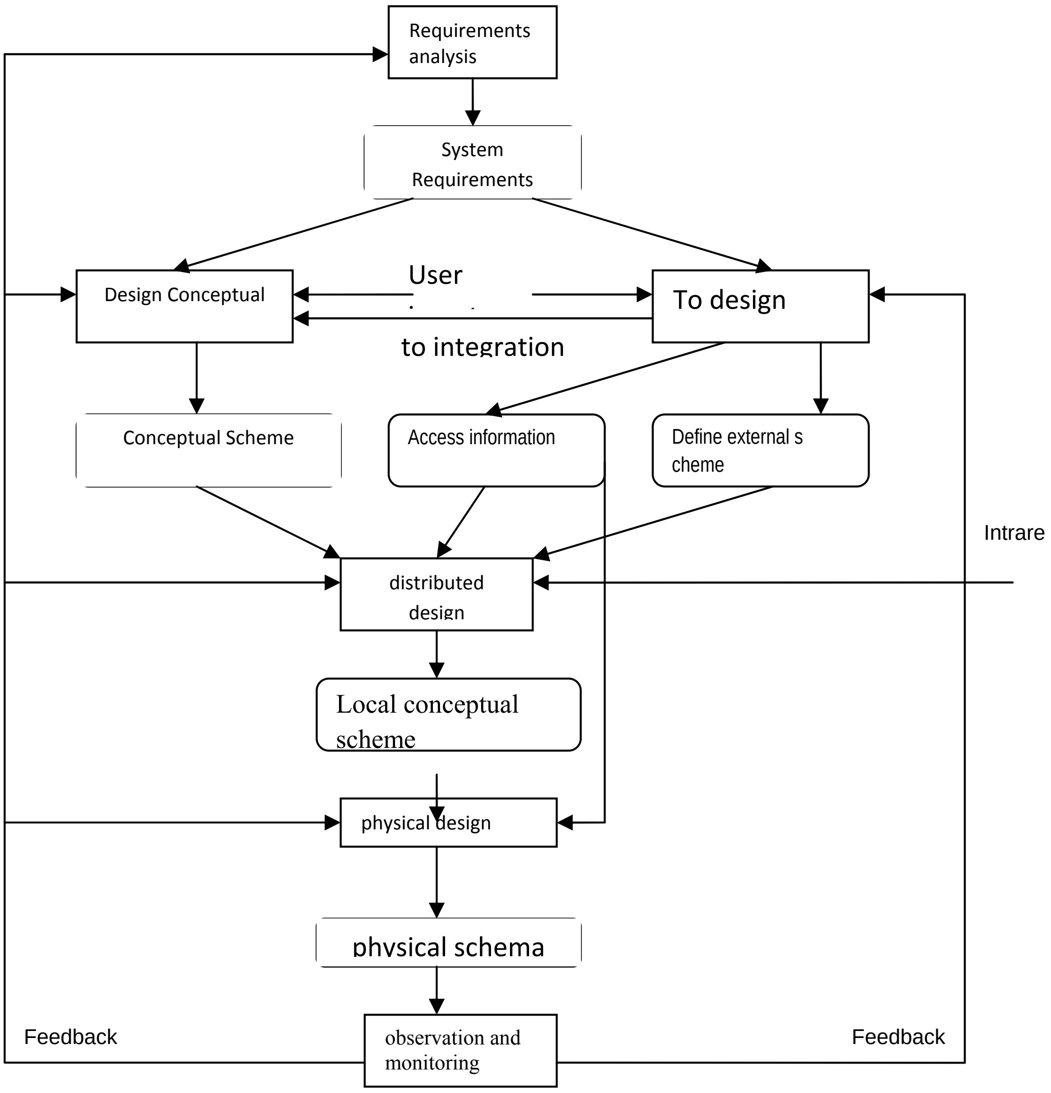

Figure 1.2 Top-Down design process

Bottom-Up Design

Top-down approach is suitable when we are designing a BDD starting from scratch. But it often happens that some databases already exist, and design activities must realize and integration. Bottom-up approach is suitable for such environments.

The starting point in designing bottom-up is local conceptual schema. The process consists in the integration of local schemes in the global conceptual schema.

Transparency SGBDD

SGBDD Transparency refers to the separation of high-level semantics of an implementation system at a low level. In other words, hide the implementation details transparent to users. Advantages of DBMS transparent sites are portability, reconfigurabilitatea, providing a development environment for complex applications.

Data Independence

Independence of data is a fundamental form of SGBDD transparency. It refers to the immunity of applications to change user in defining and organizing data, and vice versa.

You can highlight the two types of independent data:
Logical data independence refers to the immunity of user applications from changes in the structure of the database logic. If an application works with a subset of attributes of a relationship, it will not hurt to add a new attribute to the same relationship.
Physical data independence refers to hiding details of storage structure for user applications. The application does not need to be modified whenever there are changes in the organization of data.

Network Transparency

The media management of distributed databases is an asset that must be managed: the network. Preferably, a user will be protected by the network's operational details. If possible, the existing network should be hidden. Then there will be no difference (in terms of the user) between applications using centralized databases and those that use distributed databases. This type of transparency is called network transparency.

Transparency in replication

It is necessary that the data is distributed by a replicative manner between machines on a network. So the same data will be found on many cars. This improves system performance since different and competing demands of users can be more easily satisfied. For example, data are often accessed by a user can store his car, and that of other users with similar access requirements. If a network node is unavailable, data can be accessed from other locations. What data should be replicated in many children depends largely on the applications that access them. Data replication but causes some problems updating.
The fact that the user does not know how many copies were made and does not know anything about their existence, we can call the transparent replication (replication Transparency).

Transparency to fragmentation

It is intended that each database object to be divided into small fragments and each fragment to be treated as a separate object database. This is caused by reasons of performance, availability and reliability. Also, fragmentation can reduce the negative effects of replication.

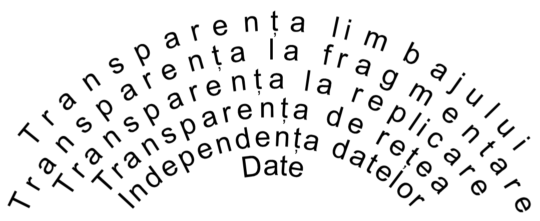

Figure 1.3: Levels of transparency

Conclusion

Internal management of distributed databases is demanding and generally difficult, because we have ensured that:

- Distribution is transparent (invisible and unobtrusive) - users must be able to interact with the system as if they were a non-distributed (monolithic)
- Transactions must also have a transparent structure (invisible and unobtrusive).

Course each transaction must maintain database integrity, despite the multiplicity of partitions. For this they are usually divided subtranzacţii, each of them working with only one partition.

References:

1. Elmasri and Navathe, *Fundamentals of database systems* (3rd edition), Addison-Wesley Longman, ISBN 0-201-54263-3
2. M. T. Ozsu and P. Valduriez, *Principles of Distributed Databases* (2nd edition), Prentice-Hall, ISBN 0-13-659707-6
3. O'Brien, J. & Marakas, G.M.(2008) Management Information Systems (pp. 185-189). New York, NY: McGraw-Hill Irwin
4. O'Brien, J. & Marakas, G.M.(2008) Management Information Systems (pp. 185-189). New York, NY: McGraw-Hill Irwin.