International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-7 Issue-4, April 2018

Various Types of Transparencies in Distributed Homogeneous and Heterogeneous Database Systems

Fatima Jahan Sarmin, Md. Mamunur Rashid

Abstract*: The architecture of database system becomes more complex day by day as different sites or organizations use different hardware, software, operating system or database. Not only is it about different systems, but also about different locations as same companies establish data centers at various part of the world. Distributed database system can connect the world in a system where a user thinks that it is a single system, but in architecture it is not. This heterogeneity leads to the thinking about the transparency in database management system. As whatever the system is, user need not to know about the internal architecture; they need only the accessibility of their information. And this is what transparency actually means presenting the whole system as a single system to the user and hiding the internal architecture. This paper briefly discusses the various types of transparency that need to be present in different types of distributed database systems.

***Index Terms*: Database, Distributed, Heterogeneous, Transparency.**

I. INTRODUCTION

To go through distributed database system, it is important to understand what a database do and why the distributed database systems appear. A Database is systematically organized or structured repository of indexed information that allows easy retrieval, updating, analysis, and output of data. Each database may involve different database management systems and different architectures that distribute the execution of transactions [1]. A Database is controlled by Database Management System (DBMS) to maintain and use large amount of data through an application software that interacts with the user, other applications, and the database itself to capture and analyze the data. DBMSs can be of different types and Distributed Database Management System (DDBMS) is one of them. DDBMS is the union of DBMS and computer network technologies. A distributed database system consists of loosely coupled sites that share no physical components. A distributed database may be stored in multiple computers located in the same or dispersed physical location over a network, but which appears to the users to be one single and whole entity. This characteristic of appearing like a single system to the user is provided by transparency.

II. DISTRIBUTED DATABASE SYSTEM (DDBS)

A. Definition

Distributed database system (DDBS) technology is the union of apparently two diametrically opposed approaches to data processing: database system and computer network technologies [2]. A distributed database is a single logical database that is spread physically across computers in multiple locations that are connected by a data communications network. It may reside on network servers on the Internet, on corporate intranets or extranets, or on other company networks. In a distributed database, storage devices are not all attached to a common processing unit such as the CPU. This system consists of loosely-coupled sites that share no physical components. Data, Process, and Interface components of an information system are distributed to multiple locations over a computer network. Distributed DBMS can also be integrated as a multiple-site processing, single-site data (MPSD), to allow more than one computer to access a single database [3]. The DDBMS offer more reliability by decreasing the risk of a single site failure. If one computer in the network fails, the workload is distributed to the rest of the computers. Furthermore, a DDBMS allows replication of data among multiple sites; data from the failed site may still be available at other sites.

Because they store data across multiple computers, distributed databases can improve performance at end-user worksites by allowing transactions to be processed on many machines, instead of being limited to one [4].Two processes ensure that the distributed databases remain up-to-date and current: replication and duplication. [3]

- ✓ Replication involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. The replication process can be complex, resource-hungry and time-consuming depending on the size and number of the distributed databases.
- ✓ Duplication, on the other hand, has less complexity. It basically identifies one database as a master and then duplicates that database. The duplication process is normally done at a set time after hours. In the duplication process, users may change only the master database. This ensures that local data will not be overwritten.

Manuscript published on 30 April 2018.
* Correspondence Author (s)
Fatima Jahan Sarmin, Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology, Rajshahi, Bangladesh, E-mail: sarminj18@gamil.com
Dr. Md. Mamunur Rashid, Associate Professor, Department of Computer Science and Engineering, Bangladesh Army University of Science and Technology, Bangladesh.

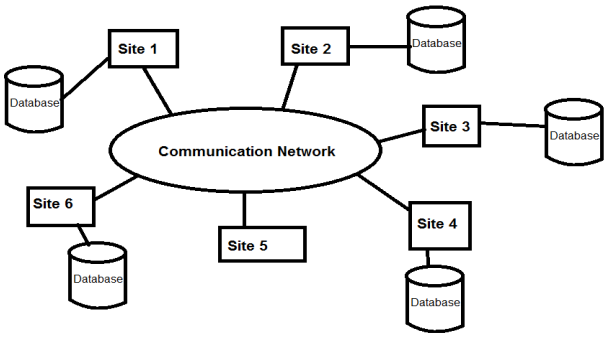

Fig 1: A Distributed Database Environment [6]

Both replication and duplication can keep the data current in all distributive locations [4].

B. Types of Distributed Database Systems (DDBS)

Homogeneous: The same DBMS is used at each node. It is much easier to design and manage. It provides incremental growth and allows increased performance.

Heterogeneous: The different DBMSs (e g., Oracle, DB2) are used at different nodes. Here, different sites implements their own databases first and integration is considered later.

C. Failures in DDBS

Several types of failures may occur in a distributed database systems [3]:

Transaction Failures: When a transaction fails, it aborts. Thereby, the database must be restored to the state it was in before the transaction started. Transactions may fail for several reasons. Some failures may be due to deadlock situations or concurrency control algorithms.

Site Failures: Site failures are usually due to software or hardware failures. These failures result in the loss of the main memory contents. In distributed database, site failures are of two types: Total Failure where all the sites of a distributed system fail, and Partial Failure where only some of the sites of a distributed system fail.

Media Failures: Such failures refer to the failure of secondary storage devices. The failure itself may be due to head crashes, or controller failure. In these cases, the media failures result in the inaccessibility of part or the entire database stored on such secondary storage.

Communication Failures: Communication failures, as the name implies, are failures in the communication system between two or more sites. This will lead to network partitioning where each site, or several sites grouped together, operates independently. As such, messages from one site won't reach the other sites and will therefore be lost.

The reliability protocols then utilize a timeout mechanism in order to detect undelivered messages. A message is undelivered if the sender doesn't receive an acknowledgment. The failure of a communication network to deliver messages is known as performance failure.

III. TRANSPARENCY

Though DDBS is distributed but it should appear as a single system to the user [7]. If a DDBMS exhibits distribution transparency then user does not need to know about the following: [4]

a) Presence of replication of data i.e., no visible explicit activity to keep consistency among all data copies

b) Location of data or data fragment

c) Partitioning technique used to distribute data among multiple distributed databases.

A. Types of Transparency

One of the major objectives of Distributed database system is providing the appearance of centralized system to end user. The eight transparencies are believed to incorporate the desired functions of a distributed database system [9]. Such an image is accomplished by using the following transparencies:

- Access Transparency
- Location Transparency
- Concurrency Transparency
- Replication Transparency
- Failure Transparency
- Migration Transparency
- Performance Transparency
- Scaling Transparency

Access Transparency: There should be no apparent difference between local and remote access methods. In other words, explicit communication may be hidden. For instance, from a user's point of view, access to a remote service such as a printer should be identical with access to a local printer. From a programmer's point of view, the access method to a remote object may be identical to access a local object of the same class. This transparency has two parts:

a. Keeping a syntactical or mechanical consistency between distributed and Non-distributed access,

b. Keeping the same semantics. Because the semantics of remote access are more complex, particularly failure modes, this means the local access should be a subset.

Remote access will not always look like local access in that certain facilities may not be reasonable to support (for example, global exhaustive searching of a distributed system for a single object may be unreasonable in terms of network traffic).

Location Transparency: Clients should see a uniform file name space. Files or groups of files may be relocated without changing their pathnames. A location transparent name contains no information about the named object's physical location. This property is important to support the movement of the resources and the availability of services. The location and access transparencies together are sometimes referred as Network transparency. The examples are File system in NFS and the pages of the web.

Concurrency Transparency: Users and Applications should be able to access shared data or objects without interference between each other. This requires very complex mechanisms in a distributed system, since there exists true concurrency rather than the simulated concurrency of a central system. The shared objects are accessed simultaneously. The concurrency control and its implementation is a hard task. The examples are NFS, Automatic Teller machine (ATM) network.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

Replication Transparency: This kind of transparency should be mainly incorporated for the distributed file systems, which replicate the data at two or more sites for more reliability. The client generally should not be aware that a replicated copy of the data exists. The clients should also expect operations to return only one set of values. The examples are Distributed DBMS and Mirroring of Web pages.

Failure Transparency: Enables the concealment of faults, allowing user and application programs to complete their tasks despite the failure of hardware or software components. Fault tolerance is provided by the mechanisms that relate to access transparency. The distributed systems are more prone to failures as any of the components may fail which may lead to degraded service or the total absence of that service. As the intricacies are hidden the distinction between a failed and a slow running process is difficult.

Migration Transparency: This transparency allows the user to be unaware of the movement of information or processes within a system without affecting the operations of the users and the applications that are running. This mechanism allows for the load balancing of any particular client, which might be overloaded. The systems that implement this transparency are NFS and Web pages.

Performance Transparency: Allows the system to be reconfigured to improve the performance as the load varies.

Scaling Transparency: A system should be able to grow without affecting application algorithms. Graceful growth and evolution is an important requirement for most enterprises. A system should also be capable of scaling down to small environments where required, and be space and/or time efficient as required. The best-distributed system example implementing this transparency is the World Wide Web.

Some other important transparencies are:

- ✓ **Execution Transparency:** Migration of processes will not be visible to the user.
- ✓ **Configuration Transparency:** The configuration does not affect the programming or use of the system.
- ✓ **Network Transparency**: The programmer/user does not have to know about the network. The system looks like a stand-alone computer.
- ✓ **Name Transparency:** Names will not change when the system is reconfigured.

B. Levels of Transparency

The three levels of transparency to hide certain complexities from the user, effectively managing the database as if it were centralized [3].

- ✓ **Fragmentation transparency:** The highest level of transparency divides the original database into fragments and disperses them throughout the DDBMS. Therefore, the user does not need to specify fragment names or locations to gain access.
- ✓ **Location transparency:** Location transparency ensures that the user can query on any table(s) or fragment(s) of a table as if they were stored locally in the user's site. The fact that the table or its fragments are stored at remote site in the distributed database system, should be completely oblivious to the end user. The address of the remote site(s) and the access mechanisms are completely hidden. In order to incorporate location transparency, DDBMS should have access to updated and accurate data dictionary and DDBMS directory which contains the details of locations of data.
- ✓ **Local mapping transparency:** The lowest level of transparency requires the user to know the name and location of fragments.

There is no reference in the following table to a situation in which the fragment name is "No" and the location name is "Yes." The reason for not including that scenario is simple: it cannot be possible to have a location name that fails to reference an existing fragment. [8]

IV. TRANSPARENCY IN HOMOGENOUS DDBS

In homogeneous distributed database system, the sites involved in distributed DBMS use the same DBMS software at every site. It might be easier to implement homogeneous systems. [5]

Distributed DBMS can choose to have multiple copies of relations at different sites or choose to have only one copy of a relation. The benefit of data replication is increased reliability – if one site fails, then other sites can perform queries for the relation. As transaction can perform queries from a local site and not worry about network problems, the performance will increase. The problem with data replication is decreased performance when there are a large number of updates, as distributed DBMS have to ensure that each transaction is consistent with every replicated data. This adds additional communication costs to ensure that all copies of the data are updated at the same time.

A homogeneous distributed database system is depicted in Figure 2. This environment is typically defined by the following characteristics (related to the non-autonomous category described previously) [3]:

- ✓ It mainly uses location transparency.
- ✓ Data are distributed across all the nodes.
- ✓ The same DBMS is used at each location.
- ✓ All data are managed by the distributed DBMS (so there are no exclusively Local data).

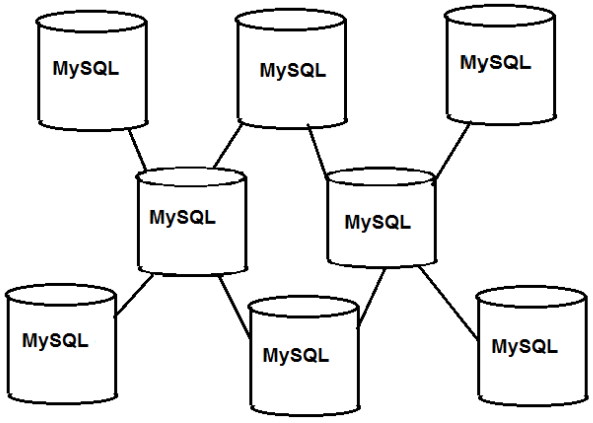

Fig 3. Homogeneous Distributed Database System

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

V. TRANSPARENCY IN HETEROGENEOUS DDBS

In homogenous DDBS, transparency is easy to maintain as only one database is used by all sites. But now a days, the entire world want to get connected with each other for their own conveniences. And every one does not have the same database system. Heterogeneous DDBMS facilitates this need, where different database is used by the sites and different DBMS software can be used at those sites. Transparency is very important in this system as user need not to know what is going on inside the system. The system should be view as single system to them.

Heterogeneous systems are preferable because organizations may have different DBMS installed at different sites and may want to access them transparently. Heterogeneous database systems have well-accepted standards for gateway protocols to expose DBMS functionality to external applications. The gateway protocols help in masking the differences of accessing database servers and bridge the differences between the different servers in a distributed system.

Some characteristics:

✓ As the database may be different at each site, but it is needed to appear a single system to the users, fragmentation transparency is applied.

✓ A connection is maintained between application and database network through middleware.

✓ The programmer writes query considering the system as a single system.

✓ Some process is maintained to transform the local query into global query.

✓ A global query is maintained for all type of database.

VI. CONCLUSION

An ideal distributed database system seems to its clients as centralized system but it consists of many database, so the use of multiple systems should be visible to client as per definition of an ideal distributed system as proposed by Tanenbaum and van Renesse [10].

So, in distributed database system, transparency is very important. Though it is easy to give transparency in homogeneous database system but it is more complex in heterogeneous system. But the requirement of transparency in heterogeneous system is increasing day by day due to use of different databases at different locations are increasing. This paper tries a little to describe the features of transparency in both systems.

REFERENCES

1. Parul Tomar and Megha, “An Overview of Distributed Databases”, International Journal of Information and Computation Technology, ISSN 0974-2239 Volume 4, Number 2 (2014), pp. 207-214.
2. M. Tamer Özsu, Patrick Valduriez, “Principles of Distributed Database Systems”
3. Nitesh Kumar, Saurabh Bilgaiyan , Santwana Sagnika, “An Overview of Transparency in Homogeneous Distributed Database System”, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 10, October 2013
4. O'Brien, J. & Marakas, G.M. (2008) Management Information Systems (pp. 185-189). New York, NY: McGraw-Hill Irwin.
5. George Coulouris, Jean Dollimore, Tim Kindberg, “Distributed Systems Concepts and Design” 3rd edition, Addison-Wesley.
6. M. Tamer Özsu and Patrick Valduriez, “DISTRIBUTED DATABASE SYSTEMS: WHERE ARE WE NOW?” IEEE Computer, Vol. 24, No. 8, August 1991.
7. P.Banupriya and M.Natarajan “An Impression of Transparency in Distributed Database Management System: A Review”, International Journal of Trend in Research and Development, Volume 2(6), ISSN: 2394-9333.
8. Prof. C. M. Jadhav and Bhaskar R Nadargi, “Implementation of Distribution Transparency in Heterogeneous Distributed Database System Using Aglet”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 10, October 2014, ISSN: 2277 128X.
9. George Coulouris, Jean Dollimore, Tim Kindberg, Distributed Systems Concepts and Design, 3rd edition, Addison-Wesley.
10. Michel Banatre, “Hiding Distribution in Distributed Systems”, Proceedings of the 13th international conference on Software engineering May 1991

AUTHOR PROFILE

Fatima Jahan Sarmin studied B.Sc. Engineering in Computer Science and Engineering from Rajshahi University of Engineering and Technology. She has research interest in Database system.

Dr. Md. Mamunur Rashid is Associate Professor at department of CSE in Bangladesh Army University of Science and Technology. He has completed his Ph.D. in Engg. From MITHT, Russia.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)