International Journal of Research in Engineering and Applied Sciences(IJREAS)
Available online at http://euroasiapub.org
Vol. 11 Issue 05, May-2021
ISSN (O): 2249-3905, ISSN(P): 2349-6525 | Impact Factor: 7.196 |

A REVIEW OF DISTRIBUTED AND DATABASE REPLICATION IN REAL-TIME DATABASE SYSTEM

Ashis Ranjan Paul
Research Scholar of Computer Application, CMJ University, Jorabat, Meghalaya
Supervisor Name - Dr Prerna Nagpal (Assistant Professor)

ABSTRACT

Today's Real-Time Systems (RTS) are defined by managing huge volumes of dispersed data making Real-Time distributed data processing a reality. Large firm houses need to conduct distributed processing for several reasons, and they typically must do so in order to stay competitive. So, effective database management methods and protocols for accessing and altering data are needed to meet temporal limits of supported applications. Therefore, new research in Distributed Real-Time Database Systems (DRTDBS) is needed to examine various techniques of adapting database systems technology to Real-Time systems. In this paper the Real-Time Database System, Distributed Database System andReplication in Real-time Database system are discussed.

Keywords: Real-Time Systems (RTS), Distributed, Replication

1. INTRODUCTION

A rising topic in computer science and engineering is real-time computing, due to the increasing use of computers in our daily lives. Several real-time technologies are increasingly being used when people's lives or costly equipment are at risk. Their long-term missions may make upkeep and reconfiguration difficult. Command and control, industrial automation, aeronautical and military systems, telecommunications, etc. are the few examples of Real Time Computing. Due to the demands of their operating environment, such systems are often subject to stringent timing limitations. A real-time transaction may be necessary depending on the severity of the problem (Real-Time).

Logical consistency and timing limits must be considered in Real-time Databases (RTDBs). In the short term, temporally consistent data may be reliable. The difficulty of maintaining temporal consistency in real-time databases is exacerbated by the unpredictable nature of data access. Since recently, researchers in the field of Real-time system development have realized the need of investigating database systems that can handle the time restrictions of reading and updating data. Distributed rather than centralized programme in this article requires communication between applications in different locations to work and this necessitates the usage of distributed Real-time databases. Database connectivity, concurrency management and preserving temporal consistency become more difficult when real-time systems and database systems are combined in the same system. There are additional costs associated with transmitting a request from a remote location. Consequently, accessing and updating local data may be quicker than doing so for remote data replicating data from a remote location and storing it locally can speed up data access and reduce reaction time. It's still necessary to factor in time-sensitive factors like request period and end date while doing replication to make sure the data is genuine.

1.1 Relational and Object-Oriented Database Systems

The relational database model was initially introduced by IBM researcher Dr. E.F. Codd in 1970, and it instantly gained attention for its simplicity. In this view, the database is shown as a Table / Relation. Many common corporate database applications have been successfully constructed using this design. For more complex database applications, however, there are significant drawbacks to this approach. Examples of complex database applications include CAD/CAM & CIM, scientific research, telecommunications, Geographical Information Systems (GIS), multimedia, etc. Complex object hierarchies, extended transaction durations, and user-defined data types for storing graphic objects and massive text items distinguish these systems from traditional commercial applications. Object-oriented databases were required for these applications (OODBs). With an OODB system, you don't have to be constrained by data types and query languages when it comes to meeting these requirements. An advantage of

OODB over relational databases is the ability to specify the structure of complex objects and the methods that may be applied to these objects. Additionally, OODBs are notable due of their strict adherence to object-oriented programming principles. OOP languages, such as C++, SMALLTALK/JAVA, etc., are becoming more popular in software development, making it easier to integrate OODB into object-oriented projects. ORION, IRIS, Open OODB, and others are among the prototypes. Examples of commercial databases are GEMSTONE/OPAL, ONTOS, and Objectivity. The ODMG, a standard framework for handling objects' data, is used extensively across the OODB. For our thesis project, we want to utilize the object-oriented database rather of the relational one, due to the aforementioned advantages.

1.2 Distributed RTOODB

There are many real-time applications that are dispersed internationally rather being centralised at one location, including command and control, military systems, aerospace, telecommunications, and more. As a result, a single database that is both real-time and object-oriented may not be adequate.

1.3 Distributed RTOODB

Real-time object-oriented database systems are needed because of the dispersed nature of many real-time applications (DRTOODB). The advantages of OODB, Real-time, and Distributed Systems are all combined in a DRTOODB. However, it also has its drawbacks. As previously said, in a distributed system, communication between database servers is a crucial consideration. Requests to distant servers are more likely to arrive late when using a DRTOODB because of the higher communication costs involved. As a result, doing the same action on a local item may be faster than performing the same operation on a distant object. As a result, the DRTOODB's ability to satisfy time limitations may be harmed.

Replication is one method of dealing with the difficulty described above. Data is replicated just once in a distributed database system. However, data replication is common in many distributed database systems, with the goal of enhancing availability and speed.

1.4 Distributed Concurrency-Control protocols

Concurrency control techniques in distributed systems without unlimited blocking time and deadlock are discussed in this section. To put it another way, "blocking time" is the amount of time it takes for a lower priority work to take precedence over another task in a given project. Because of their advantages, these protocols will be used to develop JITRT replication algorithms and transactions. Because the priorities of all jobs are known in advance, these protocols may be utilised in hard real-time systems where tasks and resources are already allotted to processors. As of now, these protocols are not appropriate for use in systems with dynamic priority.

1.5 Distributed Priority Ceiling Protocol

DPCP is also known as the Multiprocessor Priority Ceiling protocol (MPCP). A lock's Priority Ceiling determines the order in which all tasks involving that lock are prioritized. Tasks that need the use of a local or distant resource may be distinguished by the fact that they are situated on distinct machines. Tasks that make use of local processors need the utilization of global resources. Tasks requiring global resources run in the global critical part, while those requiring local resources run in the local critical section. This is because global resources are only available in the global critical section. While a global lock protects the whole system, a local lock protects a specific area of the system.

2. DATABASE REPLICATION

For the most part, database systems use replication to improve performance. The goal is to access data locally in order to speed up reaction times and reduce the need to interact with other locations. As a result of database replication, two copies of the database are kept in sync so that they act as closely as possible to each other. Changes in one database are replicated to another database via replication. It makes no difference whether you utilize the first or second database; they are identical, i.e., they are synced. Any number of databases may be replicated under the replication paradigm, not just two. Partial replication is possible, which means that just a portion

of a database's tables or columns or rows are being duplicated. Only those items that have been configured are replicated, thus they are always in sync. It is necessary to maintain numerous copies of data in various places while using replication. Having its own database, each site is able to operate with optimum efficiency and security without having to access databases outside of the local network or workstation.

The procedure of replication does not need any input from the user. As a result, replication should be seen as a low-level procedure that should be entirely unnoticed by the application users themselves. There are no feasible options for replication that include duplicating databases (backup & restore) or utilizing export and import scripts. To put it another way, replication is the practice of transferring information across redundant resources, such software or hardware, in order to ensure that the information is consistent. Figure 1.3 depicts the replication model in its simplest form. There are no tangible copies of data in this paradigm for the user or client. It is a database-distributed-system combination of data replication. As long as each server has its own database, any directory tree or sub-tree may be replicated. If any changes are made to a directory server, they are immediately sent out to all replicas. Computational tasks are often reproduced in time or space, depending on whether they are run on other devices or on the same device periodically. In most cases, access to a replicated entity is the same as access to a single, non-replicated object. An external user should be able to see nothing of the replication process. Following are some of the most critical characteristics of database replication:

1. DatabaseLocality

Data may be accessed quickly even from remote locations because to this database replication feature's ability to keep the database locally up-to-date. Since a result of this, these users will be able to obtain data from local servers rather than faraway ones, as the data access speed is much greater.

International Journal of Research in Engineering and Applied Sciences(IJREAS)
Available online at http://euroasiapub.org
Vol. 11 Issue 05, May-2021
ISSN (O): 2249-3905, ISSN(P): 2349-6525 | Impact Factor: 7.196 |

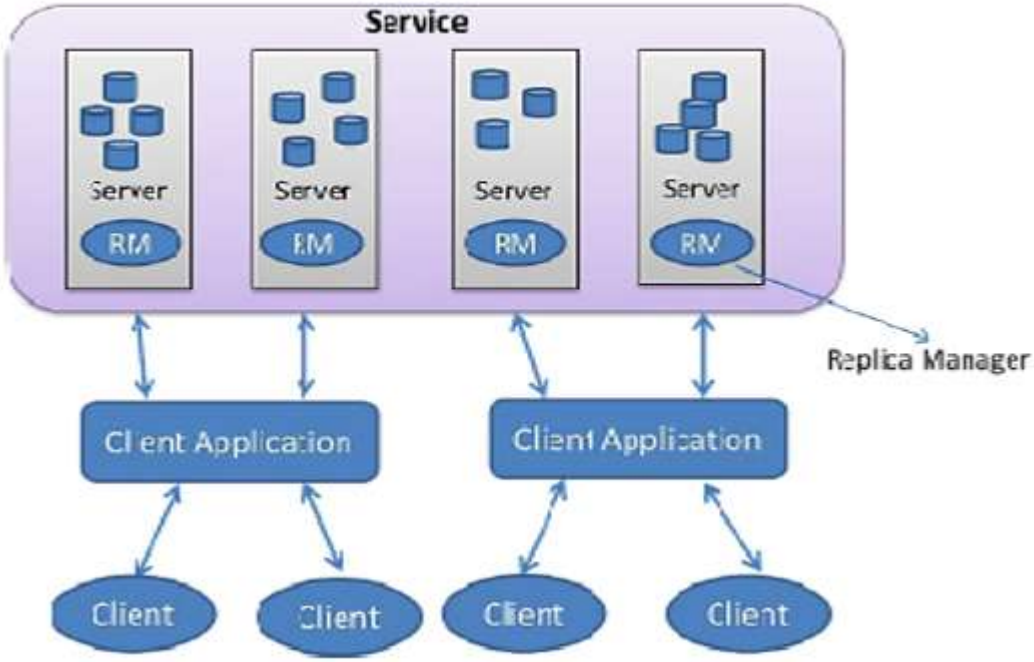

Figure1:Basicdatareplicationmodel

2.Performance

The primary goal of database replication is to increase throughput for both reading and writing operations. It is possible for a single database server to become a bottleneck for a whole system, resulting in sluggish response times and limited request throughput capacity when an application is extensively utilized over a vast network yet the database is kept on a single server. As a consequence of the data being provided by several copies, the system's performance has been improved.

3. AvailabilityandFaultTolerance

Low database system downtime is necessary for high database availability. Two types of downtime exist in a database system: one that is scheduled and one that is unplanned. When all of the software and hardware is being maintained, there is going to be some downtime. There are a variety of reasons for unexpected downtime, including hardware issues, software flaws,

human mistake, and more. Database replication's major focus is reducing downtime in order to improve database availability. Database items stored on a single server may have crashed or are not responding because the server is offline or not responding. Replication of databases may solve this issue and create a fault-tolerant database system in such instances. In the event of a server failure, users may still access their data using a duplicate of the database server. There are at least three methods for replicating a database.

3. DATA REPLICATION IN DISTRIBUTED DATABASES

A DRTOODB relies heavily on data replication. The connectivity costs and data availability issues that plagued DRTOODB are all but eliminated. Synchronous and asynchronous replications are the two main forms of replication. Data should be constant in time regardless of which copy of the object is being accessed or from where it is being accessed in synchronous replication, also known as "Real-Time replication." Because it must take into account the transaction's timeliness, deadline, and priority.Synchronous replication has a high barrier to entry. Because of asynchronous replication, several copies of a same object might temporarily have different values. However, even if this is against distributed data reliance principles, it may be seen as a reasonable compromise in many circumstances. Only real-time replication mechanisms will be developed in this current research.

Constraints on timing and consideration of real-time data's temporal consistency are critical in a DRTOODB. The following are some of the advantages of replication:

1) **Increased availability**: The original data may be retrieved at the other site if the original site is unavailable. Because of this, if we have local copies of faraway items, we are less susceptible to communication failures.

2) **Faster query evaluation**: The use of a local copy of the item instead of a distant site helps speed up queries. As a result, more DRTOODB transactions are able to achieve their deadlines.

3) **Increased Performance:** Replication in DRTOODB reduces transaction response times and improves speed by allowing the data object to be accessed locally. Depending on where and how the copies are made, replication may take on a variety of different meanings. The worst case scenario is when the whole database is duplicated at every site in the distributed system, resulting in a distributed database that is completely replicated. The system will continue to function as long as there is at least one copy available, which might significantly increase the system's availability.

4. CONCEPT AND ABSTRACTION IN DATABASE REPLICATION

5.1 Concept

To better understand how Database Replication works, let's start with the phrase "Replication," which refers to the act of transferring information across redundant resources, such as software or hardware components, in order to increase dependability, fault tolerance, or accessibility. If the same data is replicated over various storage media, it might be data replication, or it could be computation replication, depending on how many times the same computational operation is completed.

Certain replication databases benefit from database mirroring, which increases their availability. Transactional replication and database mirroring can only work together if they are supported by the replication database that is being examined. Combining database mirroring with peer-to-peer replication isn't supported. Connected replication agents may fail over to the mirrored publishing database automatically. Auto-reconnection will take place if there is a problem with a publishing database's connection. The source of data in a replication building block is often a database. It is possible for a single database to serve as the foundation for a slew of replication components. One more replication construction block may have the source database as its destination as well. It's possible to update a common movement set in either data store using the Master-Master Replication pattern, in which the same pair of data stores changes roles (the source becomes the target and the target becomes the source).

4.2 Abstraction

Server processes implement services that are called by client processes in a distributed system. The service specification outlines the collection of invocations that clients may make. When a server process is invoked, its local state is updated, creating a new state. A server's state is considered to be changed in an atomic manner, which means that the state changes resulting from an invocation are not applied in part. A local synchronization method is often used by the server to maintain isolation between concurrent invocations. "One operation" transactions in databases are a good analogy for this concept (e.g., stored procedure). Services are built in several server processes or replicas in order to withstand errors.

In most cases, access to a replicated entity is the same as access to a single, non-replicated object. An external user should be able to see nothing of the replication process. Additionally, in the event of a failure, the failover of replicas is kept as secret as feasible. There are two types of data replication: active and passive.

The concept of a group (of servers) and group communication primitives have been created in order to deal with the complexity of replication. Using a concept like "group," a client may avoid worrying about the number of instances of a replicated service that are being duplicated or even who they are. Using group communication primitives, you can communicate with a large number of people at once. The difficulties of ensuring the consistency of replicated servers are obscured by these semantics. View Synchronous Broadcast (VSB) is a basic form of group communication that is closely related to Atomic Broadcast (ABCAST). FIFO order guarantees are also available for group communication features.

5. SNAPSHOT REPLICATION

It is simple to move data from one database server to another or from one database on the same server to another. By transmitting data in bulk on a regular basis, the snapshot replication process operates. Figure 1.4 shows a server that can function in a read-only mode, as well as a server that can run for a length of time without being updated. Latency is a lapse of time during

which no new data is accessible. Snipping replication software works by copying and pasting files into and out of the distributor's working folder. Subscription servers use snippet files to generate their initial copy of a database using the published data and some additional metadata.

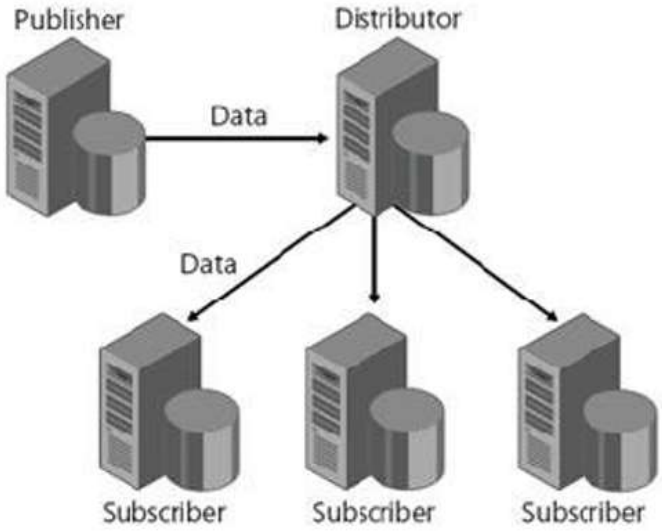

Figure1.4:Schematicofthesnapshotreplication

6. MERGER REPLICATION

Data from many databases is combined into a single database. It is possible for publishers and subscribers to make changes both online and off when data from the publisher is given to subscribers, allowing both parties to make changes at the same time (figure 1.5). In the future, merge replication will allow all of the changes from each site to be integrated into a single, consistent output. In merge replication, there are both standard and bespoke conflict resolution options. The Merge Agent employs a resolution for each conflict to determine whether data should be accepted and propagated to other places. Using Merge Replication is a wonderful concept:

- Updates must be sent to the Publisher and other Subscribers by many Subscribers at various times.
- Changes made offline are then synced with other Subscribers and the publisher.

International Journal of Research in Engineering and Applied Sciences(IJREAS)
Available online at http://euroasiapub.org
Vol. 11 Issue 05, May-2021
ISSN (O): 2249-3905, ISSN(P): 2349-6525 | Impact Factor: 7.196 |

7. TRANSACTIONAL REPLICATION

Users may acquire an initial copy of a database, and then get updates as required. In transactional replication, each successfully completed transaction is duplicated to all subscribers. Transactions can't be sent in batches, thus you can't define a certain time interval for their transmission. Using this kind of replication is beneficial in places with high bandwidth but low latency. If the server is unable to connect for replication, the Transaction Log will grow fast and may become unmanageable (figure 1.5). Snapshots of the original data are taken as the initial stage in transactional replications. The transactions that were copied from it are then updated in that copy. Once a snapshot is created, the user may decide how often it should be updated or if it should be skipped altogether. Log Reader agents are used to read the published database's transaction logs and record new transactions in the distribution database once the first snapshot has been replicated. The publisher's transactions are subsequently transferred to the subscriber's account via the distribution agent.

International Journal of Research in Engineering and Applied Sciences(IJREAS)
Available online at http://euroasiapub.org
Vol. 11 Issue 05, May-2021
ISSN (O): 2249-3905, ISSN(P): 2349-6525 | Impact Factor: 7.196 |

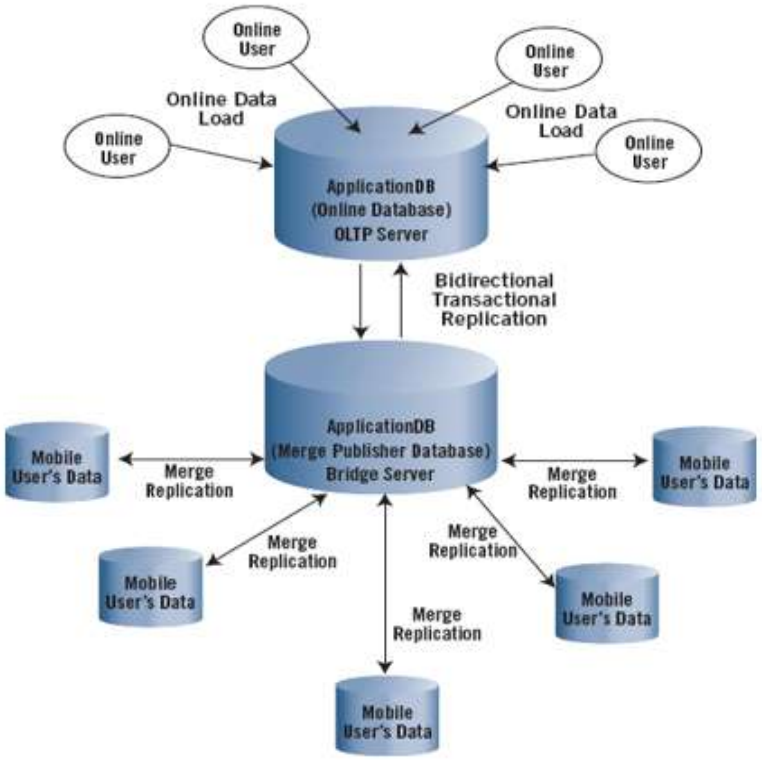

Figure1.5:Schematic of the Merger Replication.

8. AN OVERVIEW OF REPLICATED DATABASE SYSTEM

In today's world, databases and database systems are an integral part of daily life. Most of us come into contact with databases over the course of our daily lives. It's no longer enough to have a good product or service if you don't have rapid access to information and an effective way of managing it. Instead of batch processing, today's commercial apps conduct all their data processing online. On-line support of computer systems and database technology is essential for modern electronic services and e-commerce applications because of their large number of transactions. Database systems (DBS) thus play a vital role in handling and processing today's ever-increasing volumes of data. Functionalities for existing and new applications are needed. Emerging applications need novel processing methodologies in addition to the handling of massive data sources.

It's possible to think of DBS as a repository for data that's been shared by numerous people. Centralized and distributed database systems are two of the most common types of systems. There are two types of database systems: Centralised and Distributed. Centralized DBSs are single computer systems that are connected via a communication network; distributed DBSs are a collection of computer systems that are connected via some communication network; and both types of database systems have their advantages and disadvantages.

These systems are known as Real Time Systems (RTS) because their correctness is not only based on their logical qualities but also on their temporal features. RTS are often connected with life-or-death situations involving people or pricey equipment. A calculation or action that is done too late (or too early) or that employs temporally inaccurate data may be worthless and even dangerous, even if the computation or action is technically correct. A growing number of real-time systems (RTS) applications are becoming more complicated, requiring timely and predictable access to and processing of enormous volumes of real-time data. DRTDBS refers to database systems that are specifically intended for the efficient processing of real-time data of this sort. This means that DRTDBS are a collection of several logically interconnected databases that are dispersed across a computer network where transactions have certain scheduling limitations, generally in the form of deadlines. Transactions in these systems use data pieces that are spread out over several locations. Access to these common data items must be limited in order to maintain the database's logical coherence. It may be difficult to fulfil the time constraints of distinct real-time processes in distributed systems because of the scattered nature of transactions and database integrity requirements.

9. CONCLUSION

Distributed Real Time Database System's (DRTDBS) performance depend on various parameters such as transaction priority, buffer management, replica placement and replacement techniques as well as commit procedures, scheduling transactions with deadlines, Deadlocks, communication delays between different sites as well as predictability and consistency of data access mechanisms and image processing capabilities. It is our goal in this thesis to enhance the

overall performance of the DRTDBS system by addressing some of the most critical concerns such as replica management; priority assignment policy; image processing; and dynamic buffer management. In the next sections, the conclusion of this thesis and the scope for future study are laid forth in further detail.

REFERENCES

[1]. KatemboKitutaEzéchie (2018) on "ANALYSIS OF DATABASE REPLICATION PROTOCOLS", International Journal of Latest Trends in Engineering and Technology Special Issue ICRMR-2018, pp. 075-083 e-ISSN:2278-621X

[2]. S. Kashyap 2016) on "Issues in Distributed Real Time Replication Database System", Corpus ID: 22320159

[3]. B.Johnsirani, M.Natarajan (2015) on "An Overview of Distributed Database Management System", International Journal of Trend in Research and Development, Volume 2(5), ISSN 239333 www.ijtrd.com

[4]. Tarun, S., (2012). A Reputation Replica Propagation Strategy for Mobile Users in Mobile Distributed Database System. International Journal of Grid and Distributed Computing Vol. 5, No. 4.

[5]. Pratik Shrivastava, UdaiShanker (2019) on "Predicting Processing Time of Real Time Transaction in Replicated DRTDBS via Middleware", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249-8958, Volume-8 Issue-5

[6]. Kuppusamy, P.Elango (2013) on "Data Replication for the Distributed Database using Decision Support Systems", International Journal of Computer Applications (0975 – 8887) Volume 69– No.3

[7]. Sashi, K., Thanamani, A.S. (2011). Dynamic Replication in A Data Grid Using a Modified BHR Region Based Algorithm. Future Generation Computer Systems 27, pp. 202-210.

[8]. Singh, V. Rajinder and Singh, Dr.Gurvinder (2011). Optimizing Access Strategies for a

Distributed Database Design using Genetic Fragmentation. International Journal of Computer Science and Network Security, Vol.11 No.6, pp. 180- 183.

[9]. Khanli, L.M., Isazadeh, A., Shishavanc, T.N. (2010). PHFS: A Dynamic Replication Method, To Decrease Access Latency in Multi Tier Data Grid. Future Generation Computer Systems 27, 233–244.

[10]. Manoj Mishra (2009) on "Some Performance Issues in Distributed Real Time Database Systems", vol. 5, No. 6

[11]. Gray, H., Neil, O., Shasha (2008). The Dangers of Replication and a Solution, International Conference on Management of Data. Proceedings of the 1996 ACM SIGMOD international conference on Management of data, Pages: 173 – 182

[12]. Bernstein, Philip A. (2007). Newcomer Eric: Principles of Transaction Processing, Morgan Kaufmann Publishers, Inc.

[13]. Abawajy, Deris, J., Omer, M. (2006). A Novel Data Replication and Management Protocol for Mobile Computing Systems. Mobile Information Systems, 2(1): 3-19, IOS Press, Netherlands

[14]. Hellerstein, J.M., Stonebraker, M., "Readings in Database Systems", The MIT Press, (ISBN 0-262-69314-3, Library of Congress Control Number: 2004113624), pp:1-8, 2005.

[15]. Błażewicz, J., Królikowski, Z., Morzy, T., "Handbook on Data and Management in Information Systems", Springer, (ISBN 3-540-43893-9), pp: 1-18, 2003.

****