

Data Replication for the Distributed Database using Decision Support Systems

Kuppusamy, PhD.
Department Of Computer Science and
Engineering
Alagappa University ,karaikudi
Tamilnadu India

P.Elango
Department of Computer Science
Gobi Arts And Science College
Gobichettipalayam, Erode
Tamilnadu, India

ABSTRACT

Replication is a topic of interest in the distributed computing, distributed systems, and database communities. Decision support systems became practical with the development of minicomputer, timeshare operating systems and distributed computing. Replicated data may get insufficient due to system failure, fault tolerance, and reliability. A partial Replication is quantized in the replication system will increase the non replicated system. Fault tolerance is the property that enables a system (often computer-based) to continue operating properly. Transaction Processing Replication (TP-R) and Decision-support replication schema (DDS-R) will clear the non replica and it is used to clear the server problems and system error. This process is well executed in distributed systems and it doesn't fail to detect the system errors when multiple access are multiplexed.

1. INTRODUCTION

Replication is the process of sharing information so as to ensure consistency between redundant resources, such as software or hardware components, to improve reliability, fault-tolerance, or accessibility. It could be data replication if the same data is stored on multiple storage devices or computation replication if the same computing task is executed many times. It is the process of automatically distributing copies of data and database objects among SQL Server instances, and keeping the distributed information synchronized. Replication is the process of sharing information so as to ensure consistency between redundant resources, such as software or hardware components, to improve reliability, fault tolerance, or accessibility. It could be data-replication if the same data is stored on multiple storage devices or computation replication if the same computing task is executed many times [10]. The secure sharing of information in this type of environment is a complex problem. The owners of the different data sources will have different policies on access to and the dissemination of the data that they hold [12]. There are two main types of replication protocols: active replication, in which all replicas processes concurrently all input messages. Passive replication, in which only one of the replicas processes all input messages and periodically transmits its current state to the other replicas in order to maintain consistency [7]

From the past years, Distributed Databases have taken attention in the database research community. Data distribution and replication offer opportunities for improving performance through parallel query execution

and load balancing as well as increasing the availability of data [2]. In a distributed database system, data are often replicated to improve reliability and availability, thus increasing its dependability. In addition, data are also stored on computers where it is most frequently needed in order to reduce the cost of expensive remote access. [3]. Decision making involves processing or applying information and knowledge, and the appropriate information/knowledge mix depends on the characteristics of the decision making context. Most decisions fall somewhere in the middle, and for those cases human decision making capability can be supported and enhanced by the use of a decision support system [9]. The process of decision making depends on many factors, including "the context in which a decision is made, the decision maker's way of perceiving and understanding cues, and what the decision maker values or judges as important" [11].

The replication or migration of shared data blocks at arbitrary locations on chip require the use of directory or broadcast-based mechanisms for lookup and coherence enforcement, as each block is likely to have different placement requirements [18]. The usability of a storage system is dependent on its scalability in many cases. Whenever a very large amount of data items is to be stored, or the amount of requests to the store exceeds the capabilities of stand-alone systems, a logical architectural choice is the distribution of the stored data over several physical computers. If comparably few data items are served to a large number of requests, replication is to be used [5]. Replicating the stock and client related data at these different locations is desirable since it provides fast access to the local replica, and helps to survive disaster cases where all machines of a physical location crash [6]. The Personalized Search team originally built a client side replication mechanism on top of big table that ensured eventual consistency of all replicas. The current system now uses a replication subsystem that is built into the servers [20].

The Name node maintains the file system namespace and records any changes made to it. It also keeps track of the number of replicas of a file that should be maintained in the HDFS typically called the replication factor [4]. Continuous Timestamp based Replication Management (CTRM), which deals with the efficient storage, retrieval and updating of replicas in DHTs. In CTRM, the replicas are maintained by groups of peers, called replica holder groups, which are dynamically determined using a hash function [1]. The resources required by a user to perform an activity should be reachable; thus, they must be locally stored through a replication mechanism. Replication of resources increases the users' autonomy but it also adds inconsistency [16]. Read-

only universally-shared blocks (e.g., instructions) are prime candidates for replication across multiple tiles; replicas allow the blocks to be placed in the physical proximity of the requesting cores, while the blocks' read-only nature obviates coherence [17]

Network bandwidth is a scarce resource in a wide-area distributed storage system. To store objects durably, there must be enough network capacity to create copies of objects faster than they are lost due to disk failure [8]. Increasing the replication level does not help tolerate a higher average permanent failure rate, but it does help cope with bursts of failures. Reintegrating returning replicas is key to avoiding unnecessary copying. The process of breaking the system down into components is called partitioning. The process of allocating the components (partitions) around the network is called allocation. The allocation process usually has the goal of minimizing inter process communication cost, minimizing execution cost, load balancing, increasing system reliability and providing scalability [13]. Range partitioning may be able to do a better job, but this requires carefully selecting ranges which may be difficult to do by hand. The partitioning problem gets even harder when transactions touch multiple tables, which need to be divided along transaction boundaries [15].

It is a computer-based information system that supports business or organizational decision-making activities. DSSs serve the management, operations, and planning levels of an organization and help to make decisions, which may be rapidly changing and not easily specified in advance. Decision support systems (DSS) are a coherent set of instruments used in decision-making process. Taking into account multiple criteria simultaneously, led to the classification of issues studied in several classes of problems, solving a class of problems is done by using increasingly more methods and techniques, work procedures based on data and information processing [14]. It is thus crucial to ensure that database systems work correctly and continuously even in the presence of a variety of unexpected events. The key to ensuring high availability of database systems is to use replication. While many methods for database replication, most of these solutions only tolerate silent crashes of replicas, which occur when the system suffers hardware failures, power outages, etc [19]. DDS-R is used to control the sensitivity flow of data, it can be well utilized, and the database of high availability can be handled by using a constant trade-off between consistency and efficiency.

2. RELATED WORKS

Numerous researches have been proposed by researchers that the replication for the distributed database using decision support systems. In this section, a brief review of some important contributions from the existing literature is presented.

Peter A. Boncz *et al* [26] have proposed the P2P paradigm was a promising approach for distributed data management, particularly in scenarios where scalability was a major issue or where central authority/coordinators was not available solution. P2P data management had several dimensions affecting the design, the capabilities, as well as the limitations of the system. In that report, they have sketched a set of important dimensions. Furthermore, based on own experiences they discussed representative application examples which show the potential of P2P databases. It turned

out that there were a lot of different interpretations of the term "P2P Databases," depending on the research context. Also, the distinguishing characteristics against distributed and federated databases were not always strict. In the discussion, they strived to clarify those notions.

Recently the cloud computing paradigm has been receiving significant excitement and attention in the media and blogosphere. To some, cloud computing seems to be little more than a marketing umbrella, encompassing topics such as distributed computing, grid computing, utility computing, and software-as-a-service, that have already received significant research focus and commercial implementation. Nonetheless, there exist an increasing number of large companies that are offering cloud computing infrastructure products and services that do not entirely resemble the visions of these individual component topics.

Daniel J. Abadi [22] discussed the limitations and opportunities of deploying data management issues on those emerging cloud computing platforms. They speculate that large scale data analysis tasks, decision support systems, and application specific data marts were more likely to take advantage of cloud computing platforms than operational, transactional database systems. It present a list of features that a DBMS designed for large scale data analysis tasks running on an Amazon-style offering should contain. They then discussed some currently available open source and commercial database options that can be used to perform such analysis tasks, and conclude that none of those options, as presently architected, match the requisite features. They thus expressed the need for a new DBMS, designed specifically for cloud computing environments.

Iraj Mahdavi *et al* [21] described a simulation-based decision support system (DSS) to production control of a stochastic flexible job shop (SFJS) manufacturing system. The controller design approach was built around the theory of supervisory control based on discrete-event simulation with an event-condition-action (ECA) real-time rule-based system. The proposed controller constitutes the framework of an adaptive controller supporting the co-ordination and co-operation relations by integrating a real-time simulator and a rule-based DSS. For implementing SFJS controller, DSS receives online results from simulator and identifies opportunities for incremental improvement of performance criteria within real-time simulation data exchange (SDX). A bilateral method for multi-performance criteria optimization combines a gradient based method and the DSS to control dynamic state variables of SFJS concurrently. The model was validated by some benchmark test problems.

Abbe Mowshowitz *et al* [24] have proposed that the Classical work on query optimization had not taken account of the topology of distributed database networks as a cost factor in executing standard operations in relational algebra. Here the report research findings designed to remedy that deficiency. In particular, they examined the relative costs of query optimization (a) in a network whose topology (e.g., a hypercube) was known and (b) in a network whose topology was unknown. The critical factor in the advantage of a well defined topology was that the cost of determining pair wise distances between the nodes involved in a joint operation was substantially lower than it was in a network whose topology was unknown. What was more the cost of building and maintaining a hypercube was comparable to the management costs in a random network based on preferential attachment.

Narasimhaiah Gorla [25] stated that Minimization of query execution time was an important performance objective in distributed databases design. While total time was to be minimized for On Line Transaction Processing (OLTP) type queries, response time have to be minimized in Decision Support type queries. Thus different allocations of sub queries to sites and their execution plans were optimal based on the query type. They formulate the subquery allocation problem and provide analytical cost models for those two objective functions. Since the problem was NP-hard, they solve the problem using genetic algorithm (GA). The results indicate query execution plans with total minimization objective were inefficient for response time objective and vice versa. The GA procedure was tested with simulation experiments using complex queries of up to 20 joins. Comparison of results with exhaustive enumeration indicates that GA produced optimal solutions in all cases in much less time.

Jon Olav Hauglid *et al* [27] stated that in distributed database systems, tables were frequently fragmented and replicated over a number of sites in order to reduce network communication costs. How to fragment, when to replicate and how to allocate the fragments to the sites were challenging problems that had previously been solved either by static fragmentation, replication and allocation, or based on a priori query analysis. Many emerging applications of distributed database systems generate very dynamic workloads with frequent changes in access patterns from different sites. In such contexts, continuous refragmentation and reallocation can significantly improve performance. In that paper they presented a DYFRAM, a decentralized approach for dynamic table fragmentation and allocation in distributed database systems based on observation of the access patterns of sites to tables. The approach performs fragmentation, replication, and reallocation based on recent access history, aiming at maximizing the number of local accesses compared to accesses from remote sites. They showed through simulations and experiments on the DASCOSA distributed database

system that the approach significantly reduces communication costs for typical access patterns, thus demonstrating the feasibility of their approach.

Rajinder Singh Virk and Dr. Gurvinder Singh [23] have demonstrated that a key component of any Relational Distributed Database Query Optimizer was to fragment various tables and distribute fragmented Data over the sites of network. Then find a near optimal or best possible sub query operation allocation plan in a stipulated time period. In that paper they have proposed a Genetic Algorithm (GA) for finding near optimal fragmentation plan for selecting the various nodes or sites for placing recursively the vertically fragmented data attributes in two components for a Query Transaction on the Database. They discussed the advantages of using proposed Genetic Algorithm (PGA) over various other prevalent Algorithms and unpartitioned case. An experimental result for a simulated Distributed Database over a Wide Area Network shows encouraging results for the use of PGA over other techniques.

3. DATA REPLICATION

Replication” is the process of sharing information to ensure consistency between redundant resources such as software or hardware components to improve reliability, fault-tolerance, or accessibility. It could be data replication if the same data is stored on multiple storage devices, or computation replication if the same computing task is executed many times [29]. Replication has been studied in many areas, especially in distributed systems (mainly for fault tolerance purposes) and in databases (mainly for performance reasons) [28]. Replication is one of the oldest and most important topics in the overall area of distributed system. An important issue in distributed systems is the replication of data. Data are generally replicated to enhance reliability or improve performance. Replication is the process of copying data from a data store or file system to multiple computers to synchronize the data. Database replication is becoming more important role for database applications.

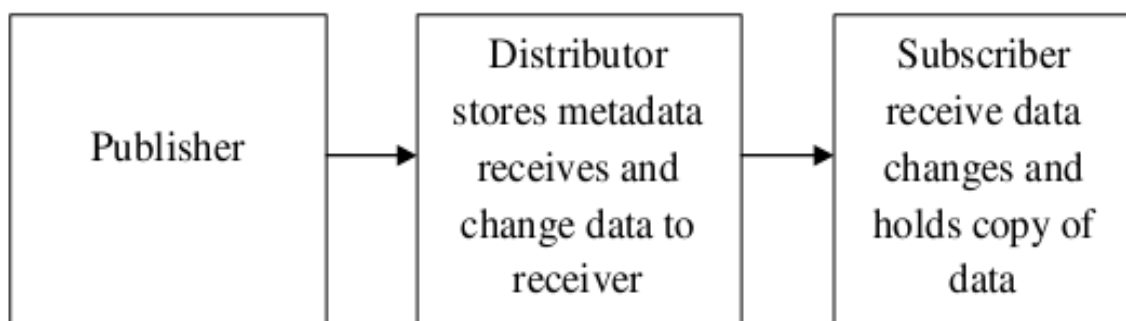


Fig. 1 Replication Process

Replicated data are becoming more and more of interest lately.

Replication is a cost effective way to increase availability and used for both performance and fault tolerant purposes thereby introducing a constant tradeoff between consistency and efficiency. Replication provides a backup database large enterprises usually have sites where it is imperative to access

data continuously. If a server collapses it is important to have access to the same data on a different server and this usually requires administrative intervention [30]. Database replication is quickly becoming a critical tool for providing high availability, survivability, and high performance for database applications. However, to provide useful replication one has to solve the non-trivial problem of maintaining data

consistency between all the replicas [31]. Replication uses the following three servers namely Publisher, Distributor, and Subscriber

The basic problem with data replication is that an update to any given logical object must be propagated to all stored copies of that object. A difficulty that arises immediately is that some sites holding a copy of the object might be unavailable (because of a site or network failure) at the time of the update. The obvious strategy of propagating updates immediately to all copies is thus probably unacceptable, because it implies that the update and therefore the transaction will fail if any one of those copies is currently unavailable. In a sense, in fact, data is less available under this strategy than it would be in the non-replicated case.

There are two primary reasons for replicating data: reliability and performance. First, data are replicated to increase the reliability of a system. If a file system has been replicated it may be possible to continue working after one replica crashes by simply switching to one of the other replicas. Also, by maintaining multiple copies, it becomes possible to provide better protection against corrupted data. Reliability is a vast domain: a considerable amount of techniques allow to account for the existence of faults, and to guarantee – to a certain extent – the continuation of computations. A sub domain of reliability, fault-tolerance aims at allowing a system to survive in spite of faults, i.e. after a fault has occurred, by means of redundancy in either hardware or software architectures [7]. The other reason for replicating data is performance. Replication for performance is important when the distributed system needs to scale in numbers and geographical area. Scaling in numbers occurs, when an increasing number of processes needs to access data that are managed by a single server. Replication provides fast, local access to shared data because it balances activity over multiple sites. Some users can access one server while other users access different servers, thereby reducing the load at all servers. Also, users can access data from the replication site that has the lowest access cost, which is typically the site that is geographically closest to them [10].

A distributed system is a set of services used by server processes and invoked by client processes. The model and the communications abstractions used by replication protocols in distributed systems, and present some replication techniques that have been processed in distributed systems. The mostly used systems are DDS-R - (Decision Supporting Replication) and TP-R (Transaction Processing Replication).

4. ANALYTICAL MODEL FOR PARTIAL REPLICATION

Let us assume that a non replicated database has a processing capacity C , that a non-replicated database can execute C transactions per time unit. All sites have the same capacity, a non-replicated database in which the entire processing capacity C is used for executing local transactions, but sites in the replicated database need to use some of its processing capacity C for coordination with other sites. We term the coordination work as remote work.

So, each site i in a replicated database uses a fraction of its processing capacity for local work Li and the remaining capacity for remote work Ri , i.e. $C=Li+Ri$ then, the local work performed in a site i is

$$Li=c-Ri \quad (1)$$

The scale out is the sum of the amount of local work executed at each site, divided by the processing capacity of a non-

replicated database $Scaleout = \frac{\sum_{i=1}^n Li}{C}$ i.e. how

many times the capacity of a non-replicated system is increased when it is replicated. The more local work Li each site executes, the better the scalability of the system. The total amount of local work Li at site i is the sum of accesses to objects O_k stored at i (Eq. 2). The objects stored at site i are defined by the function $r(i,k)$

$$L_i = \sum_{k=1}^0 C.r(i,k).a_i k, \forall i=1\dots n \quad (2)$$

Since our model uses asymmetric processing, writes on an object O_k at site i impose some (remote) work in the other sites that also store a copy of O_k . We call this fraction of remote work the writing overhead, $wo, 0 \leq wo \leq 1$. Therefore, the amount of remote work, Ri , at site i is a fraction wo of the writes on every object O_k stored at site i that are executed at the rest of the sites that store a copy of that object (Eq. 3).

$$R_i = wo \cdot \sum_{j=1, j \neq i}^n \sum_{k=1}^0 r(i,k).r(j,k).a_{jk}.u_{jk} \quad (3)$$

Therefore, replacing Li and Ri in Eq. 1 with the expressions in Eq. 2 and Eq. 3 we obtain

$$\sum_{k=1}^0 c.r(i,k).a_{ik} = c_i - wo \cdot \sum_{j=1, j \neq i}^n \sum_{k=1}^0 r(i,k).r(j,k).a_{jk}.u_{jk}, \forall i = 1 \dots n \quad (4)$$

However, maximizing the amount of local work (Li) each site processes may lead to saturation of other sites. For instance, let us assume that there are three sites $\{B1, B2, B3\}$ and two objects $\{o1, o2, \}$ with writing overhead $wo = 0.60$. $B1$ Stores $\{o1\}$, $B2 = \{o2\}$ and $B3 = \{o1, o2, \}$ i.e. $B1$ and $B2$ are partial replicas and $B3$ is a full replica. Furthermore, $B1$ and $B2$ use their entire processing capacity for local work $Ri=o, Li=c$ at both sites. The amount of remote work at $B3$ is $R3 = C + C * 0.60 = 1.2C$. That is, the amount of remote work at $B3$ surpasses its processing capacity, and therefore, that site is saturated. In order to avoid saturation of some sites, some of the remaining sites cannot use their entire capacity, e. g. in the previous example, if $B1$ and $B2$ use only $\frac{2}{3}$ of their

capacity, $R3 = (\frac{2}{3}c + \frac{2}{3}c)$ ($0.60 = C$), $B3$ could process all remote work. We model this fact by reducing the capacity of sites. Ci ($0 < ci \leq 1$) is the percentage of the capacity site i uses to prevent saturation of other sites. So, equation 4 i

$$\sum_{k=1}^0 c.r(i,k).a_{ik} = c.c_i - \omega_0 \cdot \sum_{j=1, j \neq i}^n \sum_{k=1}^0 r(i,k).r(j,k).a_{jk}.u_{jk}, \forall i \in \{1..n\}$$

(5)

Moreover, the model must take into account the work-load (A,U) and guarantee that the amount of accesses to each object O_k corresponds with the proportion. A_k So, each A_k should be equal to the fraction of accesses to O_k with respect the global number of accesses to all objects:

$$A_k = \frac{\sum_{i=1}^n c.r(i,k).a_{ik}}{\sum_{i=1}^n \sum_{k=1}^0 c.r(i,k).a_{ik}}, \forall k \in \{1..0\}$$

(6)

A similar expression is defined for writes u_k , which are a proportion of writes of the total number of accesses for an object A_k

$$A_k.U_k = \frac{\sum_{i=1}^n c.r(i,k).a_{ik}.u_{ik}}{\sum_{i=1}^n \sum_{k=1}^0 c.r(i,k).a_{ik}.u_{ik}}, \forall k \in \{1..0\}$$

(7)

Finally, the scale out is the sum of the amount of local work executed at each site, divided by the processing capacity of the non-replicated database (Eq. 8).

$$scale_out = \frac{1}{C} \sum_{i=1}^n \sum_{k=1}^0 c.r(i,k).a_{ik}$$

(8)

So, given a replicated database of n sites with a replication schema r and a load (A,U) we look for the values Ci, a_{ik}, u_{ik} that maximize the scale out solving the optimization problem

$$\max \frac{1}{C} \sum_{i=1}^n \sum_{i=1}^n c.r(i,k).a_{ik}$$

(9)

Subject to Eq. 5, 6, 7 and the domain of variables

$$0 \leq ci \leq 1; 0 \leq a_{ik} \leq 1; 0 \leq u_{ik} \leq 1; \forall i \in \{1..n\}; \forall k \in \{1..o\}$$

The analytical model is to understand the potential scalability gains of partial replication with respect to full replication. The

model quantizes the scale out, which determines how many times the replicated system, increases the performance of a non replicated system. This process can able to gain the stability of the Transaction Processing Replication (TP-R) and Decision-support replication schema (DDS-R) by clearing the non replica which may used to clear the server problems and system errors.

Fault tolerance plays a vital role to deviate the errors and enables the system (often computer-based) to continue operating properly in the event of the failure of (or one or more faults within) some of its components. If its operating quality decreases at all, the decrease is proportional to the severity of the failure, as compared to a naively-designed system in which even a small failure can cause total breakdown.

Fault tolerance is the ability of a system to perform its function correctly even in the presence of internal faults. The purpose of fault tolerance is to increase the dependability of a system. A complementary but separate approach to increasing dependability is fault prevention. This consists of techniques, such as inspection, whose intent is to eliminate the circumstances by which faults arise. The theory on fault-tolerant mechanism for distributed systems is based on different kind of networks, such as LAN, WAN. Since these networks are providing various services over the network, it makes communications between the users entrusted. So, the algorithm, which provides reliable services and scalability of the resources, is needed to be designed. In fact, scalability is achieved by a fully decentralized algorithm, in which the dynamically available resources are managed through a membership protocol. On other side, fault tolerance is assured in meaning of that the loss of up to all but one resource will not affect the quality of the solution.

4.1. TRANSACTION PROCESSING REPLICATION (TP-R)

Transactional replication is a type of replication that allows data modifications to be propagated incrementally between servers in a distributed environment. Transactional replication can be used for many different applications, from reporting servers and data warehousing environments to Web servers and e-commerce applications. Transactional replication is used at many of the predominant Web sites on the Internet that run SQL Server, including MSN.com, Passport.com, Barnes and Noble.com, and Buy.com. Transactional replication is a scalable and reliable solution for distributing data in high-performance environments. A transaction processing replication (TP-R) approach that can maintain near real time transaction integrity at data copy sites is essential. In lazy replication, transactions are executed first at one replica. Any updates are propagated to other replicas only after transaction commits, thus providing fast response times [32].

The notion of transaction was first introduced in database systems, with the objective of supporting the consistent execution of concurrent operations over shared data. However, since then transactions have been applied much more broadly, e.g., in distributed systems in many application scenarios, in which they improve reliability and guarantee data consistency. This is about the transaction commit for replicated databases. There is work in this area for some years, with authors suggesting the utilization of abstractions, commonly used to specify reliable distributed systems (e.g., consensus, total order multicast) to support crash fault-tolerant database replication or, more generically, transaction processing. TP-R

replication is primarily concerned with creating a single image of a database across distributed autonomous sites and preserving database integrity in near real-time processing. The overall integrity of databases is preserved by forwarding data changes resulting from single user transactions.

4.2 TP-R REPLICATION: PEER TO PEER & MASTER/SLAVE APPROACHES

TP-R replication is primarily concerned with creating a single image of a database across distributed autonomous sites and preserving database integrity in near real time processing. The overall integrity of databases is preserved by forwarding data changes resulting from single user transactions.

].

All data replication regardless of vendors can copy data from sources to targets. Master/slave approaches replicate data from master to slave, requiring updates to successfully complete at the master before the transaction is considered a success. On the other hand, updates in peer to peer approaches can be made to any data location and then copied into other locations. A transaction is successfully completed as soon as anyone or combination of locations is able to update one complete copy of the affected data. Peer to peer allows all locations to own and manipulate any data, broadcasting changes as required. This requirement shouldn't significantly reduce the scalability of the system if the rate of queries is much higher than the rate of updates, which is commonly the case for example in name services or knowledge base systems [32]

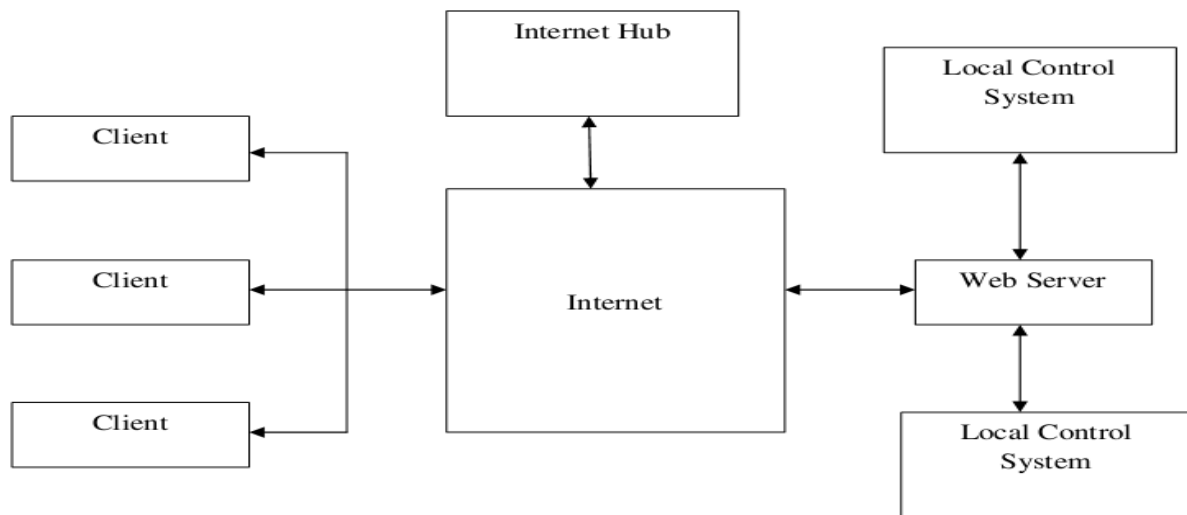


Fig. 2 Internet Based Control System

In the master/slave architecture every table or table fragment is assigned to a primary site. If the primary table's database server fails or access to that server from the network (where a transaction updating that table has occurred) is denied, replication doesn't occur and the transaction is queued. This can present a problem for remotely generated transactions because those processes cannot update their local or other sites, until they are first routed synchronously through their primary tables.

4.2.1 CASCADING REPLICATES

Cascade behavior is a Pareto improvement over behavior in which individual's base decisions only on their private information, since a cascade reflects an integration of more private information than any single individual possesses [34].

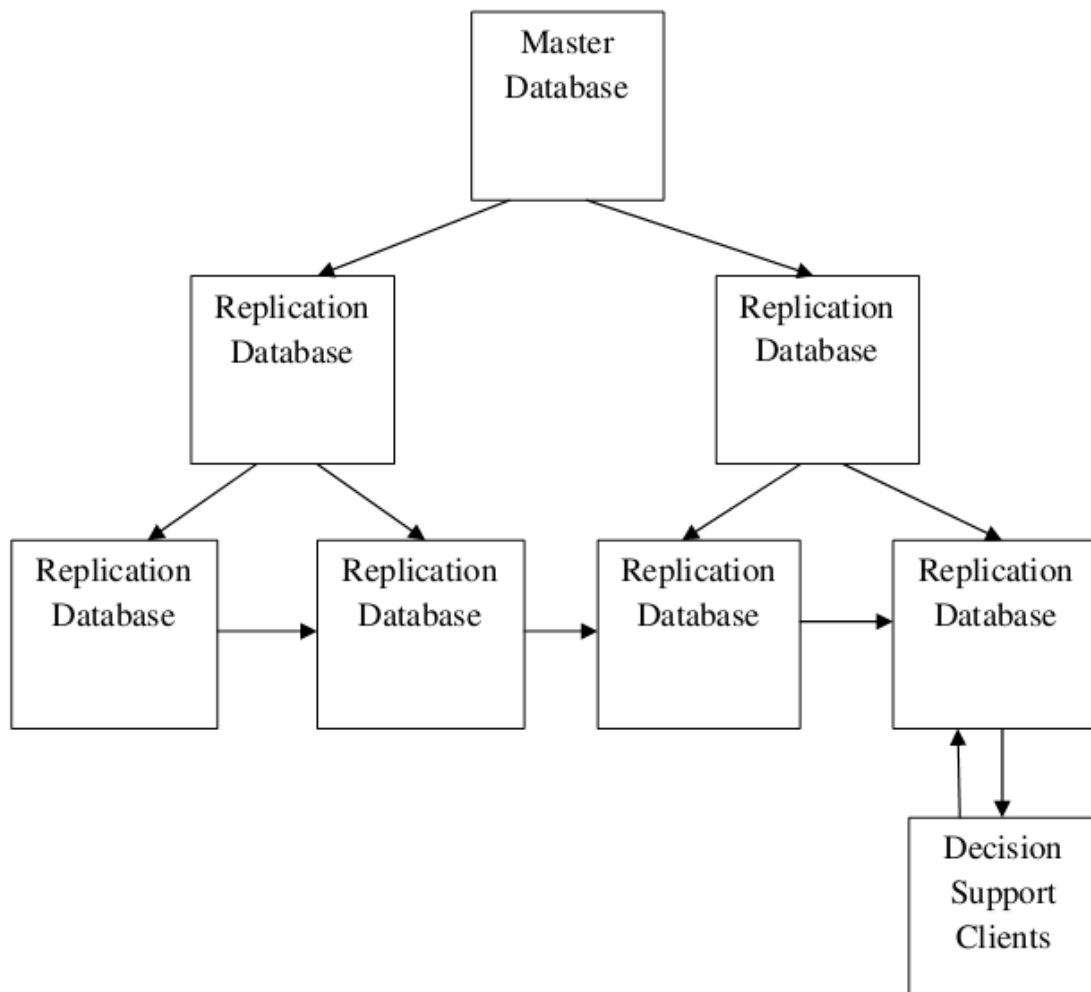


Fig. 3 Cascading database replication

A common application model is one where there are 100's or 1,000's of database servers (e.g. in branch offices) fed from a smaller number of distribution nodes, which are in turn fed from central host source. Efficient distribution requires support for cascading replicates where copies can be made from other copies. For example, the central host distributes to 20 distribution nodes each of which distributes to 20 branch offices. The replication system must distribute consistent data across each of the 400 branches (perhaps at the end of the business day), but at the same time, subset the data for branch related operations.

4.3 DECISION-SUPPORT REPLICATION SCHEMA (DDS-R)

DSS-R approaches to replication usually are built on various technology variations of table copying. The typical decision support application has a requirement for consistent period data sources and not necessarily for data that is up-to-the-minute current. DSS-R approaches, then, don't typically worry about keeping the data current. Consistent, stable data for a given period is the highest requirement for these types of applications. The decision support systems are tuned for query processing, typically by adding more indexes. In this case, then, continuous propagation of updates would interfere with

the ability of the query tool to provide reasonable performance.

The replication server should provide various timing options which can create copies based on timed events (clock or interval), on application events (e.g. end of day reconciliation completed), or on manual request. Other important requirements for decision support include the ability to access legacy production system data from sources such as IMS, RMS, VSAM, and flat files and to provide sophisticated data manipulation/enhancement to that data.

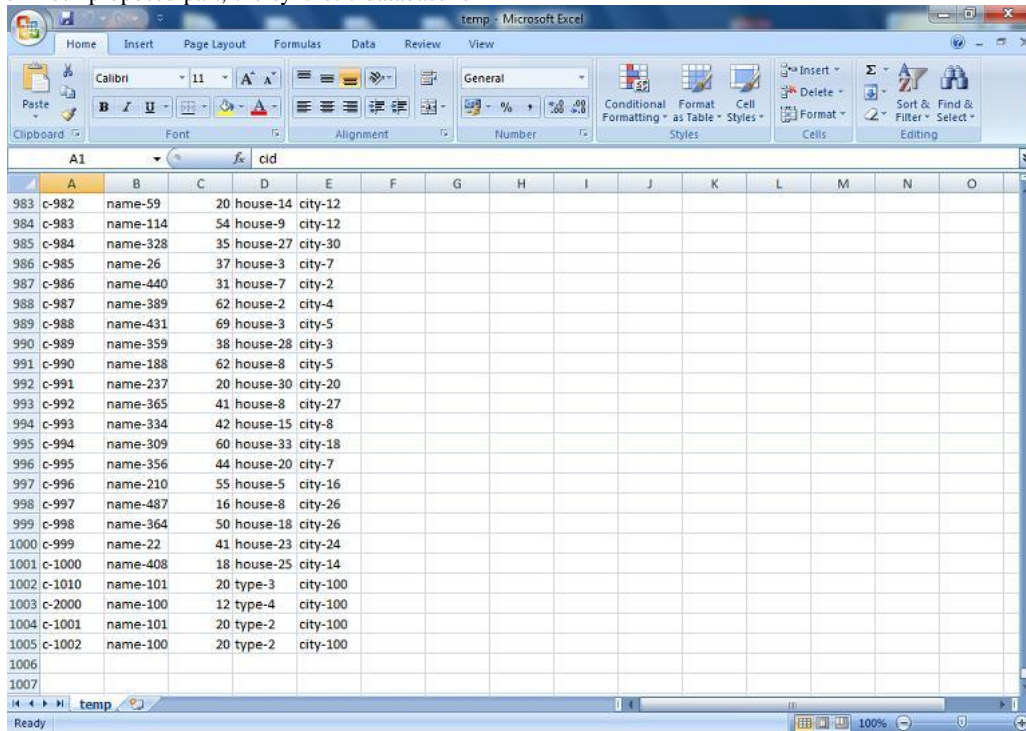
4.3.1 DSS-R SCHEMA

The value added to the data by manipulation or enhancement is very important in DSS-R environments. Sources are typically legacy systems and the replication solution should provide the ability to restructure the data from legacy formats into the relational model. Tools should provide support for joining data from multiple sources, for calculating new values, for aggregating data and for transforming encoded data into descriptive forms. An important side point to keep in mind is that one of the principal benefits of DSS-R, aggregation of data or de normalization, is something that should not be done when the replicate is updatable.

5. RESULTS AND DISCUSSIONS

In our proposed methodology, the data replication for decision supporting system is described. The implementation was done in JAVA. Here in our proposed part, the synthetic database is

used. The step by step results obtained from the proposed part is described as follows.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
983	c-982	name-59	20	house-14	city-12										
984	c-983	name-114	54	house-9	city-12										
985	c-984	name-328	35	house-27	city-30										
986	c-985	name-26	37	house-3	city-7										
987	c-986	name-440	31	house-7	city-2										
988	c-987	name-389	62	house-2	city-4										
989	c-988	name-431	69	house-3	city-5										
990	c-989	name-359	38	house-28	city-3										
991	c-990	name-188	62	house-8	city-5										
992	c-991	name-237	20	house-30	city-20										
993	c-992	name-365	41	house-8	city-27										
994	c-993	name-334	42	house-15	city-8										
995	c-994	name-309	60	house-33	city-18										
996	c-995	name-356	44	house-20	city-7										
997	c-996	name-210	55	house-5	city-16										
998	c-997	name-487	16	house-8	city-26										
999	c-998	name-364	50	house-18	city-26										
1000	c-999	name-22	41	house-23	city-24										
1001	c-1000	name-408	18	house-25	city-14										
1002	c-1010	name-101	20	type-3	city-100										
1003	c-2000	name-100	12	type-4	city-100										
1004	c-1001	name-101	20	type-2	city-100										
1005	c-1002	name-100	20	type-2	city-100										
1006															
1007															

Fig.4: Initial table before adding records

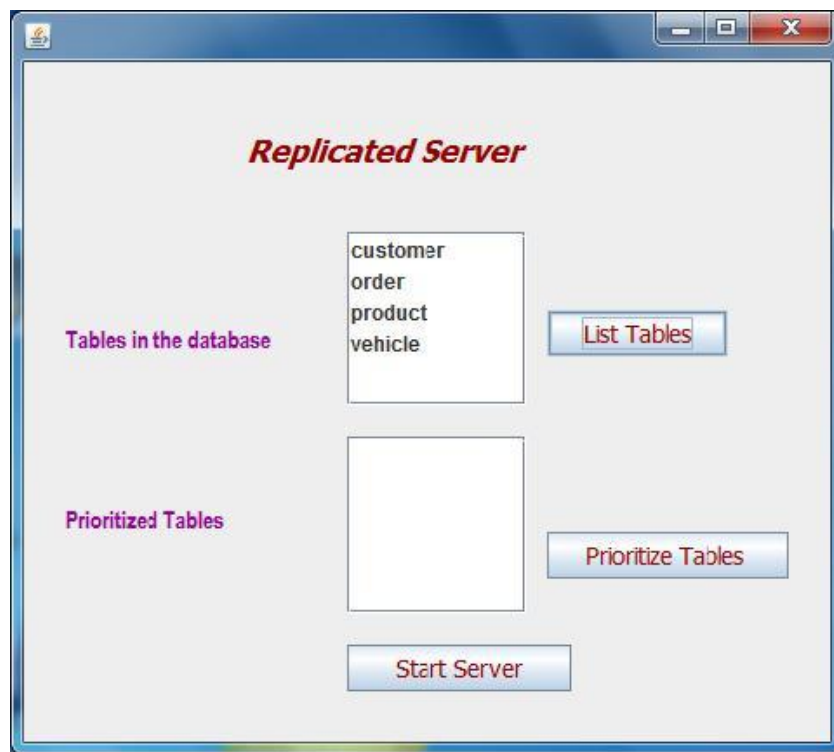


Fig. 5: List of tables in Server

The initial database used for data replication in server is described in the fig. 4. In the server, more than one table is used. Based on the prioritization, the high prioritized table is replicated in client which is shown in fig.5 and Fig. 6.



Fig. 6: Prioritized order of tables



Fig. 7: Query Processing in Client

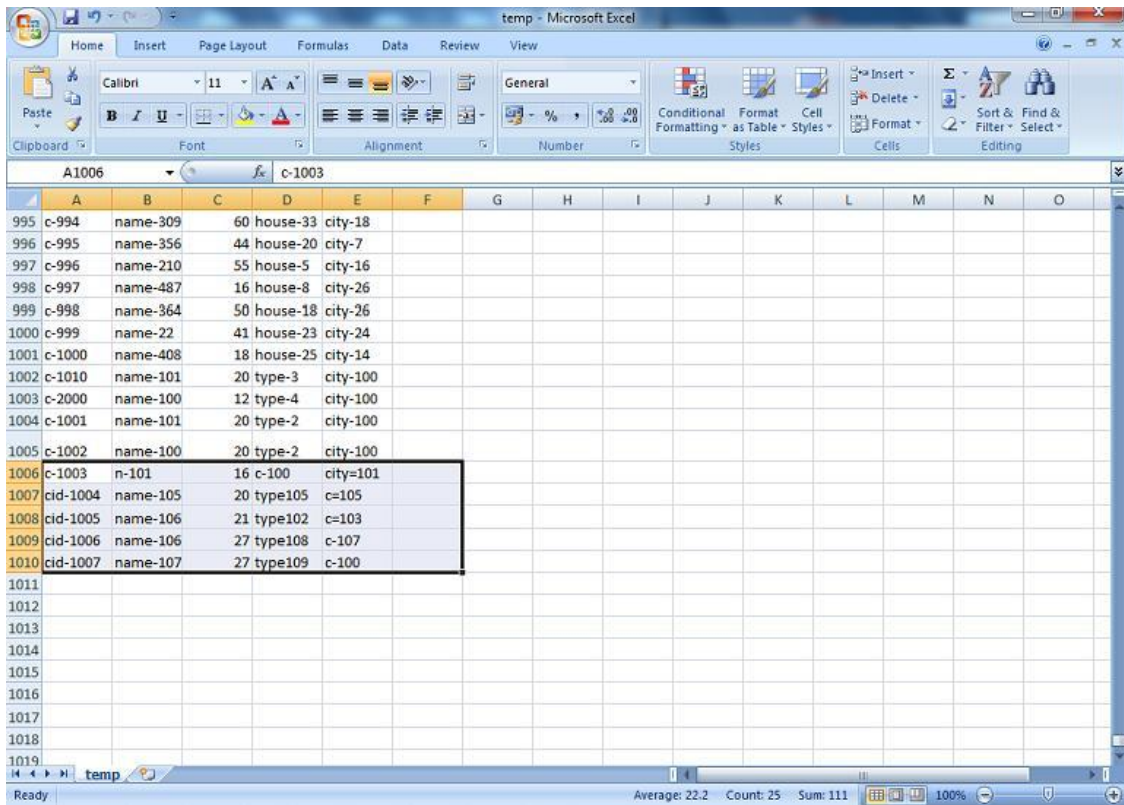


Fig. 8: Updating records after query processing in client

Based on the given query, the table is updated in the client. When the tables are updated in the client, the database in the server is automatically updated which is described in Fig.7, Fig. 8, Fig.9 and Fig. 10.

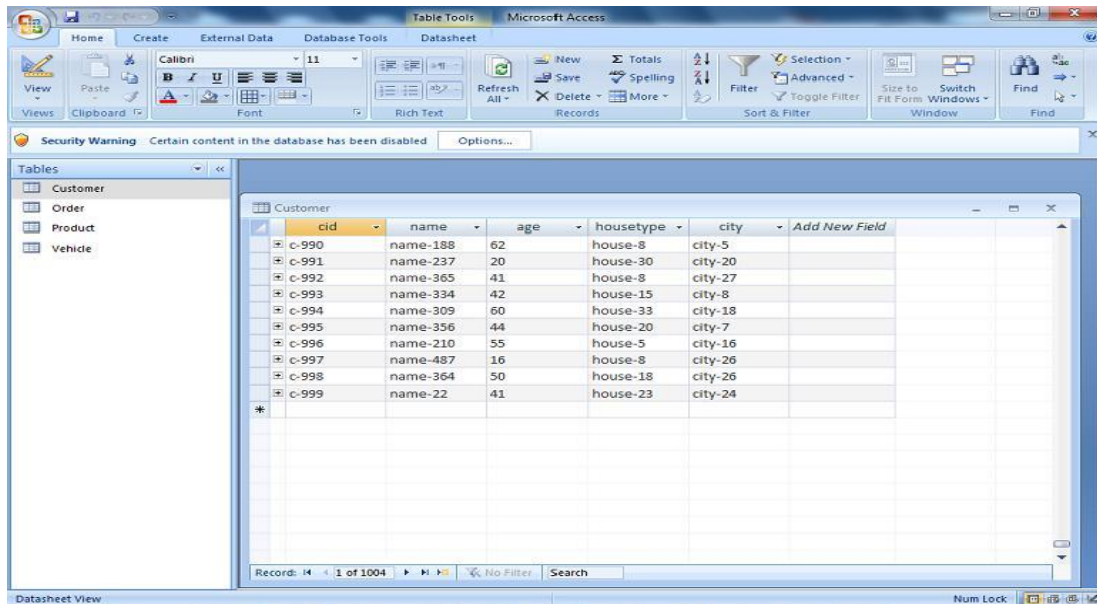


Fig. 9: Table before replication in server

cid	name	age	housetype	city
c-986	name-440	31	house-7	city-2
c-987	name-389	62	house-2	city-4
c-988	name-431	69	house-3	city-5
c-989	name-359	38	house-28	city-3
c-99	name-219	65	house-22	city-14
c-990	name-188	62	house-8	city-5
c-991	name-237	20	house-30	city-20
c-992	name-365	41	house-8	city-27
c-993	name-334	42	house-15	city-8
c-994	name-309	60	house-33	city-18
c-995	name-356	44	house-20	city-7
c-996	name-210	55	house-5	city-16
c-997	name-487	16	house-8	city-26
c-998	name-364	50	house-18	city-26
c-999	name-22	41	house-23	city-24
cid-1004	name-105	20	type105	c=105
cid-1005	name-106	21	type102	c=103
cid-1006	name-106	27	type108	c=107
cid-1007	name-107	27	type109	c=100

Fig. 10: Updation of records in the table after replication in server

The final updated database in the server is shown in Fig. 10. Hence the data replication based on decision supporting system is performed efficiently in our proposed system.

6. CONCLUSION

Replication plays a vital role in distributed systems. Systems failures often occur in replicated data's due to the unwanted scalability and fault tolerance. Decision Support systems (DSS) is used to clear the fault occurs in the database by using a partial replication with analytical model. It is to understand the potential scalability gains of partial replication with respect to full replication. It will give better performance by using a DDS-R schema. It will eradicate the system errors and the distributed systems with database can access with multiple replication without any fault tolerance. Experimental Results clearly shows the data replication for the distributed database using decision support systems will give the better performance by reducing the false errors.

7. REFERENCES

- [1] Reza Akbarinia, Mounir Tlili, Esther Pacitti, Patrick Valduriez and Alexandre A. B. lima, "Replication in DHTs using Dynamic Groups," Lecture Notes in Computer Science, Vol. 6790, No.3, pp. 1-19, 2011.
- [2] Reza Akbarinia, Mounir Tlili, Esther Pacitti, Patrick Valduriez and Alexandre A. B. lima, "Continuous Time stamping for Efficient Replication Management in DHTs," Lecture Notes in Computer Science, Vol. 6265, pp. 38-49, 2010.
- [3] Prasanna Padmanabhan, Le Gruenwald, Anita Vallur and Mohammed Atiquzzaman, "A survey of data replication techniques for mobile ad hoc network databases, International journal on very large database," Vol. 17, pp. 1143-1164, 2008.
- [4] Dutta, H., Kamil, A., Pooleery, M., Sethumadhavan, S., Demme, J.: Distributed Storage of Large-Scale Multidimensional Electroencephalogram Data Using Hadoop and HBase," Grid and Cloud Database Management, pp. 331-347, 2011.
- [5] Hannes Muhleisen, Tilman Walther and Robert Tolksdorf, "Data Location Optimization for a Self-Organized Distributed Storage System," In Proc. of the Third World Congress on Nature and Biologically Inspired Computing (NaBIC), pp. 176 - 182, Oct 2011.
- [6] Yi Lin, Bettina Kemme, Marta Patino Martinez, Ricardo Jimenez Peris, "Consistent Data Replication: Is it feasible in WANs," Information Science and Computer communication, 2004.
- [7] Zahia Guessoum, Jean-Pierre Briot, Nora Faci and Olivier Marin, "Towards reliable multi-agent systems: An adaptive replication mechanism," Journal of Multi agent and Grid Systems, Vol.6, No. 1, pp. 1-24, 2010.
- [8] Byung-Gon Chun, Frank Dabek, Andreas Haeberlen, Emil Sit, Hakim Weatherspoon, M. Frans Kaashoek, John Kubiatowicz and Robert Morris, "Efficient Replica Maintenance for Distributed Storage Systems," In Proc. of the Symposium on Networked Systems Design and Implementation, 2006.
- [9] Michael H. Zack, "The role of decision support systems in an indeterminate world," Decision Support Systems, Vol. 43, No. 4, pp. 1664-1674, 2007.
- [10] Sanjay Kumar Tiwari and A.K.Sharma, "Management Issues in Replicated Distributed Real Time Database," International Journal of Advance in Science and Technology, Vol.3, No1, pp. 75-89, 2011.
- [11] Maris G. Martinsons and Robert M. Davison, "Strategic decision making and support systems: Comparing

- American, Japanese and Chinese management," *Decision Support Systems*, Vol. 43, pp. 284 – 300, 2007.
- [12] G. Bent, D. Vyvyan, David Wood, Petros Zerefos and Seraphin Calo," Distributed Policy Based Access to Networked Heterogeneous ISR Data Sources," In Proc. of the conference on International society of optics and photonics, Vol. 7694, 2010.
- [13] H. T. Barney and G. C. Low," Object Allocation with Replication in Distributed Systems," *World Academy of Science, Engineering and Technology*, Vol. 48, pp. 558-566, 2008.
- [14] Costel Ciuchi, Dan Picu and Gyorgy Todoran," Managing Knowledge and Data for a Better Decision in Public Administration," *Administration and Public Management*, Vol. 2011, No. 17, pp. 64-81, 2011.
- [15] Carlo Curino, Evan Jones, Yang Zhang and Sam Madden," Schism: a Workload-Driven Approach to Database Replication and Partitioning," In Proc of the Conference on very large database Endowment, Vol.3, No.1, Sep 2010.
- [16] Andres Neyem, Sergio F. Ochoa and Jose A. Pino," Integrating Service-Oriented Mobile Units to Support Collaboration in Ad-hoc Scenarios," *Journal of Universal Computer Science*, Vol.14, No. 1, pp. 88-122, 2008.
- [17] Constandinos X. Mavromoustakis and Helen D. Karatza," A Gossip-Based Optimistic Replication for Efficient Delay-Sensitive Streaming Using an Interactive Middleware Support System," *IEEE Systems Journal* , Vol. 4, No. 2, pp. 253- 261, Jun 2010.
- [18] Nikos Hardavellas, Michael Ferdman, Babak Falsafi and Anastasia Ailamaki," Reactive NUCA: Near-Optimal Block Placement and Replication in Distributed Caches," In Proc. of the 36th Annual International Symposium on Computer Architecture, pp. 1- 12, Jun 2009
- [19] Rui Garcia, Rodrigo Rodrigues and Nuno Pregoica," Efficient Middleware for Byzantine Fault Tolerant Database Replication," In Proc. of the sixth conference on Computer systems, 2011.
- [20] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes and Robert E. Gruber," Bigtable :A Distributed Storage System for Structured Data," *ACM Transactions on Computer Systems*, Vol. 26, No. 2, pp. 1-26, 2008.
- [21] Peter A. Boncz, Angela Bonifati, Arantza Illarramendi, Peter Janacik, Birgitta König-Ries, Wolfgang Lehner, Pedro José Marrón, Wolfgang May, Aris M. Ouksel, Kay Römer, Brahmananda Sapkota, Kai-Uwe Sattler, Heinz Schweppe, Rita Steinmetz and Can Türker," Working Group Report on Managing and Integrating Data in P2P Databases," In Proc. of the conference on Scalable Data Management in Evolving Networks, 2007.
- [22] Daniel J. Abadi," Data Management in the Cloud: Limitations and Opportunities," *IEEE Computer Society Technical Committee on Data Engineering*, pp. 1- 10, 2009.
- [23] Iraj Mahdavi, Babak Shirazi and Maghsud Solimanpur," Development of a simulation-based decision support system for controlling stochastic flexible job shop manufacturing systems," *Simulation Modeling Practice and Theory*, Vol. 18, pp. 768–786, 2010.
- [24] Abbe Mowshowitz, Akira Kawaguchi, Andi Toce, Andrew Nagel, Graham Bent, Paul Stone and Patrick Dantressangle," Query Optimization in a Distributed Hypercube Database," *Annual Conference on International Technology Alliance*, 2010.
- [25] Narasimhaiah Gorla," Sub query Allocations in Distributed Databases Using Genetic Algorithms," *Journal of Computer Science and Technology*, Vol.10 No.1, pp.31-37, Apr 2010.
- [26] Jon Olav Hauglid, Norvald H. Ryeng and Kjetil Norvag," Dynamic Fragmentation and Replica Management in Distributed Database Systems," *Journal of Distributed and Parallel Databases*, Vol. 28 No. 3, pp. 1- 25, Dec 2010.
- [27] Rajinder Singh Virk and Dr. Gurvinder Singh," 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, Jun 2011.
- [28] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme and G. Alonso," Understanding Replication in Databases and Distributed Systems," In Proc. of the 20th International Conference on Distributed Computing Systems, pp. 464 - 474, Apr 2000.
- [29] Marius Cristian Mazilu," Database Replication," *Journal of Database Systems*, Vol. I, No. 2, pp. 33- 38, 2010.
- [30] Iuliana Scorta," The Replication Mechanism in a Romanian ERP System Environment," *Journal of information economics*, Vol.1, No. 45, pp. 140- 146, 2008.
- [31] Yair Amir and Ciprian Tutu," From Total Order to Database Replication," In Proc. of the 22nd International Conference on Distributed Computing Systems, pp. 494 - 503, 2002.
- [32] Yi Lin, Bettina Kemme, Marta Patino Martinez and Ricardo Jimenez-Peris," Enhancing Edge Computing with Database Replication," *26th IEEE International Symposium on Reliable Distributed Systems*, pp. 45 - 54, 2007.
- [33] Angela A. Hung and Charles R. Plott," Information Cascades: Replication and an Extension to Majority Rule and Conformity-Rewarding Institution," *American Economic Review*, Vol. 91, No. 5, pp. 1508-1520, 2001.