Increases the Performances of National Social Insurance Fund Information System Using Middleware Base Database Replication

Atif Mohammed Mohammed Taha

B.Sc. in Mathematics and Physics, Faculty of Science, University of Khartoum (1992)

Preliminary Qualifying Degree, Collage of Science, Sudan University of Sciences and Technology(1999)

A Dissertation
Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Sciences and Information

Department of Computer Engineering
Faculty of Engineering and Technology

November, 2016
Increases the Performances of National Social Insurance Fund Information System Using Middleware Base Database Replication

by

Atif Mohammed Mohammed Taha

Supervision Committee:

<table>
<thead>
<tr>
<th>Name</th>
<th>Position</th>
<th>Signature</th>
</tr>
</thead>
</table>
| Dr. Izzeldeen Mohammed Osman | Main supervisor     | ...........
| Dr. Sally Dafa Alla AwadElkriem| Co supervisor       | ...........

Date: November/2016.
Increases the Performances of National Social Insurance Fund Information System Using Middleware Base Database Replication

by

Atif Mohammed Mohammed Taha

Examination Committee:

Name                  Position              Signature
Dr. Izzeldeen Mohammed Osman    Chair Person         .............
Dr. Yasser Abd Elgadir Mohamed Ahmed  External Examiner  .............
Dr. Magdi Baker Mahmoud Amien    Internal Examiner    .............

Date of Examination: 16/ November/2016
DEDICATION

I dedicate this Thesis to my mother, brothers, sisters, wife and sons.
Acknowledgments

I deeply thank all the people who, during this master's study, have contributed in various ways to my work.

First of all, I want to express my gratitude to my principal supervisors, Professor IzzEldeen Mohammed Osman for guiding my first steps into academic research. I would also like to thank my supervisor Ass. Prof. Dr. Sally Dafa Alla for advising me and directing me when I was unsure about my academic decisions. I feel privileged to have studied under her guidance.

I would also like to thank those who contributed, through their comments and suggestions, to various parts of this research also have accepted to proofread this dissertation, thus making it possible to dramatically improve its content: Dr. Fakhr Eldeen Mohammed Suliman, Dr. Tag Elsir Suliman and Mrs. Intisar.

I want to thank all members of IT department in National Social Insurance Fund, past and present, for their support and friendship.

Finally, I want to express my love and gratitude to my family and close friends. Their constant support and encouragements were determinant for the completion of this dissertation. I am especially grateful to my mother, my sisters, my brothers and my wife Samia for their continuous support and belief in me has been a tremendous source of inspiration.
Abstract

Replication is one of the good ways to increase the performance of database system by separating out databases by maintaining different servers. Database replication is used to provide high availability and performance for accessing data. There are different replication strategies in today’s market, and many database replication methods to choose between, and different DBMS systems support different replication strategies. The purpose of this research is to find or devise the most efficient and reliable replication technique for national social insurance fund organization (NSIF) in Sudan, and implement a replication method adapted database middleware base replication methods, as a solution for NSIF. The methods chosen because it is easy to implement, low cost, reliable, can add new node easily, no need for extra software / hardware, no need to change the structure of existing database and it can be used as database immigration in the future. In this research the SymmetricDS Pro application was used as replication middleware layer to resolve NSIF data process delaying and replaces the exchange of data through SMS files by exchanging real data between nodes and head office. This enhances NSIF work and allows the monitoring of each office continuously. The symmetricDS Pro application can identify errors occurring during data transfer and generate error messages for the user. These errors may arise from network connection failure, configuration mistakes, or database setting. The database setting errors have been avoided by creating a database table which contains an ID for each node. This ID is used as part of the primary key in each database. The results obtained from Omdurman, Amarat, and South Khartoum nodes reflect that the objectives of this research have been achieved successfully.
زيادة الأداء من نظام معلومات الصندوق القومي للتأمين الاجتماعي عن طريق البرمجيات الوسيطة إلى قاعدة بيانات النسخ المتماثل

ملخص الدراسة

تناول هذا البحث منهجيات ربط قواعد البيانات العلائقية في المكاتب الطرفية لمؤسسة مركزية بغرض تحسين أداء المؤسسة. غطي البحث طرق منهجيات ربط قواعد البيانات الموزعة وشرح استخدام كل منها. كما حاول إيضاح معايير كل منهجية ونقاط ضعفها. وقد تم استخدام واحدة من هذه المنهجيات المتصلة لربط قواعد البيانات العلائقية الموزعة في مكاتب الصندوق القومي للتأمين الاجتماعي مع رئاسة الصندوق من أجل رفع كفاءة عمل النظام المستخدم وتحسين أداء المؤسسة. تم اختيار منهجية البرمجيات الوسيطة إلى قاعدة بيانات النسخ المتماثل (middleware base replication) كحل للمشكلة لعدة أسباب سهولة تطبيقها، قلة تكلفتها وقوة أداءها، اعتماديتها وإمكانية إضافة مكاتب جديدة جانب المكاتب الحالية بكل بساطة وبقليل التكلفة. كما راعى الحل إمكانية تهجير البيانات وقواعد البيانات الحالية إلى أنظمة بيانات أخرى في المستقبل. بما أن هناك عدة برامج تطبيقات تعمل بمنهجية البرمجيات الوسيطة إلى قاعدة بيانات النسخ المتماثل (Oracle GoldenGate وبرنامج SymmetricDS Pro) فقد تم المقارنة بين تطبيقيات مختلفين وهما برنامج replication وبرنامج
# Table of Contents

DEDICATION . ................................................................. iv  
Acknowledgments. .......................................................... v  
Abstract . ........................................................................ vi  
Abstract in Arabic............................................................... vii  
Table of contents . ............................................................ viii  
List of Figures . ............................................................... xi  

1 Introduction. ................................................................. 1  
1.1 Introduction. ............................................................... 1  
1.2 Problem Definitions ..................................................... 3  
1.3 Solution Strategies ....................................................... 3  
1.4 Research Objectives...................................................... 4  
1.5 Research Methodology................................................... 5  
1.6 The Scope of Thesis....................................................... 5  
1.7 Thesis Outline ............................................................ 5  

2 Database replication ....................................................... 6  
2.1 Database replication definition ....................................... 7  
2.2 Replication Model ......................................................... 8  
2.3 Development of replication techniques ............................. 12  
2.4 Review of the approaches .............................................. 15  
2.4.1 Time of transactions ............................................... 15  
2.4.1.1 Eager replication ................................................. 16  
2.4.1.2 Lazy replication ............................................... 19  
2.4.2 System architecture ............................................... 25  
2.4.2.1 Comparison of replication protocols ...................... 28  
2.4.3 Server interaction .................................................. 29  
2.4.4 Transaction termination .......................................... 30
4.4 Implementation results ............................................................... 54
  4.4.1 Omdurman node results ....................................................... 55
  4.4.2 Amarat node results ............................................................ 56
  4.4.3 South Khartoum node results .............................................. 58
  4.4.4 Error messages results ....................................................... 59
5. Conclusion .................................................................................... 60
  5.1 Conclusion .............................................................................. 60
  5.2 Future work. ........................................................................... 61
List of Figures

2.1 Ways of updates propagation in replicas ........................................ 11
2.2 Updates control in replicas .......................................................... 12
2.3 SRDF based data replication architecture ......................................... 19
2.4 Architecture for Snapshot based data replication ............................... 22
2.5 Cascade replication for PostgreSQL Slony-I ..................................... 25
2.6 Database replication architecture with Centralized (a) and Distributed (b) Middleware ................................................................. 28
Chapter 1

Introduction

1.1 Introduction

Systems used nowadays are more and more geographically dispersed. At the same time requirement to reduce the time of access to data is getting higher and higher. Ensuring the appropriate time of data accessibility is required in every kind of data processing system, and it is especially important in systems implemented for complex, distributed environments processing huge amounts of transactions [3, 54, 61, 84]. Moreover, the growing companies expand their activities to many countries all over the world and in the same time data processing requirements and methods change as well. Additional requirements appear in such systems, which are to ensure an appropriate level of service availability and efficiency in environments consisting of lots of data centers spread among many distant countries. On the other hand these demands are getting more and more difficult to fulfill as the complexity of the system grows (multi-node environments, long distances between remote nodes, lots of messages exchanged).

Data replication technique can be invaluable technique that allows fulfilling high demands of distributed data management systems [3, 15, 54, 61, 84]. The majority of the presented in the literature approaches designed for data replication in complex, multi-node, transactional environments focuses mainly on the fault-tolerance issues within the system [4, 32, 38] and the security of the system [97,116].

It is obvious that data synchronization in distributed, multi-node systems, similarly to many other distributed systems, is performed between a large numbers of remote nodes. This causes that the whole system that uses the replication approach must
ensure high scalability level understood as the possibility of extending of the data management system by adding remote replicas.

Data in the system covering wide areas can be stored in different database management systems running on variety of operating systems and hardware platforms, therefore, it is also necessary to enable appropriate cooperation between replication nodes in such heterogeneous systems [48].

This research takes Social Insurance Fund Organization in Sudan as a case study of spread organization in all around the Country.

Social insurance system is a mandatory legislation, which is issued by the State, in order to protect salaried workers, against specific social risks, through guaranteeing in cash and in kind benefits – which are financed through contributions – for workers and their families[1] [170].

Most countries of the world, whether rich or poor, knew and adopted social insurance systems as a legal modern system.

Sudan, like other countries of the world, knows the social insurance system since the beginning of the seventies of the past century [170]. National Social Insurance Fund (NSIF) starts since 1975. It has its Head office in Khartoum and more than 25 regional offices all round the country. The Head office and the regional offices are connected with WAN.

NSIF have four separate systems:

- Insurance System is a centralized system that all the offices can access.
- Financial and Human Resource System which is an Enterprise Resource Planning (ERP) system that runs on the Head offices LAN.
- Beneficial System work in regional offices LANs.
- Isolated Financial Systems to cover the remainder offices.
1.2 Problem Definitions
Chief Executive Officers (CEOs) face problem when they need information from the regional offices. Difference in Database versions or differences Database vendors’ prevented the CEOs getting reports directly from the regional systems. Also sometimes connection dropdown or interrupt between Head office and regional offices cause the communication to be missed between the sites. The consequences of missing communication affects the data processed and inserted during the communication interruption and it delays the time of finishing the jobs.

![Figure 1.1 NSIF system configurations](image)

1.3 Solution Strategies
Middleware Replication can be used as a solution for making NSIF System available in all different sites even in the communication dropdown. Because replication is a set of technologies for copying and distributing data and database objects from one database to another and then synchronizing between databases to maintain consistency.
Replication allows maintaining the same database multiple copies at different locations. Log, shipping and mirroring allows for maintaining complete database redundancy

1.4 Research Objectives
The aim of this research is to increase performances using the replication method in the existing systems with large number of nodes. The proposed method should also accomplish fault-tolerance requirements and should be applicable for heterogeneous environment based on various software and hardware platforms, similar to National Social Insurance Fund.

The research focuses on the approach most suitable for heterogeneous environments which is implied as the possibility of replication implementation in environments with different RDBMS, operating systems and/or hardware platforms without the necessity of using complex gateways, additional drivers, etc.

In this research a solution is suggested for the NSIF database system to achieve the following objectives:

- To improve performance
- To reduce locking conflicts when multiple users are working
- To improve availability of information systems even when the connection is interrupted or the server is damage.
- Make maintenance easy.
- To allow sites work independently, so that each location can set up its own rules and procedures for working with its copy of the data.
- To move data closer to the user.

To achieve the research objectives the following steps are applied:

- Collecting data of the existing system.
- Evaluation for the performance of the existing systems.
- Implementing the optimum replication method which fits the need.
1.5 **Research Methodology**
The research contributions of this thesis and planned research tasks are:

- Analysis of replication techniques usage spectrum,
- Propose the Middleware Replication Method,
- Evaluation of Middleware Replication Method,
- Precise and intensive description of National Social Insurance Fund Systems,
- Practical implementation of middle-ware replication in National Social Insurance Fund Organization,

1.6 **The Scope of Thesis**
National Social Insurance Fund is one of the biggest organizations in Sudan and it has many information systems (Account system, Insurance system, Retirement system, Human resource system, Health insurance system ... etc. The systems work in separate servers.

The replication method is implemented on the financial system of six offices according the size of data transfer.

1.7 **Thesis Outline**
This thesis contains five chapters:
Chapter 1 is the introduction.
Chapter 2 Literature View
Chapter 3 NASIF Organization and Middleware
Chapter 4 is the Result and Discussion.
Chapter 5 concludes the work done in this research and provides some recommendations for future works.
Chapter 2

Literature Review

2.1 Database replication

As it is stated in the introduction, the majority of the currently used systems is distributed among many remote locations. Supply chain management, customer relationship management, business intelligence and many other types of applications usually do not share data or business rules nor communicate with each other to exchange data. Thus, identical data is stored in multiple locations, and consequently there is no possibility to automate business processes.

A solution to this problem can be the Enterprise Application Integration [27], which is a process of linking applications within a single organization together, to simplify and automate business processes. An access to a huge amount of data distributed among remote sites and operating on them is also realized using data replication techniques. Besides storing copies of the same data in multiple remote locations, database replication improves performance, scalability and fault-tolerance of the systems. In data replication process it is essential to keep copies consistent across the whole system. Several correctness criteria have been proposed for database replication: 1-copy serializability [5, 49], generalized snapshot isolation [1, 22] and 1-copy snapshot isolation [61].

Data replication is one of the most advantageous techniques in dealing with failures and improving efficiency for the variety of data storing systems. It has been an area of researches for almost forty years, as the first publications related to data replication appeared in the late seventies [103, 111]. Years of detailed studies made it possible to develop a wide range of algorithms and protocols which can be
used for maintaining data replication in distributed environments. The turn of the 21st century has witnessed a rapid development of those techniques as various approaches for database replication have been proposed. There has been much development implemented for many replication approaches in several architectural systems. A typical attribute of replication approaches is that they only apply to specific applications like synchronization of certain tables, data transfer between online transaction processing systems (OLTP) and decision support systems (DSS), hot fail over, etc. It is a great challenge to design and implement such an approach that would be suitable for many systems with different characteristics. This issue is extremely complex and still requires a lot of research to be done. Moreover, it is possible that a universal replication approach can never be found since various requirements of the future systems may necessitate the use of solutions especially designed to meet new demand.

In the last few years a lot of replication techniques for database systems have been proposed in the literature and many practical implementations have appeared. A comparison of these conceptually similar techniques is not easy to deal with because of many subtle differences in the mechanisms used. Furthermore, it is extremely hard to relate results coming from different areas of their usage. All these factors make replication techniques very difficult to compare with each other. Moreover, even though a lot of research has been completed and many replication techniques have been studied, it is still highly probable that there are many other possibilities of database replication that have not been explored nor yet proposed.

2.2 Database replication definition

Database replication is the creation and maintenance of multiple copies of the same database [108]. It is a technique that allows ensuring a coherence and forces data
synchronization in the distributed database system. Database copies are called replicas and may be located in many remote sites.

The database replication should be transparent for users, in a way as if they were working on a single instance database. In a system with centralized database every client connects to the same server. In a system in which database is replicated among various sites, client may choose a replica to connect, or, which is more often used in practice, client connects to a dedicated replica, which usually is the nearest one to that client in terms of network distance.

If replicas run on separate servers, the database replication system has two important advantages comparing to the centralized database:

- **High availability** - if one replica crashes due to a software or hardware failure, the remaining replicas can still continue processing, while centralized database system becomes completely unavailable after only one crash.

- **Increased performance** - the transaction processing load can be distributed among all the replicas in the system. This leads to a larger throughput (since queries and read operations do not change the database state, they can be independently executed in one replica only) and a shorter response times for queries (because queries can be executed only in one replica, which is usually in the same location as the client, and without any additional communication among the replicas).

### 2.3 Replication Model

When replicated, a simple single-node transaction may apply its updates remotely either as part of the same transaction (**eager**) or as separate transactions (**lazy**).

In either case, if data is replicated at N nodes, the transaction does N times as much work [30].

Figure 2.1 shows two ways to propagate updates to replicas (eager and lazy).
Eager updates are applied to all replicas of an object as part of the original transaction. One replica is updated by the originating transaction and updates to other replicas propagate asynchronously, typically as a separate transaction for each node when lazy update is performed.

Figure 2.1: Ways of updates propagation in replicas

Updates may be controlled in two ways which is presented in fig. 2.2. Either all updates are initiated by a master copy of the object, or updates may be initiated by any. Group ownership has many more chances for Conflicting updates.

The considered model assumes that the database consists of a fixed set of objects. There are a fixed number of nodes, each storing a copy of all replicated objects. Each node originates a fixed number of transactions per second. Each
Figure 2.2: Object master and object group

Figure 2.2 Updates control in replicas transaction updates a fixed number of objects. Inserts and deletes are modeled as updates, reads are ignored.

Replica update requests have a transmit delay and also require processing by the sender and receiver. These delays and extra processing are ignored; only the work of sequentially updating the replicas at each node is modeled. Some nodes are mobile and disconnected most of the time. When first connected, a mobile node sends and receives deferred replica updates. Parameters used in replication model are listed in table 2.1.

Table 2.1: Variables used in the model and analysis

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB_Size</td>
<td>Number of distinct objects in the database</td>
</tr>
<tr>
<td>Nodes</td>
<td>Number of nodes; each node replicates all objects</td>
</tr>
<tr>
<td>Transactions</td>
<td>Number of concurrent transactions at a node. This is a derived value</td>
</tr>
<tr>
<td>TPS</td>
<td>Number of transactions per second originating at this node</td>
</tr>
<tr>
<td>Actions</td>
<td>Number of updates in a transaction</td>
</tr>
<tr>
<td>Action_Time</td>
<td>Time to perform an action</td>
</tr>
</tbody>
</table>

Each node generates TPS transactions per second. Each transaction involves a fixed number of actions and each action requires a fixed time to execute. Thus duration of transaction equals to Actions * Action_Time. Given these two observations, the number of concurrent transactions originating at a node is:

\[
\text{Transactions} = \text{TPS} \times \text{Actions} \times \text{Actions}\_\text{Time}
\]

(2.1)

In a system of N nodes, N times as many transactions will be originating per second.

Since each update transaction must replicate its updates to the other (N-1) nodes, it
is easy to see that the transaction size for eager systems grows by a factor of $N$ and the node update rate grows by $N^2$. In lazy systems, each user update transaction generates $N-1$ lazy replica updates, so there are $N$ times as many concurrent transactions, and the node update rate is $N^2$ higher. This nonlinear growth in node update rates leads to unstable behavior as the system is scaled up [30].

Eager Replication updates all replicas when a transaction updates any instance of the object. There are no serialization anomalies (inconsistencies) and no need for reconciliation in eager systems. Locking detects potential anomalies and converts them to waits or deadlocks [30].

In a single-node system with the eager replication transactions have about

$$\frac{\text{Transactions} \times \text{Actions}}{2}$$

resources locked (each is about half way complete).

Since objects are chosen uniformly from the database, the chance that a request by one transaction will request a resource locked by any other transaction is

$$\frac{\text{Transactions} \times \text{Actions}}{2 \times \text{DB Size}}$$

. A transaction makes Actions such requests, so the chance that it will wait sometime in its lifetime is approximately [29, 31]:

$$PW = 1 - \left(1 - \frac{\text{Transactions} \times \text{Actions}}{2 \times \text{DB Size}}\right) \times \text{Actions} \quad (2.2)$$

$$PW = \frac{\text{Transactions} \times \text{Action}^2}{2 \times \text{DB Size}} \quad (2.3)$$

A deadlock consists of a cycle of transactions waiting for one another. The probability a transaction forms a cycle of length two is $PW^2$ divided by the number of transactions. Cycles of length $j$ are proportional to $PW^j$ and so are even less likely if $PW <<< 1$.

The probability that the transaction deadlocks is approximately:
Lazy group replication allows any node to update any local data. When the transaction commits, a transaction is sent to every other node to apply the transaction updates to the replicas at the destination node. It is possible for two nodes to update the same object and race each other to install their updates at other nodes. The replication mechanism must detect this and reconcile the two transactions so that their updates are not lost.

Transactions that would wait in an eager replication system face reconciliation in a lazy-group replication system. Waits are much more frequent than deadlocks because it takes two waits to make a deadlock [30].

Eager replication waits cause delays while deadlocks create application faults. With lazy replication, the much more frequent waits are what determine the reconciliation frequency. So, the system-wide lazy-group reconciliation rate follows the transaction wait rate equation:

\[
PD = \frac{PW^2}{Transactions} = \frac{\text{Transactions} \times \text{Action}^6}{4 \times \text{DB Size}^2} \tag{2.4}
\]

\[
PD = \frac{\text{TPS} \times \text{Actions Time} \times \text{Actions}^5}{4 \times \text{DB Size}^2} \tag{2.5}
\]

2.3 Development of replication techniques

At the beginning of its development data replication for databases was realized by the modifications in a source code of the database management system engine. These changes were performed in various parts of the engine, like in the transactional log module which consists of all the modifications in database, or with the usage of additional modules ensuring group communication. An example
of the system based on this idea is the implementation of the system Postgres-R [53, 54], which is characterized by relatively good performance as the overhead related to replication process is low. However, since it is necessary to modify the source code of the essential part of database, which is its engine, systems implemented on the basis of code modifications in the database engine are very hard to realize in different system platforms. Difficulties are present not only in case of the usage of different system engines or operating systems, but also appear for different versions of the database supplied by the same vendor, even if it is the next release of this database.

The solution presented in [5] is based on the propagation of all local operations to each remote site in the system. Unfortunately, propagating of every operation to the remaining sites in the system leads to the frequent occurrences of the distributed deadlock for these operations. Thus, after some research a new ROWAA approach (Read One Write All Available) ensuring the coherency of replication process has been designed. In this approach all operations related to a single transaction are first processed in one site only, and then data modifications are transferred to the remaining nodes, without the necessity of any additional communication messages. Since fewer messages are transferred through the network to process a transaction, it is obvious that the transaction time is shorter, and the replication performance is improved.

Optimistic 2 Phase-Locking protocol (O2PL) presented in [9] is the example of this approach. It is one of the first approaches ensuring replicated data coherence on the basis of ROWAA approach. This approach is realized with the usage of an adaptation of the two phase-locking protocol BPL) in which local transactions are distinguished from remote transactions. It allows predicting and avoiding or decreasing the quantity of deadlocks. Start of the transaction processing immediately after it is delivered to the site significantly decreases the overall time
required to commit this transaction. After a complete transaction is delivered, its correct order is determined, and if the transaction is not in Conflict with preceding transactions it is committed. Otherwise, the whole transaction is processed from the beginning once more. Some approaches based on group communication were designed and implemented to decrease deadlocks influence on data replication.

Group Communication Systems (GCS) approaches used in [11, 43] provide a mechanism that guarantees necessary order of delivered messages in the network, and also enable failure detection in any site of the system. The most restrictive order of delivered messages requires the same order of delivering in every node of the system, which allows avoiding distributed deadlocks. Group communication based approaches were widely explored and also some systems using it were implemented, for instance Basic Replication Protocol (BRP) presented in [43] or Postgres-R project in [53, 54]. Unfortunately, group communication based approaches are not the most effective technique in preventing deadlocks for all possible applications or every type of transaction. It is so, because exchanging of additional messages between sites is necessary to ensure the proper order of delivered messages related to transactions. An overhead related to it can significantly decrease the performance of the approach [18]. The following researches in GCS techniques led to reduction of the influence of latencies in the network on the overall time required to deliver transaction to every node in the correct order. An example of such solution is Generic Broadcast approach discussed in [2, 78]. In this approach delivery order is important only for these transactions which are Conflicting, while the rest of transactions can be delivered in any order. Another example is Optimistic Atomic Broadcast approach presented in [55, 115]. Messages in this approach are delivered in the same order as they
were received, which enables quick application of write set in the remote nodes, despite of the necessity of waiting for the final order of transactions before they are committed. Thus, only those remote transactions whose write set did not follow the total order is rolled back, reapplying them in the correct order. Group communication based techniques also have been used for the epidemic algorithms presented in [37].

2.4 Review of the approaches

Very well known and useful classification was proposed by Gray in [30], where replication approaches are grouped according to two parameters. These parameters are: the place in which a transaction is initiated and the time when a transaction is distributed to each node. Wiesmann in [117, 118] proposes an extended classification using three parameters- first parameter is the architecture of the server, the second one is the degree of communication among database nodes during the execution of a transaction, and the last one is the transaction termination protocol.

The classification of replication techniques is included in the following subsections. Replication techniques are distinguished on the basis of the architecture of the replication system, the time of data propagation between replicas, interactions between nodes, the way of terminating transactions and the possibility of their usage in heterogeneous environments.

2.4.1 Time of transactions (when the updates are propagated)

In replicated system consistency of data in all replicas, in the presence of updates, has to be ensured. Some replication protocols require strong consistency, which means that data must be consistent all the time. In such cases replicas must coordinate updates before the response is sent to the client and response time increases. There are protocols that require only weak consistency allowing data to
be inconsistent temporarily. This lets us to get faster response for writes, but it is possible that transactions read data which is not up to date since it has not yet been applied in every remote replica. Weak consistency protocols may even require to rollback updates previously applied and committed, even when a client receives a confirmation that commit was successful. Depending on the time when modification in one site is propagated to other replication sites two replication approaches are distinguished: eager and lazy. In eager approach a transaction may be committed only when it is possible to commit it in all sites. Unlike eager approach, lazy approach allows updates to be committed before they are propagated to other sites.

2.4.1.1 Eager replication

Eager replication (synchronous) approach requires immediate propagation of changes from the database node where transaction is submitted to all nodes in the replication system. Thus, eager replication keeps all database replicas totally synchronized in all nodes by updating all the replicas as a part of one atomic transaction.

Eager replication ensures serializable execution of transactions which results in the absence of concurrency anomalies. However, eager replication reduces update performance and increases transaction response times because extra updates and messages are added to the transaction [30].

Eager replication implementations most often are based on Two-Phase Commit protocol BPC) [5] or some modifications of this protocol - Three-Phase Commit protocol CPC) [98], etc. Two-Phase Commit protocol is a distributed algorithm that needs all nodes in a distributed system to agree to commit a transaction. The protocol results in either all nodes committing the transaction or aborting otherwise, even in the case of network failures or node failures.

The greatest drawback of the two-phase commit protocol is the fact that it is a
blocking protocol as it usually uses Two Phase Locking protocol (BPL). In 2PL protocol, a node blocks data (raws, table) while it waits for a message, which means that other processes competing for resource locks held by the blocked processes will have to wait for the locks to be released.

Ensuring 1-copy serializability guarantees that data is coherent and integral and this is the most important advantage of eager replication. Data is up to date in each node and all the time. Disadvantages of this technique are reduced scalability and low efficiency. The consequence of locks may cause deadlocks, locks significantly decrease performance.

Eager replication protocols ensure strong consistency and fault-tolerance as updates must be confirmed at remote replicas before replying to clients. However, this has a meaningful impact in user-visible performance. Also in contrast with lazy protocols, approaches to eager replication provide very different tradeoffs between performance and flexibility [46].

The following part of this subsection provides popular commercial implementation of eager replication approach.

a) Volume Replication

Replication of disk volumes performed at the block I/O level is a straightforward replication approach for general purposes. By intercepting each block written by the application designated volumes and shipping it over to network, a remote copy is maintained ready for fail-over. Reads are performed on the local copy. The replication process is thus completely transparent to the application.

The downside of the approach is that remote updates are performed with a block granularity, which, depending on the application, might represent a large network overhead.

The approach is also restricted to fail-over as the backup copy cannot usually be used even for read-only access, due to lack of cache synchronization at the
application level.

Figure 2.3: SRDF based data replication architecture

Examples of volume replication solutions can be found in Veritas Volume Replicator [106], EMC Symmetrix Remote Data Facility [23] available for many different operating systems, and in the DRBD open source for Linux OS [33].

b) RAIDb protocols

A Redundant Array of Inexpensive Databases (RAIDb) provides a better performance and fault tolerance than a single instance database. It is achieved at a lower cost by combining multiple database instances into an array of databases. RAID to disks are similar to different RAIDb levels; they provide various cost/performance/fault-tolerance tradeoffs. RAIDb-0 features full partitioning, RAIDb-1 offers full replication and RAIDb-2 introduces an intermediate solution called partial replication, in which the user can define the degree of replication of each database table [46].

In RAIDb the distribution complexity is hidden from the clients and therefore they are provided with the view of a single database like in a centralized architecture.
As for RAID, a controller is located in front of the underlying resources and the clients send their requests directly to the RAIDb controller, which distributes them among the set of RDBMS. RAIDb is implemented as a software solution in C-JDBC [107].

2.4.1.2 Lazy replication

In lazy replication (asynchronous) the updates of a transaction are propagated once it has already committed. Lazy approach does not require continuous connection between all nodes. Every node works independently and reads and writes are processed locally.

Updates are propagated to remote nodes either as SQL statements or as log records containing the results of executed operations.

Despite many disadvantages, lazy approach has been used in several commercial DBMSs and it is the best choice when mobile or frequently disconnecting databases are considered. Lazy replication is used to synchronize data for certain tables only, not for the whole database instance. Synchronization can be executed periodically, on the user's demand or just after successful modification in local node.

Processing transactions locally allows transaction quick completion, but does not always ensure replicas consistency and may lead to Conflicts, data incoherency and high abortion rates. Advantages of lazy replication approach include: scalability, no overhead related to Two Phase Locking, resistance to single node failures and constant connection among all nodes not required.

Lazy replication has become a standard feature of many commercial database management systems and open-source database projects [42, 63, 67, 99, 105]. In the replication system with lazy approach implementation every transaction is executed and committed at a single replica without synchronization with other replicas. Every other replica is synchronized later as changes are captured,
distributed and applied. Because of this, the overall performance of the whole system measured as a time of transaction processing increases significantly. However, since updates can be lost after the failure of any replica, lazy replication approaches are not suitable for fault-tolerance solutions with strong data consistency required [46].

There is a variety of implementations in which lazy replication operates in different ways. The main differences between implementations of lazy approach exist in the following areas:

- The way of the whole system is managed,
- System architecture determining which replica processes and publishes updates to other replicas,
- The way in which updates are captured, distributed and applied,
- Filtering of the processed transactions.

Lazy replication protocols can be divided to three phases: first Capture - updates performed on replicated objects transformed into a format suitable for publication, second one Distribution - changes in published objects are propagated to the relevant replicas and last one Apply - updates applied in the relevant replicas.

The following part of this subsection contains a general description of the most popular commercial database management systems with implementation of lazy replication approach.

a) **Oracle**

Oracle database in version 11g and 10g offers three basic replication solutions [8, 67, 68]:

- Snapshot replication,
- Streams replication,
- Advanced replication.
**Oracle snapshot replication** also called materialized view replication is based on the capturing of changes for the materialized views, which are then propagated and applied in the remote sites. A snapshot is a query that has its data materialized, or populated in a form of a data table. When a snapshot is created a table corresponding to the column list in the query is created. When the snapshot is refreshed, the underlying table is populated with the results of the query. For replication, as data changes to a table in the master database, the snapshot refreshes as scheduled, and that data are distributed to the replicated databases.

![Figure 2.4: Architecture for Snapshot based data replication](image)

**Oracle Streams Replication** enables the propagation and management of data, transactions and events in a data stream either within a database, or from one database to another. Streams replication passes published updates, events or even messages to subscribed destinations, where they are applied or further processed. The Oracle Streams technology uses the operation logs (redo logs) as an input for a
capture process. The capture process formats both operations on data (DML) and data definitions (DDL) into the events which are enqueued for distribution process. The distribution process reads the input queue and enqueues the events in a remote database. The queue from which the events are propagated is called the source queue, while the queue receiving the events is called the destination queue. There can be a one-to-many, many-to-one, or many-to-many relationship between source and destination queues. Depending on the type of events can apply process at the destination site applies them directly to the database, or may dequeue these events and send them to an apply handler which performs customized processing of the event and then applies it to the replicated object.

**Oracle Advanced Replication** is realized as a set of support triggers and procedures for each replicated object. These triggers and procedures enqueues a specific remote procedure calls according to the command executed. This queue is consumed by the Oracle implementation of the distribution process which pushes and pulls the deferred calls, propagating the changes from sources to destination databases. Then, the apply process dequeues these information and updates the subscriber. Since Oracle Advanced Replication allows performing updates in every replica, a Conflict detection mechanism is used in every site. When a Conflict is detected, actions defined for the specific types of Conflict are performed (Conflict resolution procedure).

b) **MS SQL Server**

MS SQL Server 2008/2012 provides three replication solutions: [63, 101, 104]

- Transactional Replication
- Snapshot Replication
- Merge Replication
**MS SQL Transactional Replication:** It uses the log as a source to capture incremental changes that were made to a published object. The capture process copies transactions marked for replication from the log to the distribution agent. Basically, the capture process reads the transaction log and queues the committed transactions marked for replication. Then, the distribution agent reads the queued transactions and applies them to the subscribers.

**Snapshot Replication:** Here the entire published object is captured thus it distributes data exactly as it appears at the specific moment of time. This solution does not monitor the updates made against an object. Roughly, the capture process is implemented by a snapshot agent. It periodically copies the replicated object schema and data from the publisher to a snapshot folder for future use by a distribution agent, which also acts as an apply process. Snapshot replication can be used by itself but the snapshot process, which creates a copy of all of the objects and data specified by a publication, is very often used to provide the initial state of data and database objects for the transactional and merge replication.

**MS SQL Merge Replication** allows the publisher and the subscribers to make updates while they are connected or disconnected. When both are connected, it merges the updates. The capture process tracks the changes at the publisher and at the subscribers, while the distribution process distributes changes and also acts as an apply process.

Merge replication allows various sites to work autonomously and later merge updates into a single, uniform result. Because updates are made at more than one node, the same data may have been updated by the publisher and by more than one subscriber. Therefore, Conflicts can occur when updates are merged and merge
replication provides a number of ways to handle Conflicts.

c) **IBM DB2**

IBM DB2 Universal Database 8.2 provides two different solutions that can be used to replicate data from and to relational databases: [41, 42].

- SQL replication
- Q replication

**IBM DB2 SQL Replication** changes are captured at sources and staging tables are used to store committed transactional data. The changes are then read from the staging tables and replicated to corresponding target tables. With staging tables, data can be captured and staged once for delivery to multiple targets, in different formats, and at different delivery intervals.

**IBM DB2 Q Replication** is the replication solution that can replicate large volumes of data at very low levels of latency. Q replication captures changes to the source tables and converts committed transactional data to messages. This data is sent as soon as it is committed to the source and read by Q replication. The data is not staged in tables. The messages are sent to the target location through the message queues. These messages are then read from the queues at destination sites, converted back into the transactional data and applied to the target tables.

IBM DB2 UDB version 8.2 also provides a solution called event publishing for converting committed source changes into messages in an XML format and publishing those messages to applications such as message brokers.

d) **PostgreSQL**

**Slony-I** is a replication solution for the PostgreSQL database, which implements lazy replication as a "master to multiple slaves" replication [99].

In Slony-I, the capture process is implemented using triggers which log the changes made against the published objects. These changes are then periodically distributed using a replication daemon which connects directly to the publisher,
reads the logged changes and forwards them to the subscribers.

Figure 2.5: Cascade replication for PostgreSQL Slony-I

Slony-I allows connecting several subscribers in cascade. The management process is not integrated or centralized, thus the maintenance tasks must be done in each replica separately.

e) **MySQL**

MySQL includes a built-in lazy replication protocol that can be configured to propagate updates between replicas [64]. Replication enables data from one MySQL database server (called the master) to be replicated to one or more MySQL database servers (slaves). Replication is asynchronous and replication slaves do not need to be connected permanently to receive updates from the master. Replication in MySQL features support for one-way, asynchronous replication, in which one server acts as the master, while one or more other servers act as slaves.

**2.4.2 System architecture (where the updates can take place)**

The first parameter to consider in database replication classification is the place where transactions start to execute. Depending on transactions, location in primary approach updates can be executed only in primary (master) site, whilst update
everywhere approach allows update execution at any site (usually updates at the user's local site). According to propagation time updates done at any site have to be propagated to other sites.

**Primary copy**\(^1\) (**Centralized**) approaches are designed for Master-Slave architecture and require to have a specific node (the primary copy) associated with all data in system. Each update of datum, before being sent to all nodes, has to be sent to the primary node, where it is processed - executed or analyzed to determine the order of execution. After transactions processing the primary copy propagates update or its results to the remaining nodes.

Replication mechanism in primary copy approach does not require distributed protocols like Two-Phase Commit (BPC) or Two Phase Locking (BPL), which eliminates overhead related to these protocols. There are also eliminated Conflicts in transactions in primary copy approach as transactions are processed only in primary node and Conflict resolution is realized in a single instance database. On the other hand, this technique has many drawbacks which prevent it from being used widely. These disadvantages are: single point of failure connected with one primary node, bottlenecks in primary node and read-only access to data in nodes that are not primary copy. Since primary copy approach enforces a bottleneck in the replication system and a single point of failure, some modifications for the approach are used to overcome these limitations. This approach easily overloads the primary node, therefore read-only transactions can be applied in secondary nodes which balances the load. Bottlenecks can also be avoided by data partitioning among more than one node. Having different primary nodes for subsets of data, update transactions can be executed in parallel, dividing the load among these nodes. Furthermore, in case of crash of the primary copy, one of the

---

1 Centralized techniques are referred to, in the literature, as primary copy or single master, while distributed ones are referred to as multi-master or update anywhere. These terms, in particular “single master” is confusing, since it refers to one alternative architecture for implementing centralized protocols. Thus, we prefer the more descriptive terms “centralized” and “distributed”.
other nodes can become primary one. Therefore, primary copy approach is mainly used for fault tolerance. In case of one node crash the database is still available for users. This technique is applicable for data transfer between Online Transactional Processing systems (OLTP) and Decision Support Systems (DSS).

**Update everywhere**\(^2\) (Distributed) approach is related to Multiple Master architecture.

This database replication technique does not impose limitations on a node where updating transactions can be processed, and updates can be performed in each node in the system. In update everywhere approach updates can reach two different nodes at the same time, even though they are Conflicting, which is opposite to primary copy approach. In update everywhere replication, nodes are equal to each other. Each of the client's transaction is processed by one node and then changes are propagated to other replicas. However, updating data in every node causes appearance of Conflicts between transactions. Conflict resolution approaches are used to deal with Conflicts.

Update everywhere approaches is suitable for dealing with failures, as election protocols are not necessary to continue processing. Another update everywhere approach advantage related to the symmetric architecture of this replication system is the possibility of reading and writing data in every replica.

When update everywhere technique is used in connection with eager approach it usually requires usage of Two-Phase Commit or Two Phase Locking protocols, which significantly decreases the overall performance of the system. Replication Conflicts appearing when approach is used in connection with lazy replication are also serious problems, as they lower replication performance and need special treatment. The complexity of concurrency control in update everywhere approach

---

\(^2\) Centralized techniques are referred to, in the literature, as primary copy or single master, while distributed ones are referred to as multi-master or update anywhere. These terms, in particular “single master” is confusing, since it refers to one alternative architecture for implementing centralized protocols. Thus, we prefer the more descriptive terms “centralized” and “distributed”.
requires additional maintenance, and usually is realized in middleware subsystem, which coordinates global transactions.

2.4.2.1 Comparison of replication protocols

Table 2.2 Comparisons of update propagation techniques

<table>
<thead>
<tr>
<th>Consistency</th>
<th>Updating</th>
<th>Performance</th>
<th>Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eager update</td>
<td>Strong consistency</td>
<td>Up-to-date with High response time</td>
<td>Not transactional inconsistency, Changes are atomic</td>
</tr>
<tr>
<td>Lazy update</td>
<td>Weak consistency</td>
<td>Out-of-date problem and Low response time</td>
<td>Not fault tolerant, good response time</td>
</tr>
<tr>
<td>Centralized</td>
<td>techniques Up-to-date with Update without synchronization</td>
<td>Appropriate for few master sites</td>
<td>High overload and bottleneck problems</td>
</tr>
<tr>
<td>Distributed techniques</td>
<td>Up-to-date with Concurrency control methods</td>
<td>Highest system availability</td>
<td>Management problem, Copies need to be synchronized</td>
</tr>
</tbody>
</table>

Table 2.3 Update propagation vs. propagation techniques

<table>
<thead>
<tr>
<th>Centralized (Primary Copy)</th>
<th>Distributed (Update Everywhere)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eager</td>
<td>Eager Update Everywhere</td>
</tr>
<tr>
<td>Eager Primary Copy</td>
<td></td>
</tr>
<tr>
<td>One Master by Restricted Transparency</td>
<td></td>
</tr>
<tr>
<td>One Master with Full Transparency</td>
<td></td>
</tr>
<tr>
<td>Lazy</td>
<td>Lazy Update Everywhere</td>
</tr>
<tr>
<td>Lazy Primary Copy</td>
<td></td>
</tr>
<tr>
<td>Single Master with Limited Transparency</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.4 Comparison of replication protocols

<table>
<thead>
<tr>
<th>Replication strategies</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eager Centralized</td>
<td>The coordination do not needs for Update transactions, there is no inconsistencies</td>
<td>Extensive response time, Local copies are can only be Read, Only useful with few updates</td>
</tr>
<tr>
<td>Lazy Centralized</td>
<td>The coordination do not needs for Update transactions, there is diminutive response times</td>
<td>Inconsistencies, Local copies are not refresh</td>
</tr>
</tbody>
</table>
### 2.4.3 Server interaction

The following section provides more information on the classification based on server interaction, parameter related to the number of messages which are sent among database nodes to ensure the applying of transactions. This interaction impacts amount of network traffic, and in consequence is a significant overhead to the processing of transactions. Since applications are more and more geographically dispersed and distances among nodes are getting extremely large, network communication becomes an important factor in replication process.

Server interaction is explained as a function of the number of messages necessary to handle the operations of a transaction [118]. The number of network interactions influences replication protocol significantly and determines the order in which transactions are to be processed.

On the basis of interaction among nodes two cases are distinguished: constant interaction and linear interactions.

**In constant interaction** the number of messages that are used to synchronize database nodes is constant and does not depend on the number of particular operations in the transaction. Typically, these approaches merge all operations related to the transaction in a single message and use only one message per transaction. Replication techniques might also use a greater than one message, but for constant interaction approach the number of messages is always unchangeable, independently of the complexity of the transaction.

On the contrary, **linear interaction** enables technique in which processing of transaction requires exchanging variable amount of messages. The amount of messages depends on the number of operations in a particular transaction as it is
usually proportional to the amount of single operations which are parts of the entire transaction. These single operations might be simply SQL statements, changes collected in the write sets or records in database logs which contain the results of transaction executed in specific node.

2.4.4 Transaction termination

The way of transaction termination decides how atomicity is guaranteed. It leads to the existence of two replication techniques. One of them needs exchange of messages to fulfill requirements of the ACID, the other allow satisfying ACID properties without any additional transmission of messages [117].

An approach based on voting termination requires additional messages to terminate transaction for establishing a coherent state of the replicas. In voting protocols the decision to abort transaction can be made by primary node (weak voting) or by any replica (strong voting).

Voting messages might be very straightforward as a single message sent by one replica to confirm its state. On the basis of these messages the other replicas determine their states. Message exchange can also be much more complex as in the case of protocols like two-phase commitment protocol.

Non-voting termination techniques allow replicas in all nodes to decide on their own whether they commit the transaction or abort it. However, transactions have to be executed in the same order in each replica. It is only a semblance that this is extremely strict requirement to ensure the same order of transactions processing, because for non-Conflicting transactions identical order is not necessary.

2.4.5 Environment complexity

The architectures of the replication environment might be quite simple as in systems with all replicas configured in exactly the same way. These architectures can also be extremely complex in connection with many layers used in such systems, like client, middleware and database tiers. Requirement of cooperation
and realization of replication among different database engines used in one replication system is a great challenge. Depending on an approach possibility to be adapted for replicating data among miscellaneous types of databases, we can distinguish two approaches: heterogeneous and non-heterogeneous replication [48].

Non-heterogeneous approaches are the approaches working in environments consisted of each replica implemented upon the same type of database. This kind of replication is much easier to implement than replication for heterogeneous databases. However, the requirement of replication among different types of databases is getting stronger and stronger and heterogeneous replication is an area for future research.

Heterogeneous approaches can be used in systems with a number of varied databases, which means all these databases might be supplied by different software vendors. Especially, it might happen that these databases have different APIs. Thus, realization of the replication for heterogeneous databases is very complex and a demanding task.

There are some interfaces for accessing data in a heterogeneous environment, like Open DataBase Connectivity driver (ODBC) and Java DataBase Connectivity (JDBC). They can be used for accessing data among replicas, however, it adds another layer, in already complicated environment, and might adversely affect replication performance. Object-relational mapping libraries represented by Enterprise Objects Framework, Hibernate, Enterprise Java Beans and so forth, which convert data between incompatible databases and object-oriented programming languages are also possible to use for replication, but overhead connected with objects mapping might be too high reducing replication effectiveness in practice.

### 2.4.6 Middleware based replication
It is possible to find many algorithms and protocols which are used for maintaining data replication in distributed environments [3, 43, 61, 76]. To ensure data consistency, concurrency control in database management systems are based either on internal mechanisms of database management system, or use a middleware tier [7, 50]. Concurrency control based on middleware protocols is getting more and more popular because of its flexibility and simplicity.

Unlike systems with modified code of the database engine, replication systems built in middleware architecture make possible realization of replication that is transparent for the users and applications. If middleware based replication is implemented it is not necessary to do any modifications in the code of the database engine.

Figure 2.6: Database replication architecture with centralized (a) and distributed (b) middleware

In recent years there appeared many middleware based replication protocols for databases. The middleware protocols maintain replication control in middleware subsystem placed between client and databases. Middleware based replication can be developed and maintained irrespective of database management systems, and
there is a possibility to use it in heterogeneous environments. Distributed middleware architecture has many advantages in regard to replication control, which gives this architecture the advantage of being chosen for further research. The most important features of the middleware based replication are:

- **Increased fault tolerance** of the system - in case of failure of one of the middleware components, it is easily replaced.
- **Scalability** as new databases and middleware servers can be easily added In the location it is required.
- Possibility of **online upgrades** and maintenance.
- **Increased performance** of the middleware.

2.6 **Review of replication techniques usage**

Database management systems used in present day systems can be classified according to the way of data management as operational and analytical databases. Operational databases are used to perform everyday duties in many organizations, institutions and companies. They are applied not only in the systems which require together and store data, but in the systems which need to modify data as well. Operational database stores dynamically changed data which reflects actual state of reality.

On the other hand, analytical databases are used to store historical and archival data, or information related to some events. When a company wants to analyze market tendencies, to gain access to long-term statistic data, or to perform business forecasts, then uses data stored in analytical databases. Data maintained by analytical databases are hardly ever modified, if modified at all. Moreover, this data always presents state of objects in some established moment in the past. Furthermore, appearing new technologies and data formats related to designed database management systems, more and more frequently require storing diverse data types like sounds, graphics, animations and videos, while storing traditional
Existing applications are continuously developed using modern, advanced relations between different types of data, which makes it easier to search demanded data or objects.

Table 2.5 Aspect of data replication usage

<table>
<thead>
<tr>
<th>Aspect</th>
<th>System example</th>
<th>Replication Approaches</th>
<th>Practical implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactional Data</td>
<td>OLTP Systems</td>
<td>Eager replication approaches, MADIS middleware replication, Optimistic replication, Middleware based replication with Snapshot Isolation, Ganymed for web replication, Replication approaches using Group Communication Protocol, Total order multicast, Epidemic commit protocols, Voting Copy approaches</td>
<td>Oracle Advanced Replication, Oracle Streams, TimesTen In-Memory DB Replication, IBM DB2 Q Replication, IBM DB2 Master Replication, MS SQL Server Transactional Replication, MS SQL Server Merge Replication</td>
</tr>
<tr>
<td>Multi-Agent Systems, OLTP in Distributed Multi-Node Systems, Mobile Transactional Systems</td>
<td>Multi-Agent Systems, OLTP in Distributed Multi-Node Systems, Mobile Transactional Systems</td>
<td>Approaches ensuring fault tolerance and security</td>
<td>-</td>
</tr>
<tr>
<td>Large amounts of data</td>
<td>OLAP Systems, Reporting &amp; Analytical Systems, Data warehouses</td>
<td>Lazy replication approaches, RepDB* partial replication, Lazy database replication with ordering guarantees, Middle-R Consistent database replication, Replication approaches using Group Communication Protocol, Partial Write Operations</td>
<td>Oracle Partitioning, Oracle Snapshot Replication, Oracle Streams, IBM DB2 SQL Replication, PostgreSQL Slony, MySQL Replication</td>
</tr>
<tr>
<td>Multimedia Data</td>
<td>Media Warehouses, Data Warehouses, Streaming Systems</td>
<td>Layered Data Replication, Voting With Witnesses approach, Voting With Ghosts approach</td>
<td>Replication in RDBMS, Replication in file-system level</td>
</tr>
<tr>
<td>Mobile Data</td>
<td>Systems accessed by mobile devices (mobile phones, TV, mp3 players, notebooks)</td>
<td>Lazy replication approaches, Replication approaches enabling recoverability, Middleware based replication approaches</td>
<td>Pumatech Sync-it, Intellisync, Sylo SIM, fusionOne InternetSync, Aether ScoutSync, Replication in RDBMS</td>
</tr>
<tr>
<td>Data in Real Time Processing</td>
<td>Real Time Systems</td>
<td>Replication approaches based on RCCOS algorithm, MIRROR approach</td>
<td>TimesTen In-Memory DB Replication</td>
</tr>
<tr>
<td>Spatial Objects</td>
<td>Systems supporting spatial measurement or</td>
<td>Spatial Relationship-based 2PC protocol, Replication based on Region locking.</td>
<td>GIS S/W, Gothic</td>
</tr>
<tr>
<td>construction functions</td>
<td>Replication based on Spatial Relationship-Bound Write locking</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replication for High Availability</td>
<td>System in Disaster Recovery Process</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>One-way replication, Standby replication, Redundant Array of Inexpensive Databases RAIDb</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EMC Symmetrix Remote Data Facility, Symantec Veritas Replicator, Oracle DataGuard, IBM DB2 HADR, IBM Logical VM, DRBD for Linux O</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 3

Applicable replication method

3.1 National Social Insurance Fund Organization

Social insurance is one of the modern social systems, which provide protection for citizens against risks of life that they may face. It is distinguished from other systems, which aim to provide economic security for citizens, by the special nature of its field, its styles and its technical means. It has several characteristics and the most important of these are:

- It is mandatory system, which is applied through the power of law. Its mandatory and coercive nature is attributed to the principle of solidarity among individuals, who live a community life.
- It is managed by the State, through an independent establishment, or institution.
- It is self-financed, through contributions and its resources are not mixed with public funds because they are allocated for specific purposes, which are defined by the law.
- It is not a profit-targeting system and therefore, private insurance companies do not deal in this kind of insurance.
- It is universal in the sense that it includes all workers regardless of race, gender, colour and religion.
- It guarantees in cash and in kind benefits for contributors and their families.

Sudan, like other countries of the world, know the social insurance system since the beginning of the seventies of the past century, when the social Insurance
Act of Sudanese Expatriates in 1997. This Act expanded the insurance umbrella to include Sudanese expatriates in immigration countries, on optional basis. According to the Social insurance Act, the Social Insurance Fund is a financially and administratively autonomous establishment and has a legal personality and a perpetual nature. It is entrusted with the management of the social insurance system in Sudan. The national Social Insurance Fund functions under the supervision of the Minister of Social Care and Women and Child Affairs.

The Social Insurance Fund has a Joint Council of Administration, which includes the National Pensions Fund and the National Health Insurance Fund. The organizational structure of the Fund is composed of a General Manager, Departments, Specialized Departments and 31 offices, which are diffused in all Federal States of Sudan.

3.1.1 Power and authorities of Social Insurance Fund Organization:

The Social Insurance Fund is entrusted with the implementation of the Social Insurance Act and therefore, it assumes the following responsibilities:

- Registration of employers who are subject to the Insurance Act.
- To take necessary procedures for the insurance of workers who are subject to the provisions of the law.
- Collection of contributions from employers in the different sectors of the economy.
- Payment of insurance benefits such as pensions, compensation to qualified insured workers and beneficiaries.
- To disseminate social insurance culture and enlighten insured persons and employers with the social insurance system and its objectives.

3.1.2 Finance of Social Insurance Fund Organization:
The social insurance system in Sudan is basically financed through contributions paid by employers and insured workers at the rate of 25% of the total salary liable to contribution, which the worker receives. The employer bears 17%, while 8% is born by the worker. Additional sources of finance come from fees, investment profits, and other incomes from the activities of the Fund, in addition to aid and donations.

The funds of the Social Insurance Fund in Sudan are private funds, which are allocated for the purpose of social insurance. Amounts of money payable to the Social Insurance Fund, such as contributions, have the priority of payment over all payable monies of the debtor. This means that they come immediately after legal expenses in priority. The Fund has the right to collect them though administrative detention.

Monies of the Fund such as pensions and compensations, which are paid to insured workers, in addition to movables and immovable’s and its investments, are exempted from taxes and duties.

3.1.3 Insurance Covered by of Social Insurance Fund Organization:

The provisions of the Social Insurance Act are compulsorily applied to all workers in the private sector, including foreigners, with the exception of workers in diplomatic missions. They also apply to workers of public establishments, which are self-financed. Workers in companies, which are established under the Companies Act of 1925 and workers of government-owned companies and banks, are also subject to the provisions of the Social Insurance Act. This Act applies to workers with employers who employ one, or more than one worker. Professionals, tradesmen, lawyers, doctors, mechanics, blacksmiths, carpenters etc. are also included in this Act.

In the social insurance system, the employer means every legal or natural person who employs one, or more than one worker for a salary, whatsoever may be the
kind of that salary. The definition of the term “employer” includes – for example, 
but not limited to – the following:
Establishments, factories, workshops, hotels, restaurants, cafeterias, bakeries, 
coffee shops, hospitals, medical laboratories, pharmacies, universities and private 
schools, farms, gardens, poultries etc. and every commercial establishment that 
employs one, or more than one worker.
Also, insurance coverage includes Sudanese expatriates, who have their own 
separate law.

3.1.4 Salary liable to contribution:
According to the Sudanese social Insurance Act, contributions, which should be 
paid to the Fund by the employer, are assessed at the rate of 25% of the total salary 
received by the insured worker. This includes the basic salary + living standard 
allowance + housing allowance + transport allowance + nature of work allowance 
+all fixed and permanent increments and allowance.
Telephone number of departments and office of social insurance Fixed and 
permanent increments were gradually added to the salary liable to contribution: 
30% in 2004, 60% at the beginning of 2005 and finally 100% at the beginning of 
2006.

3.1.5 Risks Covered by Social Insurance Fund Organization:
Risks, which are covered by the social insurance system, are usually divided into 
two groups: occupational risks related to the occupation such as work injuries and 
occupational diseases and human risks, which do not affect the worker alone, but 
also all members of the society like old-age, invalidity and death etc.
In accordance with the Sudanese social insurance system, the following risks are 
included in coverage:

1. work injuries and occupational-diseases insurance
2. Old-age insurance (retirement)
3. Health disability insurance
4. Death insurance

The Social Insurance Act of Sudanese Expatriates covers, in addition to the risks 1, 2 and 3 above, the risk of missing and of definite return to homeland.

3.1.6 Insurance benefits conditions of qualification and calculation

The social insurance system in Sudan provides several insurance benefits such as pensions and compensations, which are paid to insured workers and their families after fulfillment of qualification conditions. According to these conditions the risk should take place during service or during a specific period with effect from the date of termination of service. In some cases such as the case of old-age, the condition of a specific period of contribution in social insurance should be fulfilled.

3.2 National Social Insurance Fund Database Systems

In 1984 National Social Insurance Fund decided to establish computerized system for National Social Insurance Fund for the following reasons:

- The National Social Insurance Fund opened number of offices in all states of Sudan. The offices need to coverage the insured and applied the Social Insurance Law to all workers.

- The Social Insurance Laws change according to actuarial studies and the actuarial studies take place every five years.
  Numbers of the insured and numbers of the employers increases.

- Instability of the insured in one job or in the geographical areas.

- Make the organization works very easy.

- National Social Insurance Fund has four separate databases isolated from each others.

3.2.1 Technical System
The main objective of the system is registers employees and employers, generates social number for each employee, and generates identification number for each employer. Also the system records employees and employers information. Tracking the status of employees and employers in the same time calculates and record contributions paid by employers and insured workers. Finally records if the works takes pension the change in his status.

The system works in Windows Server 8 Operating System, Oracle 10g Enterprise Database and Dot Net base web base frontend. The system is Centralize system installed in the Head Office. All offices access to system by WAN.

3.2.2 ERP System (Financial and HR System)

Like any Governmental Accounting Systems the ERP System does the process of recording, analyzing, classifying, summarizing communicating and interpreting financial information about NSIF in aggregate and in detail reflecting transactions and other economic events involving the receipt, spending, transfer, usability and disposition of assets and liabilities.

NSIF ERP system has eight main components:

- Documents providing evidence of transactions
- Bank accounts through which payments and receipts are handled
- Accounting records (Payables includes pension and compensation; Receivables includes contributions of paid by employers and insured workers; Assets and liabilities including cash management cash book; ledgers etc.)
- Procedures and controls
- A means of aggregation of accounting data
- Internal accounting reports
- External accounting reports (financial statements)
Human Resources and Payroll Systems are Sub-systems contains the following:

- Position Management and Classification
- Recruitment and Staffing
- Personnel Action Administration
- Benefits Administration
- Labor-Management and Employee Relations
- Work Force Development
- Time and Attendance Processing
- Leave Processing
- Pay Processing
- Labor Cost and Distribution
- Reporting, Reconciliation, and Records Retention

ERP system is installed in the Head office. The front end of the system is Oracle 6i and the back end is Oracle 10g Enterprise. The system work in LAN located in Account and HR Departments.

1- Beneficiaries System

The beneficiary system to follow and record the situation of pensions paid for the retirement or his family.

2- Regional Offices Accounting System

- Documents providing evidence of transactions
- Bank accounts through which payments and receipts are handled
- Accounting records (Payables includes pension and compensation; Receivables includes contributions of paid by employers and insured workers; Assets and liabilities including cash management cash book; ledgers etc.)
- Procedures and controls
• A means of aggregation of accounting data
• Internal accounting reports
• External accounting reports (financial statements)

Every Office has its own system installed in its LAN. The system installed in Oracle 10g database and the user front end designed by Oracle developer 6i.

3.3 Problem Description:
NSIF Head Office always need to know the financial status including the administrative expenses of Offices besides monitoring the performances of offices towered contributions, pension and compensation. Also compare the performances between the Offices.
Gathers reports from each office take from three to five days. These delays affect negatively in the performance of the organization.
Database in regional offices is installed in different versions some of them are installed in different vendors. They want to keep current database working without changes in their structure.
The goal is to find away to monitors all transaction happen in Offices Account Systems instantaneously and simultaneously in the same time to keep the data close to the users.

3.4 Database middleware replication methods:
One of the ways is using Database Replication Methods. Traditional replication approaches implement replica control within the database kernel. Although kernel-based solutions can benefit from optimizations in the database engine, the approach has a number of shortcomings. First, it requires access to the source code of the database management system limiting the implementation of the replication to database vendors or open source solutions only. Even if the source code is available, modifying it is not an easy task. Second, such protocols are tightly integrated with the implementation of the regular database functionality, and
therefore are difficult to maintain in continuously evolving software. For the sake of portability and heterogeneity, replication protocols should be independent of the underlying database management system. As a consequence, middleware database replication has received a considerable amount of attention in the last years. Such solutions can be maintained independently of the database engine, and can even be used in heterogeneous settings. Middleware protocols match the semantics of standard database access interfaces (e.g., ODBC or JDBC), which makes it straightforward to migrate from centralized to replicated environments.

Database middleware provides a number of important benefits as, an interface to an application, the ability to convert the application language into something understandable by the target database (e.g. SQL), the ability to send a query to a database over a network, the ability to process a query on the target database.

Most of the database vendors adopt or have their own database middleware software. As an example Oracle GoldenGate software was acquired from the software company GoldenGate in 2009, IBM has Infosphere software and Dell has Shareplex software, for the following reasons:

- Database middleware replication software becomes tools of data cloud.
- Database middleware replication software can work as migrations tools.
- Database middleware replication software easily replicates data between different database vendors and different database versions.

### 3.5 Used Tools

The middleware application tool I used was SymmetricDS Pro version 3.7.30 server. SymmetricDS Pro is chosen because it is open source (license OMT) solution for Data Replication, Change Data Capture, and Data Transformation for both databases and file systems in a heterogeneous enterprise environment.
Performance and scalability allow it to replicate thousands of systems asynchronously in near real time. With flexible configuration, powerful scripting, and rich programming interfaces, SymmetricDS can be extended to meet a wide range of data synchronization requirements.

3.5.1 Database Replication Supported by SymmetricDS

- Support for most major databases: Oracle, IBM DB2, SQL Server, MySQL, MariaDB, Informix, Firebird, Sybase ASE, SQL Anywhere, PostgreSQL, Greenplum, Redshift, Interbase, Apache Derby, HSQLDB, H2, SQLite and MongoDB.
- Trigger-based Change Data Capture (CDC) system replicates data in near real time with low system overhead.
- Transaction awareness groups row changes to be committed together.
- Horizontal and vertical data filtering for subsets of a table.
- Transformation for translation, enhancement, and filtering of data.
- Map data between columns and tables to bridge different applications.
- Detect conflicts and resolve using automated rules or manual notification.
- Efficient data protocol for low-bandwidth operation.
- Encryption of data stream for secure communication.
- Withstand periods of downtime and automatically recover from a network outage.
- Deploy nodes across local or wide area networks.
- SymmetricDS can run on any system with a Java Runtime Environment. It is adaptable, running on small memory systems and scaling to support large central installations.

Also SymmetricDS is available commercially with following auxiliary benefits:

- Web management console for easy setup and support.
- Configure and monitor all nodes from central location.
- Scripting support for event handling and data transformation.
- Java programming interface allows customizations and extensions.
- Deploy as a standalone server, web application server, or embedded application.

### 3.5.2 SymmetricDS Hardware Requirements:
- Windows, Linux, Solaris, Mac OS X, Android, iOS
- Java SE version 6 or newer

SymmetricDS supports both single-homed and multi-homed distributed middleware replication

Each database has its own instance of SymmetricDS running on a server that is attached to the database (#1, #2 and #3). A one to one relationship between SymmetricDS instance and database is called a single-homed setup.
Fig 3.1 Single-homed distributed middleware replication

Centralized Replication Middleware (multi-homed setup)
Multi-homed setup where a single instance of SymmetricDS is attached to, and responsible for, multiple databases

3.5.3 SymmetricDS Architecture
Each subsystem in the node is responsible for part of the data movement and is controlled through configuration. Data flows through the system in the following steps:
3.5.4 How SymmetricDS Works

A node is responsible for synchronizing the data from a database or file system with other nodes in the network using HTTP. Nodes are assigned to one of the node Groups that are configured together as a unit. The node groups are linked together with Group Links to define either a push or pull communication. A pull causes one node to connect with other nodes and request changes that are waiting,
while a push causes one node to connect with other nodes when it has changes to send. Each node is connected to a database with a Java Database Connectivity (JDBC) driver using a connection URL, username, and password. While nodes can be separated across wide area networks, the database a node is connected to should be located nearby on a local area network for the best performance. Using its database connection, a node creates tables as a Data Model for configuration settings and runtime operations. The user populates configuration tables to define the synchronization and the runtime tables capture changes and track activity. The tables to sync can be located in any Catalog and Schema that are accessible from the connection, while the files to sync can be located in any directory that is accessible on the local server.

Fig 3.4 show how SymmtricDS works

At startup, SymmetricDS looks for Node Properties Files and starts a node for each file it finds, which allows multiple nodes to run in the same instance and share resources. The property file for a node contains its external ID, node group, registration server URL, and database connection information. The external ID is the name for a node used to identify it from other nodes. One node is configured as
the registration server where the master configuration is stored. When a node is started for the first time, it contacts the registration server using a registration process that sends its external ID and node group. In response, the node receives its configuration and a node password that must be sent as authentication during synchronization with other nodes.

3.5.5 SymmetricDS Supported Topologies

3.5.5.1 Data synchronization / consolidation
Data from multiple databases across the enterprise are synchronized and combined, working across low band width connections and withstanding periods of network outage. Thousands of databases can be connected to a central one using one or more tiers of synchronization.

3.5.5.2 Workload distribution
The operational database is replicated to a data warehouse or a reporting database. The main application continues to use the operational database, while the secondary database is used for reporting and analysis. The data can be filtered and transformed to enhance reporting.

3.5.5.3 Bridge between databases
Data is transformed during replication between different databases. Secure web transport protocols are used to replicate from a protected database to a front-end database.

3.5.5.4 Database backup
Critical database applications are protected by continuously replicating to a standby database. Changes are sent to the standby, which may be on a local or wide area network. In an emergency, the application is reconnected to the standby database. Once the production database is available again, SymmetricDS will replicate changes to it from the standby. The switch-over can also be used to
perform planned maintenance and system upgrades. Since SymmetricDS is flexible enough to replicate between different databases and table layouts, some tables of the upgraded database may have new columns in old tables.
Chapter 4
Implementation and experimental evaluation of middleware replication in NSIF System

4.1 Introduction
This chapter introduces the experimental evaluation implemented on NSIF system using SymmetricDS pro. The main purpose of these experiments is verification of middleware replication software presented in chapter 3. The following factors are taken under consideration:

- High scalability level,
- Transaction processing in parallel,
- Portability and possibility of the usage in heterogeneous environments,
- Reduced communication,
- Resistance to failures,
- Easiness of introducing changes.

4.2 SymmetricDS Pro Setting
SymmetricDS captures synchronization data using database triggers. SymmetricDS’ Triggers are defined in the TRIGGER table. Each record is used by SymmetricDS when generating database triggers.

4.2.1 Trigger Example

4.2.1.1 Insert Trigger:

insert into sym_trigger (TRIGGER_ID, SOURCE_CATALOG_NAME, SOURCE_SCHEMA_NAME, SOURCE_TABLE_NAME, CHANNEL_ID, SYNC_ON_UPDATE, SYNC_ON_INSERT, SYNC_ON_DELETE, SYNC_ON_INCOMING_BATCH, NAME_FOR_UPDATE_TRIGGER, NAME_FOR_INSERT_TRIGGER, NAME_FOR_DELETE_TRIGGER, SYNC_ON_UPDATE_CONDITION, SYNC_ON_INSERT_CONDITION, SYNC_ON_DELETE_CONDITION, ...,
SYNC_ON_DELETE_CONDITION,EXTERNAL_SELECT,
TX_ID_EXPRESSION,EXCLUDED_COLUMN_NAMES,
CREATE_TIME,LAST_UPDATE_BY,LAST_UPDATE_TIME)
values ('SALE_TRANSACTION_DEAD',null,null, 'SALE TRANSACTION', 'transaction',
0,0,0,0,null,null,null,null,null,null,
current_timestamp,'demo',current_timestamp);
insert into sym_router
 ROUTER_ID,TARGET_CATALOG_NAME,TARGET_SCHEMA_NAME,
TARGET_TABLE_NAME,SOURCE_NODE_GROUP_ID,TARGET_NODE_GROUP_ID,ROUTER_ROUTER_EXPRESSION,SYNC_ON_UPDATE,SYNC_ON_INSERT,SYNC_ON_DELETE,
CREATE_TIME,LAST_UPDATE_BY,LAST_UPDATE_TIME)
values ('CORP_2_STORE',null,null,null, 'corp','store',null,null,1,1,1,
current_timestamp,'demo',current_timestamp);
insert into sym_trigger_router (TRIGGER_ID,ROUTER_ID,INITIAL_LOAD_ORDER,
INITIAL_LOAD_SELECT,CREATE_TIME,LAST_UPDATE_BY,LAST_UPDATE_TIME)
values ('SALE_TRANSACTION_DEAD','CORP_2_REGION',100,null,
current_timestamp);
INSERT INTO sym_router
(router_id,target_catalog_name,target_schema_name,target_table_name,
source_node_group_id,target_node_group_id,
router_type,router_expression,sync_on_update,sync_on_insert,sync_on_delete,
create_time,last_update_by,last_update_time)
VALUES
('server_2_client',null,null,null,'server','client','default',null,1,1,1,
current_timestamp,'example',current_timestamp);

4.2.3 Sync Triggers Job

SymmetricDS examines the current configuration, corresponding database triggers, and the underlying tables to determine if database triggers need created or updated. The change activity is recorded on the TRIGGER_HIST table with a reason for the change. The following reasons for a change are possible:

- N - New trigger that has not been created before
- S - Schema changes in the table were detected
- C - Configuration changes in Trigger
- T - Trigger was missing

A configuration entry in Trigger without any history in Trigger Hist results in a new trigger being created (N). The Trigger Hist stores a hash of the underlying table, so any alteration to the table causes the trigger to be rebuilt (S). When the last_update_time is changed on the Trigger entry, the configuration change causes the trigger to be rebuilt (C). If an entry in Trigger Hist is missing the corresponding database trigger, the trigger is created (T).
The process of examining triggers and rebuilding them is automatically run during startup and each night by the SyncTriggersJob. The user can also manually run the process at any time by invoking the syncTriggers() method over JMX.

### 4.3 Database triggers list

The SymmetricDS' Triggers used are listed in table 4.1.
Table 4.1 Trigger table

<table>
<thead>
<tr>
<th>TRIGGER_ID</th>
<th>CHANNEL_ID</th>
<th>RELOAD_CHANNEL_ID</th>
<th>SYNC_ON_UPDATE</th>
<th>SYNC_ON_INSERT</th>
<th>SYNC_ON_DELETE</th>
<th>SYNC_ON_INCOMING_BATCH</th>
<th>SYNC_ON_UPDATETYPE_CONDITION</th>
<th>SYNC_ON_INSERTCONDITION</th>
<th>SYNC_ON_DELETECONDITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>active_type</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>bank</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>closing_year</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>employees</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>estimations</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>insured</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>insured_hist</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>ins_stop_reasons</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>look</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>menu_user</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>no_salary</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>offices</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>payable</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>revenues</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>revenue_types</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>sectors</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>stop_reasons</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>titles</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
<tr>
<td>update_salary</td>
<td>default</td>
<td>reload</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1=1</td>
<td>1=1</td>
<td>1=1</td>
</tr>
</tbody>
</table>

4.4 Implementation results

Results obtained after implementing symmetricDS Pro on three different nodes are provided in this section. These nodes are Omdurman, Amarat, and South Khartoum nodes. Since a huge number of tables and graphs have been generated for batch transfer in each node, only some of them will be illustrated and discussed here.

4.4.1 Omdurman node results
In figures 4.1 and 4.2 below samples of the raw transfer from Omdurman node are illustrated.

Figure 4.1 presents the raw transfer of batch 16 while figure 4.2 shows the raw transfer of batch 29. The activity column records the incoming and outgoing raws.

Figure 4.1 Capture of the raw transfer of batch 16 for Omdurman node
Figure 4.2 Capture of the raw transfer of batch 29 for Omdurman node

4.4.2 Amarat node results

Some examples of the results obtained from Amarat node are presented in figures 4.3 and 4.4 below. Figure 4.3 presents the raw transfer of batch 14 while figure 4.4 shows the raw transfer of batch 22.
Figure 4.3 Capture of the raw transfer of batch 14 for Amarat node

Figure 4.4 Capture of the raw transfer of batch 22 for Amarat node
4.4.3 South Khartoum node results

Some examples of the results obtained from South Khartoum node are presented in figures 4.5 and 4.6 below. Figure 4.5 presents the raw transfer of batch 17 seen from the node side while figure 4.6 shows the raw transfer of the same batch seen from the server side.

Figure 4.5 Node side capture of the raw transfer of batch 17 for South Khartoum node
Figure 4.6 Server side capture of the raw transfer of batch 17 for South Khartoum node

4.4.4 Error messages results

The symmetricDS pro application can identify errors occurring during data transfer and generate error messages for the user. These errors may arise from network connection failure, configuration mistakes, or database setting. Figure 4.7 shows an error message due to network connection failure. An error message because of a configuration mistake is given in figure 4.8. The database setting errors have been avoided by creating a database table which contains an ID for each node. This ID is used as part of the primary key in each database.
Figure 4.7 Error message due to network connection failure

Figure 4.7 Error message due to a configuration mistake
Chapter 5

Conclusion and Recommendation

5.1 Conclusion

Chief Executive Officers (CEOs) face problem when they need information from the regional offices. The difference in database versions and database vendors’ prevented them from getting reports directly from the regional systems. Also sometimes connection dropdown or interrupt between Head office and regional offices cause the communication to be missed between the sites. The consequences of missing communication affects the data processed and inserted during the communication interruption and it delays the time of finishing the jobs.

The replication method can be used as solution to overcome all the above problems. Replication is a set of technologies for copying and distributing data and database objects from one database to another and then synchronizing between databases to maintain consistency. Replication allows maintaining same database multiple copies at different locations. Log shipping and mirroring allows maintaining complete database redundancy.

In this research the SymmetricDS Pro application was used as replication middleware layer to resolve NSIF data process delaying and replaces the exchange of data through SMS files by exchanging real data between nodes and head office. This enhances NSIF work and allows the monitoring of each office continuously. The results obtained from Omdurman, Amarat, and South Khartoum nodes reflect that the objectives of this research have been achieved successfully.
5.2 Future work

As future work based on the work carried in this research the following can be recommended:

1. Since Symmetric DS Pro is open source code software it can be modified to include a friendly GUI to make line command easier for users
2. The different database systems explained in chapter 3 can be integrated into one general system to serve different applications
3. For large organizations with huge data transfer rates the symmetricDS pro implementation can also tested for reliability
4. The replication solution method implemented for NSIF system can be done using other different replication methods and compared

5. References

88. [76] M. Patino-Mart_nez, R. Jimenez-Peris, B. Kemme, and G. Alonso. Middle-R:

140.

141.