JOURNAL OF BAKU ENGINEERING UNIVERSITY- MATHEMATICS AND COMPUTER SCIENCE
2024. Volume 8, Number 2 *Pages 120-129*

UOT:002.6:025.3/.4
DOI: https://doi.org/10.30546/09090.2025.01.2027

ARCHITECTURE AND MODEL OF A SPECIAL-PURPOSE ELECTRONIC DOCUMENT MANAGEMENT SYSTEM WITH A DISTRIBUTED STRUCTURE

Vagif GASIMOV[1], MAMMADZADA Nargiz[*2]

[1]Baku Engineering University, Baku, Azerbaijan

vaqasimov@beu.edu.az

[2]Azerbaijan Technical University, Baku, Azerbaijan

mammadzada.nargizw@gmail.com

ARTICLE INFO

Article history:
Received: 2025-04-14
Received in revised form:2025-04-15
Accepted:2025-04-16
Available online

Keywords:
Distributed databases, fragmentation, replication, database conflicts, database synchronization

ABSTRACT

The importance of forming and bringing the organization of databases of medical institution branches into a more suitable form in terms of their structure, accessibility and flexibility of applications, and the development of methods in this regard are presented in the article. Thus, having each branch's own database increases the flexibility of local queries, minimizes duplication of general data, and provides a structure for distributing certain data in a way that is specific to the functions and medical aspects of the branches. Many existing methods for distributed databases have been considered and the suitability of some of them for medical documents has been noted.

1. Introduction

The geographical location of medical institutions affects the structure of the databases they use [1]. As is known, branches of medical institutions differ from each other in terms of functionality. Thus, the equipment or laboratory analysis technique system available in branch X of a medical institution may not be available in its other branches. This leads to differences in the types of documents produced in each of them and necessitates the organization of the database content between branches in accordance with their functionality. On the other hand, in order to ensure the organization of each branch's own databases, it is also proposed to ensure their geographical organization. Thus, when a patient who applies to a branch applies to that branch for the second time, if the query for his previous information is made not from the center, but from the database of the branch where he is currently located, then the query result can be obtained faster [2]. On the other hand, collecting existing databases in one center and creating replication is considered an important factor in ensuring that it becomes a more sustainable system [3]. In such a case, it is more appropriate to create a database system where each branch has its own database, as well as a replication (copy) of the databases in each other branch in a single center (Figure 1).

[*]Mammadzada Nargiz.
E-mail addresses: **mammadzada.nargizw@gmail.com**

In distributed databases, data can be distributed geographically or functionally [4]. A distributed database system is a network of databases where data is stored in multiple physical or logical locations [5]. In general, when a query is received by the system, the response is generated across all databases or by routing it to specific databases depending on the type of query [4]. Database distribution allows for faster query execution and better error recovery by distributing large amounts of data across multiple physical databases [2,6].

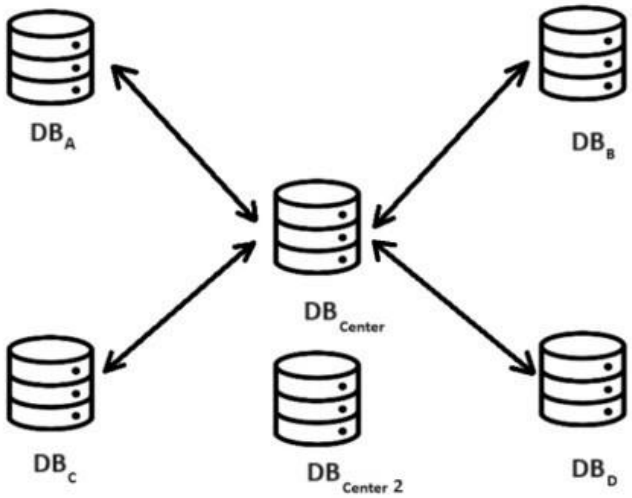

Figure 1. Organization of special-purpose distributed databases

There are two main concepts in distributed database management:

-Replication process

In this method, the data available in each database is copied to two or more different databases. If all the available data is copied to all databases, it is called a fully replicated system. Data replication is a method that increases the durability and consistency of databases. Database consistency is the fact that when users access any database node in the system, the same query produces the same result on each node. At the same time, replication is an effective factor in ensuring the security of databases. Thus, data deleted or lost in one database can be restored through other replications. The replication process ensures that data is constantly synchronized and copied with other databases [6].

Advantages:

- Minimizes loss by increasing overall data availability.
- It causes queries to be answered faster in the system, as queries are executed in parallel in distributed databases compared to a central database , so they are answered faster.
- When users send requests to different databases, the system becomes more resilient, as lost data is resynchronized and restored through other databases.
- It speeds up the process of reading from databases, as the request entering the system is redirected to the nearest node, minimizing network delays.

Disadvantages:

- The data in the databases must be constantly synchronized and updated, which increases the traffic load of the databases.

- Any changes made to any database node must be made to other databases as well, otherwise inconsistencies may arise between databases.
- Updating data causes delays, as each change is executed taking into account other databases.
- If the synchronization process is not performed correctly, inconsistencies arise between databases.
- Fragmentation process:

In this method, data fragments are distributed across different databases. When these fragments are combined, the original whole data is obtained again. Since the data is simply distributed in the fragmentation process, there is no consistency problem between them.

Advantages:

- It increases the productivity and performance of the system by allowing the parallel execution of multiple queries **(**especially for queries related to different fragments**)** [7].
- Storing the data that each center uses most frequently in databases located in close geographical locations leads to more efficient searches[2,4].
- Ensures that databases are more secure. As a result, if one node is attacked by hackers, not all of the data can be accessed, which ensures that the data is more secure overall [7].

Disadvantages:

- Distributing data across multiple databases and making queries on them more complex.
- The performance of applications depends on the speed of the network connecting the databases.
- Complexity of query optimization

The fragmentation process is divided into two parts in terms of the distribution of elements:

Horizontal fragmentation and vertical fragmentation. Horizontal fragmentation is basically the division of tables used in databases into rows and storing them in different databases (Figure 2). Vertical fragmentation is the process of dividing tables used in databases into columns or managing each table by storing it in a different database (Figure 3). To improve the performance of databases for enterprises, a hybrid distributed database structure is often used. Since both replication and fragmentation processes are used in databases, it allows ensuring the durability, consistency, and security of databases.

2. Building distributed databases for medical records

Databases for medical document databases should be structured as follows:

- The data generated in each branch should be stored in its own database. Thus, the data of those who apply to the branch is stored in the database of that branch and the results of the existing functions in each branch are stored in the database of that branch. This ensures both geographical and functional division of the databases.
- The data generated in each branch is also transmitted to the central database and copied there. The central database can be one or more than one. This structure makes the databases more secure by ensuring that they are more resistant to loss.

- During a search in one branch, information not available in the current database is generally queried through other branches or through the central database.

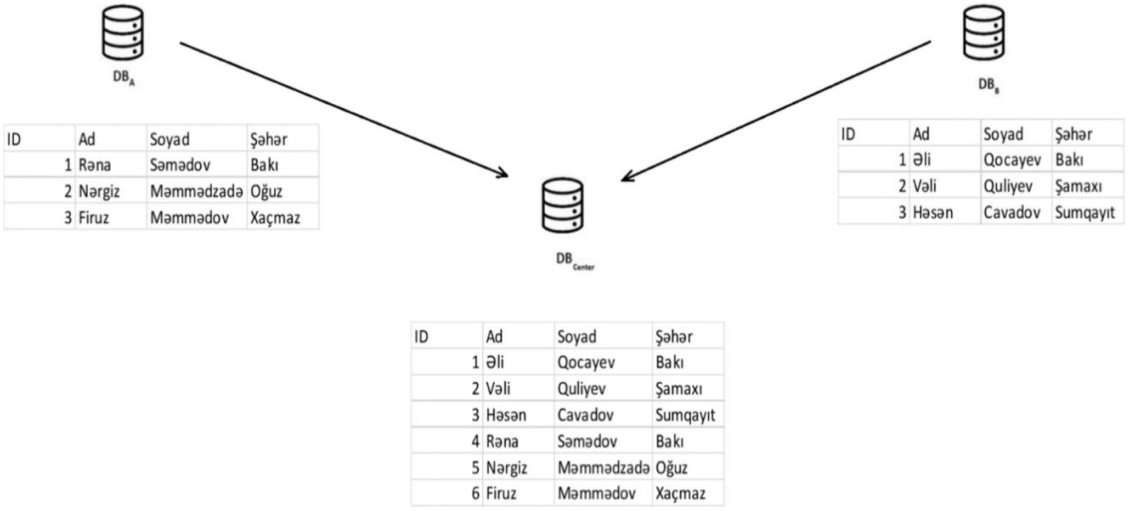

Figure 2. Database replication with horizontal distribution: how data within a table is distributed row by row and replicated at a central node

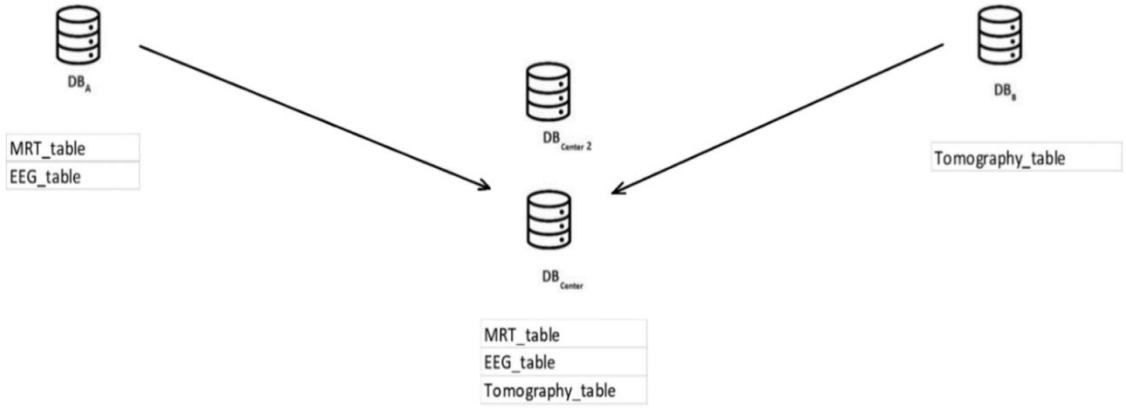

Figure 3. Database replication with vertical distribution: distribution of tables created in each node according to their functions across nodes and replication at the central node

3. Current methods used in database replication

The main issue in the replication process is the correct management of copying information generated in one database to other databases. Since a small error that may occur in the organization of work can lead to a violation of consistency between databases or a loss. There are two types of approaches to solving this problem:

1. Master-Slave approach or leader replication: Here, synchronization occurs by transmitting every change made on the master to the slave database(Figure 4). In this method, write operations are mostly performed on the master ,while read processes are performed from the slave databases [8].

2. Another replication management approach is leaderless replication, where each database is considered a master, where each database can perform read and write operations, and where each database communicates with other databases to ensure consistency among themselves[6,9].

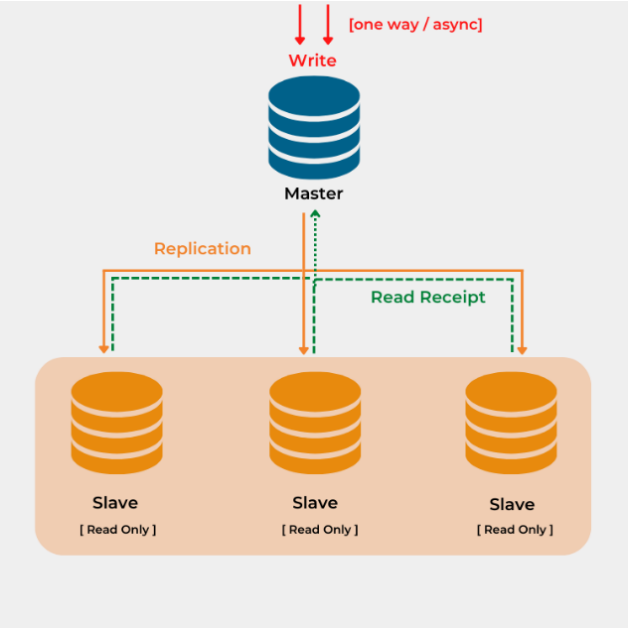

Figure 4. Master-Slave structured replication control method[11]

Both approaches have their own advantages and disadvantages:

Advantages of led replication:

1. Centralized management: The replication process is managed from a single center, which ensures easy management of requests and authorization processes.

2. Consistency: Leader-based replication is a very powerful method for ensuring consistency between databases, and in such systems, consistency is guaranteed to be high. Here, writes made to the leader are propagated to its followers, ensuring that all followers share the same data as the leader.

3. Simplicity: Configuration and management in leader-based replication are simpler than in the leaderless approach.

Advantages of leaderless replication:

1. High availability: In this method, since each of the system's databases participates in both write and read processes, the system's durability is significantly higher. Thus, in leader-managed replications, if the leader database fails, the system stops writing operations until another leader is elected.

2. High durability of database writes. In leaderless replication, the write process is more durable because it can be performed on any node in the system[9].

3. Load distribution: Through replication, data can be distributed to different geographical locations, so that requests to the system can be made over nearby locations, which supports increased performance[9].

There are many methods created to date for both approaches:

Several algorithms are used for data replication in distributed databases, and the choice of algorithm depends on factors such as consistency requirements, network conditions, and specific characteristics of the distributed system. Commonly used replication algorithms are as follows:

Primitive n -type replication:

In this approach, one node is designated as the primary node and the others are backup nodes. All write operations go to the primary node and the changes are then replicated to the backup nodes [8].

Voting-based Replication:

In this method, data written to one or more replicas is voted on among them to be replicated to other nodes. If there are n replicas, then data accepted by at least n/2+1 majority votes is written to the databases, achieving consistency[6].

Version vectors:

A version vector helps to know the versions of data stored in distributed databases and to track replication and consistency between them with these vectors. Along with consistency, it also helps to detect conflicts that may occur in the system[10].

Anti-Entropy Protocol:

This protocol performs periodic comparison and reconciliation of data between replicas to ensure consistency[6,11].

CRDTs (Conflict-Free Replication Data Structures):

CRDTs are data structures designed to be replicated across distributed nodes without the need for a centralized coordinator[12].

They ensure consistency by allowing simultaneous updates without conflicts.

4. Algorithm for managing replications in special-purpose databases

In the proposed model that ensures the distribution of medical documents between branches, the database located in each branch not only stores its own data, but also synchronizes with the central database, preventing data loss. In this system, the writing and reading process can be performed in each database. And this makes the system compatible with the leaderless replication model, despite the fact that it is synchronized with the central server. Here, a vector of versions that are interconnected with each other is used to manage replications.

Unlike special-purpose databases, let's initially look at the transaction version-based replication management method for managing n number of replications. Let's assume that each of the n number of databases performs the write process separately. Here, after each write operation, the changes must be synchronized with other databases. For the synchronization process, initially, each write transaction query rows (sql queries) are added to a table called “Transactions” in each database. The structure of the Transactions table reflects whether the rows are transactions and the columns are whether those transactions have been executed in n number of databases. And initially, all cells have the value 0. In whichever database the query is executed, the value 1 is written to the cell located at the intersection of the row and column corresponding to the query and that database in that table. If a database is down (down) and then restored (up), it takes the data in this Transactions table from other databases and combines them and executes them in the appropriate sequence on the current node. The advantage of this method is that the databases simply check the transaction tables in all databases with each other, which does not check whether all the data in each database is synchronized or not (as in the Snapshot method), which helps to perform the synchronization process more easily . Another issue in managing replications is the conflicts that may arise between them. Thus, when writing to the same data on

two or more database nodes, conflicts arise between the values of the same data between these nodes, which leads to the problem of which data should be taken as the basis during synchronization. For managing replications, it is necessary to compare the synchronization status of transactions in the databases and to compare the versions of the changed data. Let's take a closer look at the above method:

Let's imagine that there are 4 replication databases in the system. A Tr_1 transaction is entered for a write operation to database 1. In this case, this transaction is initially added to the transaction table of the current database and, as soon as the transaction is executed here, it is assigned to the cell 1 located at the intersection of Db_1 and Tr_1 in that table (where Db_n represents the database on node 1). Then, a synchronization request is sent from this database to other databases. If each of the remaining 3 databases is up, this Transaction is added to their tables and, as soon as it is executed there, it is assigned to the corresponding cell 1 in that table. During the execution process of 1 in each database, information about this is sent to other databases, and in each sent database, to reflect the execution of this Transaction in other databases, this transaction is assigned to the corresponding cell 1 in the Transaction table according to the database execution. And finally, whichever database executed the query last, after all databases have executed this query, this transaction row is deleted from all tables. Thus, only queries for which the synchronization process has not been completed remain in this transaction table.

	Database 1			
	Db1	Db2	Db3	Db4
Tr1	1	0	1	0
Tr2	1	0	0	0
Tr3				
Tr…				
Trn				

	Database 2			
	Db1	Db2	Db3	Db4
Tr1				
Tr2				
Tr3				
Tr…				
Trn				

	Database 3			
	Db1	Db2	Db3	Db4
Tr1	1	0	1	0
Tr2				
Tr3				
Tr…				
Trn				

	Database 4			
	Db1	Db2	Db3	Db4
Tr1				
Tr2				
Tr3				
Tr…				
Trn				

a) The state of the databases before synchronization

	Database 1			
	Db1	Db2	Db3	Db4
Tr1	1	1	1	1
Tr2	1	1	1	1
Tr3				
Tr…				
Trn				

	Database 2			
	Db1	Db2	Db3	Db4
Tr1	1	1	1	1
Tr2	1	1	1	1
Tr3				
Tr…				
Trn				

	Database 3			
	Db1	Db2	Db3	Db4
Tr1	1	1	1	1
Tr2	1	1	1	1
Tr3				
Tr…				
Trn				

	Database 4			
	Db1	Db2	Db3	Db4
Tr1	1	1	1	1
Tr2	1	1	1	1
Tr3				
Tr…				
Trn				

b) State of databases after synchronization

Figure 5. Description of the method of leaderless management of replications in a distributed database

Let's say Db_2 and Db_4 are down. A transaction enters Db_1 and is sent to the others for synchronization. In this case, the values in the Transaction table in Db_1 and Db_3 will be $Tr_1\{1,0,1,0\}$, respectively. Then, when Db_4 becomes down, another Tr_2 transaction enters the system. In this case, the only up database Db_1 will have $Tr_1\{1,0,1,0\}$, $Tr_2\{1,0,0,0\}$. Then, as soon as each down database becomes up, it synchronizes the Trs from all other databases to its Transaction table and, as each one is executed, it sends information about this to the other up databases as well as its own database. And thus, the databases synchronize with each other and after synchronization, the values corresponding to the transactions in each database take the status $Tr_1\{1,1,1,1\}$, $Tr_2\{1,1,1,1\}$. The worst case scenario that can occur in the transaction response process is that all down databases are up while other up databases are down. Then, when the databases are up, there is a problem of confusion about which transaction is executed in which sequence. To prevent this, the initial execution date for each transaction in the Transactions table (the date that reflects the first execution of the transaction in any of the databases) is recorded in front of each transaction in the Transactions table. When this situation arises, the transaction dates are executed sequentially, and consistency and correct sequence will be preserved. Another issue in the replication synchronization process is conflict management. If two databases perform a write operation on the same data at the same time, then during synchronization, there is confusion about which data will be taken as the basis and which will be copied to the others. In this case, this issue is solved to some extent with the execution date of the transactions. In this method, the execution date of the transactions on the same data is selected as the basis and synchronized with other databases. In the transaction table, if all values for each transaction are 1, then the information about the execution of that transaction is deleted from the transaction table in the database of each node. This process is checked in the database of the last synchronized node and sends information about the deletion to other databases. The advantage of this method is that the system is more durable, so even if only 1 of the n databases responds to the server, the system can continue to work. An example of the implementation of these processes is illustrated in Figure 5.

It was noted that in the database structure shown above for special-purpose databases, replications are performed only on the central database. Although the proposed synchronization process was considered for n number of databases , this method can also be easily applied to databases without a leader and replicated only on a central server.

5. Conflict management solutions in organizing leaderless replications

One of the main problems in managing leaderless replication is the occurrence of conflicts. Thus, if two different nodes perform operations on the same data at the same time, or if two different nodes make changes to the same data during a period of time when the connection with each other is lost, two different versions of the same data will exist when the connection is restored. Various methods have been implemented to prevent such situations to date.

If an internet connection is restored on a node that has lost connection to other nodes, the following methods are available to balance the difference between them:

1. **Last Write Wins**: This method provides authentication between nodes by providing a timestamp for each transaction that occurs, determining which transaction came last.

2. **Conflict-free Replicated Data Types**: A CRDT is a data structure that can successfully merge update operations.

3. **Manual intervention**: In some cases, conflicts between databases can be resolved by manually intervening by analyzing the error that occurred.

With the help of these methods, databases can be identified and consistency between them can be ensured.

In databases intended for medical documents, the use of time stamps is considered appropriate for resolving conflicts. Thus, during the execution of each transaction in each node, the IDs of the affected objects are collected in a table using triggers and stored in the nodes. Later, when the nodes exchange information about the transactions, the time stamp reflecting the transaction and the execution time of the transaction and the ID of the affected object are compared. To ensure the accuracy of the time stamp, each node must use online international time indicators and the time indicators must be synchronized at certain time intervals when there is an Internet connection. If two separate nodes make changes to an object with the same ID without informing each other, the first or the last one among them is selected and sent to all nodes, and the transaction that is not selected is rolled back on the node where it was executed. The algorithmic sequence of this process is as follows:

For simplicity, let's look at the synchronization process between two nodes, A and B:

1. Both nodes that are disconnected from each other retain the transactions executed on them until the synchronization process. $\sum Tr_i^A$ and $\sum Tr_j^B$

2. Between them is restored, the transactions in them are combined with hash-structured sets of object IDs: $Tr_{conflicts} = HashSet_{by\ Object\ ID}(\sum Tr_i^A + \sum Tr_j^B)$

3. $Tr_{conflicts}$ The timestamp for each object is selected for each of the initial incoming requests, and the rollback process is performed on the database where the other request was executed, and the newly selected execution request is executed on all nodes.

As a result, existing methods for managing the load of distributed structured special-purpose databases, both fragmentation and functional and geographical distribution of data between databases, as well as organizing their replication to increase the durability of databases, were reviewed, and appropriate methodologies and algorithms were developed for such special-purpose databases.

Conclusion:

The article examines the advantages of organizing databases in a distributed manner for medical documents, and develops their structure in a manner appropriate to the purpose. The methods used for replication, fragmentation, and conflict resolution of databases, which are the main issues for distributed databases, are examined and developed in a manner appropriate for medical documents. The structures formed aim to increase system performance, increase file availability, and increase system security by minimizing losses.

REFERENCES:

1. Haux, R. (2006). Health information systems – past, present, future. *International Journal of Medical Informatics* , 75(3-4), 268-281. DOI: 10.1016/j.ijmedinf.2005.08.002 *(Health information systems, geographic presence)*
2. Özsu, MT, & Valduriez, P. (2011). *Principles of Distributed Database Systems* (3rd ed.). Springer. *(Distributed DB fundamentals, fragmentation, query speed, data locality)*
3. Tanenbaum, AS, & Van Steen, M. (2007). *Distributed Systems: Principles and Paradigms* (2nd ed.). Prentice Hall. *(Distributed systems, availability, fault tolerance, replication)*
4. Özsu, MT, & Valduriez, P. (2011). *Principles of Distributed Database Systems* (3rd ed.). Springer. *(Fragmentation types, replication basics, distribution strategies)*
5. Bernstein, PA, & Newcomer, E. (2009). *Principles of Transaction Processing* (2nd ed.). Morgan Kaufmann. *(Distributed DB definition, replication, consistency)*
6. Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems* . O'Reilly Media. *(Replication (with/without leader), consistency models, fragmentation (partitioning), conflict resolution (LWW, CRDT), quorum, anti-entropy, availability)*
7. Navathe, SB, Elmasri, R. (2016). *Fundamentals of Database Systems* (7th ed.). Pearson. *(Fragmentation, performance, security aspects)*
8. Kleppmann, M. (2017). Chapter 5: Replication. In *Designing Data-Intensive Applications* . O'Reilly Media. *(Master-Slave/Primary-Backup replication, advantages, limitations)*
9. Kleppmann, M. (2017). Chapter 5: Replication. In *Designing Data-Intensive Applications* . O'Reilly Media. *(Multi-Leader/Leaderless replication, advantages, conflicts, high availability)*
10. Parker Jr, DS, Popek, GJ, Rudisin, G., Stoughton, A., Walker, BJ, Walton, E., Chow, JM, Edwards, D., Kiser, S., & Kline, C. (1983). Detection of mutual inconsistency in distributed systems. *IEEE Transactions on Software Engineering* , SE-9(3), 240-247. DOI: 10.1109/TSE.1983.236733 *(Concept of version vectors) - Or Kleppmann (2017) explains this concept.*
11. https://diadem.in/blog/how-to-configure-mysql-master-slave-replication/
12. Shapiro, M., Preguiça, N., Baquero, C., & Zawirski, M. (2011). Conflict-free replicated data types. *Proceedings of the 13th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS'11)* , 386-400. DOI: 10.1007/978-3-642-24550-3_29 *(Basics of CRDTs) - Or Kleppmann (2017) explains this concept.*