A Survey On Fragmentation In Distributed Database Systems

Mohsen Taki (✉ taki_mohsen@yahoo.com)

Islamic Azad University <https://orcid.org/0000-0003-2573-0950>

Mohammadreza Mollahoseini Ardakani

Islamic Azad University

Survey paper

Keywords: database, distribute systems, fragmentation, horizontally, mixed, table, vertically.

Posted Date: April 13th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-395198/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Abstract

One of the most critical aspects of distributed database design and management is fragmentation. If the fragmentation is done properly, we can expect to achieve better throughput from such systems.

The primary concern of DBMS design is the fragmentation and allocation of the underlying database. The distribution of data across various sites of computer networks involves making proper fragmentation and placement decisions. The first phase in the process of distributing a database is fragmentation which clusters information into fragments. This process is followed by the allocation phase which distributes, and if necessary, replicates the generated fragments among the nodes of a computer network. The use of data fragmentation to improve performance is not new and commonly appears in file design and optimization literature.

An efficient functionality of any distributed database system is highly dependent on its proper design in terms of adopted fragmentation and allocation methods. Fragmentations of large, global databases are performed by dividing the database horizontally, vertically or combination of both. In order to enable the distributed database systems to work efficiently, the fragments have to be allocated across the available sites in such a way that reduces communication cost of data.

In this article, we have tried to describe the existing methods of database fragmentation and have an overview of the existing methods. Finally, we conclude with suggestions for using machine learning to solve the overlap problem in fragments.

1. Introduction

Distributed database systems comprise a single logical database that is partitioned and distributed across various sites in a communication network. Database technology has become prevalent in most business organizations. Distributed Database System (DDS) are becoming more affordable and useful. A DDS typically consist of a number of distinct yet interrelated databases (fragments) located at different geographic sites which can communicate through a network[42].

Typically, such a system is managed by a distributed database management system (DDBMS). Each site of the DDS has its own hardware and is capable of autonomous operation. A site participates in the execution of global transactions involving databases at two or more remote sites[42].

In other definition distributed database is a collection of data that logically belongs to the same system but is spread over the sites of a computer network. A distributed database management system (DDBMS) is defined as the software system that provides the management of the distributed database system and makes the distribution transparent to the users [17, 33]. It is not necessary that database system have to be geographically distributed. The sites of the distributed database can have the same network address and may be in the same room but the communication between them is done over a network instead of shared memory[9].

The primary concern of DBMS design is the fragmentation and allocation of the underlying database. The distribution of data across various sites of computer networks involves making proper fragmentation and placement decisions. The first phase in the process of distributing a database is fragmentation which clusters information into fragments. This process is followed by the allocation phase which distributes, and if necessary, replicates the generated fragments among the nodes of a computer network. The use of data fragmentation to improve performance is not new and commonly appears in file design and optimization literature[9].

There is an emerging need for efficient support of databases consisting of very large amounts of data that are created and used by applications at different physical locations. Examples of application areas include telecom databases, scientific databases on grids, distributed data warehouses, and large distributed enterprise databases. In many of these application areas the delay from accessing a remote database is still significant enough to make necessary the use of distributed databases employing fragmentation and replication[23].

The strategy partition the database into disjoint fragments ,with each fragment assigned to on site .If data items are located at the site where they are used most frequently ,locality of reference is high .As there is no replication ,storage cost are low ;similarly reliability and availability are low, although they are higher than in a centralized case ,as the failure of a site result in loss of only that site's data. Performance should be good and communication cost low if the distribution is designed properly .In fragmentation data is stored close to where it is most frequently used. In addition, data that is not needed by local applications is not stored. With fragments as the unit of distribution, a transaction can be divided into several sub queries that operate on fragment[8].This should increase the degree of concurrency, or parallelism, in the system, thereby allowing transaction that can do so safely to execute in parallel. Fragmentation cannot be carried out haphazardly there are some rules that must be followed during fragmentation .The first one is completeness, if the relation R is decompose into fragments $R_1, R_2, \dots, R_n$ each data item that can be found in R must appear in at least one fragment .This rule is necessary to ensure that there is no loss of data during fragmentation .The another rule that must be followed during fragmentation is disjoint ness ,If a data item d_i appears in fragment R_i , then it should not appear in any other fragment .Vertical fragmentation is the exception of this rule ,where primary key attributes must be repeated to allow reconstruction .This rule ensure minimal data redundancy. The another rule that must be applied during fragmentation is reconstruction, it must be possible to define a relational operation that will reconstruct the relation R from the fragment .This rule ensure that functional dependencies are preserved. In the case of horizontal fragmentation, a data item is a tuple; for vertical fragmentation a data item is an attribute. There are mainly three type of fragmentation called horizontal, vertical, mixed[8]. The first is horizontal fragmentation ,in this type of fragmentation it is the subset of tuple .Horizontal fragmentation groups together the tuple in a relation that are collectively used by the important transaction .A horizontal fragment is produced by specifying a predicate that perform a restriction on the tuple in the relation .It is defined using the selection operation of the relational algebra .The selection operation groups together tuples that have some common property ;for example the tuples are all used by the same application or at the same site. In the vertical fragmentation it group together the attributes in a

relation that are used jointly by the important transaction. The vertical fragmentation is the subset of attributes, it is defined using the projection operation of the relational algebra. The advantage of vertical fragmentation is that the fragments can be stored at the site that need them in addition the performance is improved, as the fragment is smaller than the original base relation. Horizontal fragmentation splits the relation by assigning each tuple of r to one or more fragments. Vertical fragmentation splits the relation by decomposing the schema R of relation r . In horizontal fragmentation, a relation r is partitioned into a numbers of subset, $r_1, r_2, \dots, r_n$. Each tuple of relation r must belong to at least one of the fragments, so the original relation can be reconstructed, if needed. Horizontal fragmentation is usually used to keep tuple at the site where they are used to most, to minimize data transfer. To illustrate vertical fragmentation, consider a university database with a relation `employee_info` that store, for each employee, `employee_id`, `name`, `designation` and `salary`. For privacy reason, this relation may be fragmented into a relation `employee_private_info` containing `employee_id` and `salary`, and another relation `employee_public_info` containing attributes `employee_id`, `name` and `designation`. These may be store at different sites, again for security reason. Two type of fragmentation can be applied to single schema; the fragment obtained by horizontal fragmenting a relation can be further partitioned vertically. Fragment can also be replicated. In general, a fragment can be replicated, replies of fragments can be fragmented[8].

In distributed databases, the communication costs can be reduced by partitioning database tables horizontally into fragments, and allocating these fragments to the sites where they are most frequently accessed. The aim is to make most data accesses local, and avoid remote reads and writes. The read cost can be further reduced by the replication of fragments when beneficial. Obviously, important challenges in fragmentation and replication are how to fragment, when to replicate fragments, and how to allocate the (replicated) fragments[23].

Fragmentation of data indicates partition of database into number of small independent parts called fragments. Accessing of data from fragments introduce partial data access and an environment of working with table views. It is a step towards selection of data items using fine grained rather than coarse grained approach[45].

The sole purpose of different data distribution methods is to achieve overall distributed performance by:

- Dividing the workforce load into fragments and maintain easy data availability to them without wait or delay.
- Modular approach to ensure fast execution of subqueries.
- Allowing further network expansion without complexity.
- Controlling usage of storage space.
- Ensure easy data sites maintenance[45].

Some of the hindrances in the distribution of data in distributed system faced are:

- Network Latency

Delay during query response, replica propagation delay, and communication delay are common network hindrances who discard the effectiveness of data distribution approaches.

- Resources Availability:

Resources like power during regular usage, insufficient bandwidth to transfer and communication with others is required to smooth line the distributed database operations. efficiency in these resources leads to implementation hurdle.

- Disconnection

Frequent disconnection of network during distributed operations is a hindrance to achieve performance. To perform operations trusted service connection with the network is required[45].

2. History

Work on distributed Resource Description Framework (RDF) is performed to manage the growing massive RDF. To utilize this large volume RDF is partition into small parts called fragments and further approach the same for allocation in the distributed database environment. usually, focus is given to reduce the communication cost during the query processing tasks. It also ensures to maintain data integrity and approximation ratio due to frequent access patterns from outside. It is also focus on balancing and allocation of fragments into different sites [34].

A heuristic approach for fragmentation is proposed to reduce transmission cost (TC) of queries in distributed environment. at initial stage fragmentation is based on cost effective model in context of relational model and at later stage based on DDBS design. There are different replication based allocation scenario i.e. mixed replication based data allocation scenario (MAS), full replication based data allocation scenario (FAS), and nonreplication data allocation scenario (NAS)[3].

Modified Bond Energy Algorithm (BEA), it is a hierarchical process to make fragments vertically and allocate the fragments into geographical sites across the network. This algorithm use affinity of attributes and is helpful to generate cluster of attributes, to calculate cluster allocation cost and also decide about their appropriate sites for allocation. Here attributes accessed collectively by the same query are placed into one fragment [35].

A study was to review and compare the existing algorithms in design perspective with a view to identify their strength and weakness. This is just to present an affective design for the distribution of data fragments on the distributed environment [18].

A nonredundant dynamic fragment allocation technique is proposed and is based on the changing access pattern at different sites with a view to improve the performance. Here fragments reallocation is depend on access made on each fragment data volumes based on defined time constraint and threshold value. This proposed technique change the reallocation strategy by modifying the read and write data

volume factor and introduced threshold time volume and Distance Constraints Algorithm. Write data volume is considered for the reallocation process when more than one sites approach for the fragments. This ensures the overall improvement of distributed system performance [28, 45].

Primary concern of distributed database system design is to making fragmentation of the relations in case of relational database or classes in case of object oriented databases, allocation and replication of the fragments in different sites of the distributed system, and local optimization in each site.

Fragmentation is a design technique to divide a single relation or class of a database into two or more partitions such that the combination of the partitions provides the original database without any loss of information this reduces the amount of irrelevant data accessed by the applications of the database, thus reducing the number of disk accesses. Fragmentation can be horizontal, vertical or mixed/hybrid[9].

A hybrid optimized model using information on the type and frequency of queries for fragmentation of data horizontally and vertically and is based on supervised machine learning approach to produce non overlapping fragments. These fragments are maintained by archiving process rather than deletion operation on them. These fragments are used to facilitate searching operations based on index so that database tables are partition horizontally and vertically [32].

Two algorithms Modify Create Read Update Delete (MCRUD) and Matrix based Fragmentation (MMF) for efficient partitioning of large databases without query statistics. It shows that earlier approaches of partitioning were based on type and frequency of the queries called observed or experimental data. Here it is also indicated that earlier partitioning approach were not suitable because at the initial stage of the design of distributed database query statistics are not available. In his paper, an optimal fragmentation technique is proposed to partition global relations of a distributed database when there is no data access statistics and no query execution frequencies are available. When data access statistics and query execution frequencies are not available at the initial stage then MMF is responsible to partition relation in the distributed database. MCRUD is responsible to take fragmentation decision without using empirical data [27].

Work on different replication strategies in MANET, mobile database, distributed database, and cellular network etc is highlighted. It discuss about replication protocols as ROWA, ROWA-A and Quorum Based Protocol. All are replica control protocol, ROWA is responsible to fetch the read request values to the nearest site from the occurrence of request location and replicate the changes to all the sites. An alternative approach is in the form of ROWA-Available and is same as ROWA in the case of read operations but replicate the changes only to all available replica copies and do not bother about any replication failure. ROWA-A is responsible for maintaining the availability of data but do not compromised with the correctness of data. In case of failure users are working with stale value of data. It shows incorrect or out of date copy of replica. Quorum based replica is to update the subset of replicas rather than replicate the changes as a whole and helpful to maintain consistency of data [36].

An integrated approach is proposed for DDBMS namely data fragmentation, network sites clustering and allocation of fragments. This work is responsible to improvise problems in the form of; fragmentation,

redundancy in data allocation and redistribution problem due to complexity, to maintain data availability and consistency issues [27].

It is also highlighted, to maintain inconsistency issues faced by the mobile users during the access of database in his mobility from any of the activity center. Here a 5-cube structure with nearest neighbors propagation distribution protocol is proposed to make useful distributed database system for the mobile users. It ensures consistent data to all mobile users/sites by dynamically replicates the changes to all its adjoining sites from the transactional node [44].

A new dynamic deallocation approach for a given fragment as Update Matrix (UM) and Distance Cost Matrix (DM) is proposed. It works on the basis of changing data database system. It was assumed that fragments are allocated on network site is based on applied frequency value of the database data items. Reallocation of data fragments on the remote sites is planned based on communication and update cost value. Each fragment is having update cost value. Fragment having maximum update cost value is considered for reallocation and chosen candidate site to store fragments to minimize the communication cost. UM is defined as the value getting after issuing of update query at a particular site for the manipulated fragment. In this approach when same query is applied at more than one site, then queries can be treated different to each other and having different frequency value [1].

An algorithm called Simulates Annealing with Genetic Algorithm (SAGA) is used for optimal allocation of fragment in distributed environment. Here, allocation of data is depends on access patterns for fragments and focused on reducing the allocation cost during movement of data fragment from one site to another [2].

A decentralized approach for dynamic table fragmentation and allocation in distributed database systems is proposed. It is based on observation and monitoring of the sites access patterns to tables which reforms fragmentation, replication, and reallocation based on recent access history aiming at maximizing the number of local accesses compared to accesses from remote sites [26].

A new technique called Attribute Level Precedence (ALP) to partition global schema/database relations at initial and later stage in case of nonavailability of data access statistics and query execution frequencies. ALP technique is capable to take advance decision for fragmentation at the initial stage (i.e. knowledge gathered during requirement analysis phase) without empirical data statistics. ALP is a table responsible to fragment a relation horizontally based on the importance of an attribute in a network site [35, 45].

The problem of fragmenting tables so that data is accessed locally has been studied before. It is also related to some of the research in distributed file systems [20].

One important difference between distributed file systems and distributed database systems is the typical granularity of data under consideration (files vs. tuples) and the need for a fragmentation attribute that can be used for partitioning in distributed database systems.

Fragmentation is tightly coupled with fragment allocation. There are methods that do only fragmentation [5, 37, 39, 49, 50] and methods that do only allocation of predefined fragments [6, 7, 12, 15, 19, 29, 46]. Some methods also exist that integrate both tasks [14, 16, 24, 25, 38, 40, 43]. Replication, however, is typically done as a separate task [10, 13, 22, 30, 31, 48], although some methods, take an integral view of fragmentation, allocation and replication [16, 40, 43]. Dynamic replication algorithms [10, 22, 30, 31, 48] can optimize for different measures, but we believe that refragmentation and reallocation must be considered as alternatives to replication. In DYFRAM we choose among all these options when optimizing for communication costs. that replication scheme is somewhat similar to that of DIBAS [16], but DYFRAM also allows remote reads and writes to the master replica, whereas DIBAS always uses replication for reads and do not allow remote writes to the master replica. This operation shipping is important when analyses [13] of replication vs.

remote reads and writes conclude that the replication costs in some cases may be higher than the gain from local data access. A key difference between DIBAS and DYFRAM is that DIBAS is a static method where replication is based on offline analysis of database accesses, while DYFRAM is dynamic and does replication online as the workload changes.

Another important categorization of fragmentation, allocation and replication methods is whether they are static or dynamic. Static methods analyze and optimize for an expected database workload. This workload is typically a set of database queries gathered from the live system, but it could also include inserts and updates. Some methods also use more particular information on the data in addition to the query set [39]. This information has to be provided by the user, and is not available in a fully automated system. A form of static method is the design advisor [50] which suggests possible actions to a database administrator.

The static methods are used at major database reconfigurations. Some approaches, such as evolutionary algorithms for fragment allocation [6, 15], lend themselves easily to the static setting.

Static methods look at a set of queries or operations. It can be argued that the workload should be viewed as a sequence of operations, not as a set [4]. Dynamic methods continuously monitor the database and adapt to the workload as it is at the moment and are thus viewing a sequence of operations. Dynamic methods are part of the trend towards fully automatic tuning [47], which has become a popular research direction. Recently, work has appeared aiming at integrating vertical and physical partitioning while also taking other physical design features like indices and materialized views into consideration [5]. Adaptive indexing [4, 11] aims to create indices dynamically when the costs can be amortized over a long sequence of read operations, and to drop them if there is a long sequence of write operations that would suffer from having to update both base tables and indices. In adaptive data placement, the focus has either been on load balancing by data balancing [14, 24], or on query analysis [25].

Closest to modern approaches may be the work of Brunstrom et al. [12], which studied dynamic data allocation in a system with changing workloads. Their approach is based on predefined fragments that are periodically considered for reallocation based on the number of accesses to each fragment.

A third aspect is how the methods deal with distribution. The method can either be centralized, which means that a central site gathers information and decides on the fragmentation, allocation or replication, or it can be decentralized, delegating the decisions to each site. Some methods use a weak form of decentralization where sites are organized in groups, and each group chooses a coordinator site that is charged with making decisions for the whole group [22, 30].

In DYFRAM, fragmentation, allocation and replication decisions are fully decentralized. Each site decides over its own fragments, and decisions are made on the fly based on current operations and recent history of local reads and writes.

Mariposa [40, 41] is a notable exception to the traditional, manually fragmented systems. It provides refragmentation, reallocation and replication based on a bidding protocol. A Mariposa site will sell its data to the highest bidder in a bidding process where sites may buy data to execute queries locally or pay less to access it remotely with larger access times, optimizing for queries that have the budget to buy the most data. A DYFRAM site will split off, reallocate or replicate a fragment if it optimizes access to this fragment, seen from the fragment's viewpoint. This is performed also during query execution, not only as part of query planning, as is the case in Mariposa[23].

Finally there are general methods for fragmentation in figure (1), but it figure just shown Horizontally, vertically and sum of general mixed methods. For dynamic and other ways, usually use mixed methods with combination of dynamic or other algorithm for change fragments depend on time or requests.

3. Horizontal Fragmentation

Horizontal fragmentation (HF) allows a relation or class to be partitioned into disjoint tuples or instances. Intuition behind horizontal fragmentation is that Every site should hold all information that is used to query at the site and the information at the site should be fragmented so the queries of the site run faster[9]. Horizontal fragmentation is defined as selection operation, $\sigma_p(R)$.

Computing horizontal fragmentation (idea)

- a. Compute the frequency of the individual queries of the site q_1 ,
- b. Rewrite the queries of the site in the conjunctive normal form (disjunction of conjunctions); the conjunctions are called minterms.
- c. Compute the selectivity of the minterms
- d. Find the minimal and complete set of minterms (predicates)
 - The set of predicates is complete if and only if any two tuples in the same fragment are referenced with the same probability by any application.
 - The set of predicates is minimal if and only if there is at least one query that accesses the fragment

e. There is an algorithm how to find these fragments algorithmically (the algorithm CON MIN and PHORIZONTAL) DDB[9].

An example on horizontal fragmentation is the PROJ table shown in figures(2,3,4).

Horizontal fragmentation of PROJ relation into:

PROJ1: projects with budgets less than 200, 000.

PROJ2: projects with budgets greater than or equal to 200, 000.

4. Vertical Fragmentation

Combination of horizontal and vertical fragmentations is mixed or hybrid fragmentations (MF). In this type of fragmentation scheme, the table is divided into arbitrary blocks, based on the needed requirements. Each fragmentation can be allocated on to a specific site. This type of fragmentation is the most complex one, which needs more management. In most cases simple horizontal or vertical fragmentation of a DB schema will not be sufficient to satisfy the requirements of the applications[9,21,42].

Mixed fragmentation (hybrid fragmentation) Consists of a horizontal fragment followed by a vertical fragmentation, or a vertical fragmentation followed by a horizontal fragmentation. Mixed Fragmentation is defined using the selection and projection operations of relational algebra:

$$\Pi_p(_A1, \dots, An(R))$$
$$\Pi_A1, \dots, An(_p(R))$$

A fragment of a relation is a relation itself. Fragments can be further fragmented

$$\text{Projects1} = \Pi \text{ PNo, PName, Location } (\text{Projects})$$
$$\text{Projects2} = \Pi \text{ PNo, Budget } (\text{Projects})$$
$$\text{Projects1:1} = \sigma \text{ Location='Rom'} (\text{Projects1})$$
$$\text{Projects2:1} = \sigma \text{ Location='Munich'} (\text{Projects1})$$
$$\text{Projects3:1} = \sigma \text{ Location='Paris'} (\text{Projects1})$$
$$\text{Projects} = (\text{Projects1:1} \bowtie \text{Projects1:2} \bowtie \text{Projects1:3}) \bowtie \text{Projects2}$$

5. Hybrid Fragmentation

Combination of horizontal and vertical fragmentations is mixed or hybrid fragmentations (MF). In this type of fragmentation scheme, the table is divided into arbitrary blocks, based on the needed requirements. Each fragmentation can be allocated on to a specific site. This type of fragmentation is the most complex one, which needs more management. In most cases simple horizontal or vertical fragmentation of a DB schema will not be sufficient to satisfy the requirements of the applications[9,21,42].

Mixed fragmentation (hybrid fragmentation) Consists of a horizontal fragment followed by a vertical fragmentation, or a vertical fragmentation followed by a horizontal fragmentation. Mixed Fragmentation is defined using the selection and projection operations of relational algebra:

$$\Pi_p(_A1, \dots, An(R))$$

$$\Pi_A1, \dots, An(_p(R))$$

A fragment of a relation is a relation itself. Fragments can be further fragmented

$$\text{Projects1} = \Pi \text{ PNo, PName, Location (Projects)}$$

$$\text{Projects2} = \Pi \text{ PNo, Budget(Projects)}$$

$$\text{Projects1:1} = \sigma \text{ Location='Rom' (Projects1)}$$

$$\text{Projects2:1} = \sigma \text{ Location='Munich' (Projects1)}$$

$$\text{Projects3:1} = \sigma \text{ Location='Paris' (Projects1)}$$

$$\text{Projects} = (\text{Projects1:1} \Join \text{Projects1:2} \Join \text{Projects1:3}) \Join \text{Projects2}$$

6. Correctness Rules Of Fragmentation

a. Completeness: Decomposition of relation R into fragments R1, R2, . . . , Rn is complete iff each data item in R can also be found in some Ri.

b. Reconstruction: If relation R is decomposed into fragments R1, R2, . . . , Rn, then there should exist some relational operator ∇ that reconstructs R from its fragments, i.e., $R = R \nabla \dots \nabla Rn$

i. Union to combine horizontal fragments

ii. Join to combine vertical fragments.

c. Disjointness: If relation R is decomposed into fragments R1, R2, . . . , Rn and data item di appears in fragment Rj , then di should not appear in any other fragment Rk, k $\neq$ j (exception: primary key attribute for vertical fragmentation)

- i. For horizontal fragmentation, data item is a tuple
- ii. For vertical fragmentation, data item is an attribute[9].

7. Fragmentation On Unstructured Data

The world wide web (WWW) is often considered to be the world's largest database and the extensible Markup Language (XML) is then considered to provide its data model. There raises the question, how to obtain a suitable distribution design for XML documents.

Actually horizontal and vertical fragmentation techniques are generalised from the relational data model to XML.

Furthermore, splitting is introduced as a third kind of fragmentation.

In this concept, XML is described as a data model. Extended DTDs(Document Type Definitions) are used to define schemata. Equivalently, XML Schema is used, but extensions would be needed. Then it is considered to be the standard for XML documents as databases over such schemata. The queries are used with an extension of XML SQL. Equivalently, XQuery could be used, but again extensions would be needed in both cases[42].

8. Conclusion

One of the most critical aspects of distributed database design and management is fragmentation. If the fragmentation is done properly, we can expect to achieve better throughput from such systems[32].

Making proper fragmentation of the relations and allocation of the fragments is a major research area in distributed databases. Many techniques have been proposed by the researchers using empirical knowledge of data access and query frequencies. But proper fragmentation and allocation at the initial stage of a distributed database has not yet been addressed[27].

To design an effective distributed model, it is important to manage an appropriate methodology for data fragmentation and fragment allocation. Nevertheless, very little works address this problem in a distributed context; an optimization problem including the several interrelated problems like data fragmentation, allocation and local optimization. Each problem can be solved by using several different approaches[21].

In spite of several significant features of existing models, there are still key features that need to be built into subsequent improvements or studies. For instance, there is need to design the fragmentation method, using preferably machine learning techniques, so that it can produce only non overlapping fragments that can be archived rather than being constantly deleted . Also, when fragments increase, there may be need to build an index of fragments so as to facilitate searches[32]. Furthermore, we

recommend that future studies expand the classifier used in the study by including queries that are not only retrieval queries, for example, insertion, update and delete queries.

Declarations

Ethics approval and consent to participate, Consent for publication, Availability of data and materials, Competing interests

We wish to submit an original research article entitled “A Survey on Fragmentation in Distributed Database Systems” for consideration by Cluster Computing journal.

We confirm that this work is original and has not been published elsewhere, nor is it currently under consideration for publication elsewhere. We believe that this manuscript is appropriate for publication by Journal of Big Data, because its subject. We have no conflicts of interest to disclose.

Thank you for your consideration of this manuscript.

Sincerely,
Mohsen Taki

Funding, Authors' contributions, Acknowledgements

Not applicable

References

1. Abdalla HI. A New Data Re-Allocation Model for Distributed Database System. International Journal of Database Theory Application. 2012;5(2):45–60.
2. Abdalla HI, An Efficient Approach for Data Placement in Distributed Systems, Fifth FTRA International Conference on Multimedia and Ubiquitous Engineering, pp. 297–301, 2011.
3. Abdalla H, Artoli AM. Towards an Efficient Data Fragmentation, Allocation, and Clustering Approach in a Distributed Environment, Information, Vol 10, No.112, 2019.
4. Agrawal S, Chu E, Narasayya VR, Automatic physical design tuning: workload as a sequence In Proceedings of SIGMOD 2006, 2006.
5. Agrawal S, Narasayya V, Yang B, Integrating vertical and horizontal partitioning into automated physical database design. In Proceedings of SIGMOD, 2004.
6. Ahmad I, et al. Evolutionary algorithms for allocating data in distributed database systems. Distributed Parallel Databases. 2002;11(1):5–32.
7. Apers PMG. Data allocation in distributed database systems. ACM Trans Database Syst. 1988;13(3):263–304.

8. Bhardwaj Ankur singh, Amrinder kaur, Prabhdip singh, Brahmibir, Role of Fragmentation in Distributed database system, International Journal of Networking & Parallel Computing, Volume 1, Issue 1, September, 2012.
9. Bhuyar PR, Gawandw AD, Distributed Database. Fragmentation and allocation, Journal of Data Mining and Knowledge Discovery, ISSN: 2229–6662 & ISSN: 2229–6670, Volume 3, Issue 1, 2012.
10. Bonvin N, Papaioannou TG. K. Aberer. A, self-organized, fault-tolerant and scalable replication scheme for cloud storage. In Proceedings of SoCC '10, 2010.
11. Bruno N, Chaudhuri S. An online approach to physical design tuning. In Proceedings of ICDE, 2007.
12. Brunstrom A, Leutenegger ST, Simha R, Experimental evaluation of dynamic data allocation strategies in a distributed database with changing workloads. In Proceedings of CIKM '95, 1995.
13. Ciciani B, Dias D, Yu P. Analysis of replication in distributed database systems. IEEE Transactions on Knowledge and Data Engineering, 2(2):247–261, Jun 1990.
14. Copeland G, et al, Data placement in Bubba. In Proceedings of SIGMOD 1988, 1988.
15. Corcoran AL, Hale J, A genetic algorithm for fragment allocation in a distributed database system In Proceedings of SAC'94, 1994.
16. Didriksen T, Galindo-Legaria CA, Dahle E, Database de-centralization - a practical approach. In Proceedings of VLDB 1995, 1995.
17. Elmasri Ramez, Navathe SB. Fundamentals of Database Systems, 1999.
18. Fuaad HA, Ibrahim AA, Majed A, Asem A, A Survey on Distributed Database Fragmentation, Allocation and Replication Algorithms, British Journal of Applied Science and Technology, 27(2): pp.1–12, 2018.
19. Furtado P, Experimental evidence on partitioning in parallel data warehouses. In Proceedings of DOLAP, 2004, 2004.
20. Gavish B, Sheng ORL. Dynamic file migration in distributed computer systems. Commun ACM. 1990;33(2):177–89.
21. Gurpreet Kaundal K, Sukhleen V, Sheveta, Review on Fragmentation in Distributed Database Environment, IOSR Journal of Engineering (IOSRJEN), 2278–8719 Vol. 04, 03, March. 2014.
22. Hara T, Madria SK. Data replication for improving data accessibility in ad hoc networks. IEEE Trans Mob Comput. 2006;5(11):1515–32.
23. Olav HJ, Nørvag NHRyeng, K. DYFRAM: Dynamic Fragmentation and Replica Management in Distributed Database Systems, Distributed and Parallel Databases manuscript, 2011.
24. Hua KA, Lee C, An adaptive data placement scheme for parallel database computer systems. In Proceedings of VLDB 1990, 1990.
25. Ivanova M, Kersten ML, Nes N, Adaptive segmentation for scientific databases. In Proceedings of ICDE 2008, 2008.
26. Khan SI, Efficient Partitioning of Large Databases without Query Statistics, Database System Journal, pp. 34–53, 2016.-1-15 Hauglid JO, Ryeng NH, Norvag K. DYFRAM: Dynamic Fragmentation

- and Replica Management in Distributed Database Systems, Distributed Parallel Databases, pp 157–185, 2010.
27. Khan SI, Hoque AS, L.. A New Technique for Database Fragmentation in Distributed System. International Journal of Computer Application. 2010;5(9):20–4.
 28. Lwin NKZ, Naing TM. Non-Redundant Dynamic Fragment Allocation with Horizontal Partition in Distributed Database System. Bangkok: ICIIBMS; 2018. pp. 300–5.
 29. Menon S. Allocating fragments in distributed databases. IEEE Trans Parallel Distrib Syst. 2005;16(7):577–85.
 30. Mondal A, Madria SK, Kitsuregawa M, CADRE: A collaborative replica allocation and deallocation approach for mobile-p2p networks. In Proceedings of IDEAS 2006, 2006.
 31. Mondal A, Yadav K, Madria SK, EcoBroker: An economic incentive-based brokerage model for efficiently handling multiple-item queries to improve data availability via replication in mobile-p2p networks. In Proceedings of DNIS 2010, 2010.
 32. Oriji IB, Ejiofor IC. A Hybrid Model for Data Fragmentation in Distributed System. International Research Journal of Computer Science Issue. 2018;04(Volume 5):186–92.
 33. Özsu, Tamer. Valduriez Patrick, Distributed Data-base Management Systems, 1998.
 34. Peng P, Lei Zou, Lei chen, Dongyan Zhao, Adaptive Distributed RDF Graph Fragmentation and Allocation based on Query Workload. IEEE Trans Knowl Data Eng. April 2019;31(4):670–85.
 35. Rahimi H, Parand F. Hierarchical Simultaneous Vertical Fragmentation and Allocation using Modified Bond Energy Algorithm in distributed databases, Applied Computing and Informatics (14), pp.127–133, 2018.
 36. Rane D, Dhore MP. Overview of Data Replication Strategies in Various Mobile Environment, IOSR Journal of Computer Engineering, pp. 01–06, 2016.
 37. Rao J, et al, Automating physical database design in a parallel database. In Proceedings of SIGMOD2002, 2002.
 38. Sacca D, Wiederhold G. Database partitioning in a cluster of processors. ACM Trans Database Syst. 1985;10(1):29–56.
 39. Shin D-G, Irani KB. Fragmenting relations horizontally using a knowledge-based approach. IEEE Trans Software Eng. 1991;17(9):872–83.
 40. Sidell J, Aoki PM, Sah A, Staelin C, Stonebraker M, Yu A, Data replication in mariposa. In Proceedings of ICDE 1996, 1996.
 41. Stonebraker M, et al. Mariposa: A wide-area distributed database system. VLDB J. 1996;5(1):48–63.
 42. Suganya A, Kalaiselvi R. Efficient Fragmentation and Allocation in Distributed Databases, International Journal of Engineering Research & Technology (IJERT), Vol. 2 Issue 1, January 2013.
 43. Tamhankar A, Ram S. Database fragmentation and allocation: an integrated methodology and case Study Systems, Man and Cybernetics, Part A, IEEE Transactions on, 28(3):288–305, May 1998.

44. Tarun S. A Reputation Replica Propagation Strategy for Mobile Users in Mobile Distributed Database System, International Journal of Grid and Distributed Computing, Vol. 5, No. 4, 2012.
45. Tarun Sashi BR, Singh K, Sukhpreet. A Review on Fragmentation, Allocation and Replication in Distributed Database Systems, International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Amity University Dubai, UAE,2019.
46. Ulus T, Uysal M. Heuristic approach to dynamic data allocation in distributed database systems Pakistan. Journal of Information Technology. 2003;2(3):231–9.
47. Weikum G, et al. The COMFORT automatic tuning project, invited project review. Information Systems. 1994;19(5):381–432.
48. Wolfson O, Jajodia S, Distributed algorithms for dynamic replication of data. In Proceedings of PODS'92, New York, NY, USA, 1992. ACM.
49. Wong E, Katz RH. Distributing a database for parallelism. SIGMOD Rec. 1983;13(4):23–9.
50. Zilio DC, et al, DB2 design advisor: integrated automatic physical database design. In Proceedings of VLDB 2004, 2004.

Figures

(a) Unfragmented

(b) Horizontal

(c) Vertical

(d) Row-biased

(e) Column-biased

(f) Unbiased

Figure 1

fragmentation methods

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Development	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

Figure 2

PROJ Table[9]

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Development	135000	New York

Figure 3

PROJ_1 Table[9]

PNO	PNAME	BUDGET	LOC
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

Figure 4

PROJ_2 Table[9]

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Development	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

Figure 5

PROJ Table[9]

PNO	PNAME	LOC
P1	Instrumentation	Montreal
P2	Database Development	New York
P3	CAD/CAM	New York
P4	Maintenance	Paris
P5	CAD/CAM	Boston

Figure 6

PROJ_1 Table[9]

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000
P5	500000

Figure 7

PROJ_2 Table[9]