A Hybrid Approach for Efficient Database Fragmentation in Distributed Systems

Muhammad Shoaib[1], Muhammad Usman Younus[2,3*], Kalsoom Safdar[1,4*], Abdul Basit Dogar[5]
[1]Department of Computer Science & IT, University of Jhang, Jhang, Pakistan.
[2]Department of Computer Science & IT, Baba Guru Nanak University, Nankana Sahib, Pakistan.
[3]Ecole Doctorale Matheematiques, Informatique, Telecommunication de Toulouse, University of Toulouse (III), Paul Sabatier, Toulouse, France.
[4]Faculty of Electronic Engineering and Technology, University Malaysia Perlis, 02600 Arau, Perlis, Malaysia.
[5]Department of Informatics and Systems, School of Systems and Technology, University of Management and Technology Lahore, Pakistan.
***Correspondence**: usman1644@gmail.com, kalsoomsafdar11@gmail.com

Citation | Shoaib. M, Younus. M. U, Safdar. K Dogar. A. B, "A Hybrid Approach for Efficient Database Fragmentation in Distributed Systems", IJIST, Vol. 08 Issue. 01 pp 162-178, January 2026.
Received | December 14, 2025 **Revised |** January 10, 2026 **Accepted |** January 14, 2026 **Published |** January 20, 2026.

Distributed Database Management Systems (DDBMSs) require effective data allocation techniques to ensure scalability, availability, and performance in distributed systems of geographically decentralized nodes. The traditional fragmentation procedures fail to consider the dynamic access frequency and attribute locality, leading to increased query costs and storage overhead. In this paper, we develop a new Derived Horizontal Fragmentation method, called MCRUD, that aims to optimise fragment placement by incorporating attribute locality and access frequency into the fragmentation process. The proposed model was tested and applied in a simulated distributed database environment, and its performance was compared with that of conventional horizontal fragmentation techniques. The metrics used to measure performance included query response time, storage overhead, data transfer rate, and computation time. Experimental findings show that the MCRUD method significantly improves system performance by reducing data retrieval time and data storage redundancy, without affecting data consistency or availability. The results show that the proposed fragmentation strategy improves the scalability and load balancing in the distributed database systems. MCRUD model offers a powerful data distribution mechanism that can be applied to a large-scale distributed environment and optimised to adapt to workload and modify database optimisation in future studies.

Keywords: Horizontal Fragmentation, Distributed Database, Query Optimisation, Attribute Locality, MCRUD.

Introduction:

Distributed Database Management Systems (DDBMSs) are becoming increasingly useful in modern organizations that process large volumes of data across geographically decentralized nodes. Though distribution enhances scalability and availability, it poses major challenges for data allocation and query efficiency. Misplaced data increases communication costs, storage redundancy, and query latency, especially under dynamic load [1][2][3][4]. Hence, effective fragmentation and allocation plans are necessary to achieve the best in distributed settings.

One of the most commonly used methods for distributing relational data is horizontal fragmentation, in which tuples are partitioned according to predefined predicates, allowing parallel query processing [5]. Nevertheless, conventional horizontal fragmentation methods are primarily based on static conditions and do not adequately account for access frequency and attribute locality [6]. As a result, often-used attributes can be located on remote nodes, leading to increased network overhead and poor system performance. These limitations become more evident in real-world distributed database applications amid the increasing prevalence of data-intensive applications.

Some researchers have sought to address fragmentation by using workload-aware, adaptive approaches [7]. Privacy-preserving fragmentation schemes have been proposed to improve secure data splitting, and cost-based allocation models are used to reduce query execution time and enhance load balancing. Dynamically adaptable to changes in workload has also been introduced using reinforcement learning based and hybrid fragmentation techniques [8][9]. Regardless of these developments, the majority of current techniques are primarily concerned with minimizing query costs or security, and rarely with attribute locality and access frequency when placing fragments. As a result, distributed systems continue to exhibit redundant data transfer and inefficient resource use.

Contributions of the paper:

To provide a Derived Horizontal Fragmentation approach based on Modified CRUD (MCRUD), which uses Attribute Local Precedence (ALP) to improve fragment placement.
Using the suggested model in a financing structure for a real client and analyzing it using a broader range of performance metrics.
To show some improvements in query speed, storage overhead, and flexibility by contrasting the new fragmentation and allocation algorithms.

This work combines dispersion and allocation approaches to accommodate distributed database systems. We further enhance this by integrating workload-based fragmentation methods with cost-based optimization, which demonstrates superior gains in query turnaround time, throughput, and scalability compared to current approaches.

There are various sections to this study. The traditional database fragmentation technique, called horizontal fragmentation, is covered in the first section. Derived Horizontal Fragmentation is reviewed and applied in the second section to overcome the shortcomings of traditional approaches. Afterward, we discuss the implementation of the derived fragmentation technique in the customer financing system in Section III. As DBMS modeling developed, a horizontal fragmentation strategy emerged. Section IV presents the pros and cons of the applied fragmentation strategy. In Section V, the conclusion is finally covered.

Related Work:

Database Management Systems can be effectively operated only if they are structured with appropriate fragmentation and allocation processes. An extensive dataset can be divided into multiple fragments in horizontal, vertical, or combined orientations. Hybrid approaches also serve their purpose when paired with developers' repeated testimonials [10]. This paper discusses the different types of fragmentation and then presents an improved model of the technique for greater accuracy, efficiency, reliability, and cost-effectiveness. The assembly is designed to perform a set of steps, such as (i) data fragmentation, which distributes the entire dataset into smaller subsets without losing any data; (ii) fragment allocation, which is the placement of each data fragment into a known location for its easy retrieval; (iii) placement of fragment replicas, which is the separate allocation of multiple copies of fragments prepared [11]. These copies should not exceed a limit to allow easy access to each copy. Too many replicas can slow down software systems, while only a few copies might result in the loss of important information; (iv) concurrency manager, which is the core step offering synchronization of different components into a joint infrastructure; (v) processing of data as a query so that each piece of data remains safe and sound even after several successful attempts to access the data.

Initial research on distributed database design has focused on horizontal fragmentation, which partitions a global relation into subsets of tuples according to a predicate specification [12]. The method is best suited for improving response time by minimizing the transfer of irrelevant data during query execution. Vertical fragmentation, by contrast, partitions attributes into smaller relations according to access patterns [13]. Several publications have noted that vertical fragmentation is more effective when individual attribute subsets are accessed [14]. The flexibility of hybrid fragmentation, or the combination of horizontal and vertical strategies, allows the structure to adapt to application requirements. It is noted that hybrid fragmentation may be more effective at distributing loads and optimizing system use, especially in a heterogeneous computing environment [15].

The other critical element, which has been extensively covered in previous literature, is the control of concurrency mechanisms needed in a distributed environment. These schemes facilitate synchronized access to fragments when multiple users or applications access them simultaneously [16]. Timestamp ordering, distributed locking protocols, and optimistic concurrency control have been important for ensuring the reliable operation of distributed DBMSs. The concurrency manager is the most important mechanism for ensuring data integrity, minimizing conflicts, and aligning transactions across sites, especially in complex fragmentation schemes.

Recent studies have demonstrated the effectiveness and optimality of distributed query processing for coordinating the search of fragmented data across multiple locations [11]. Some query optimization methods are reported to significantly reduce communication overhead in distributed systems, including semi-join elimination, query rewriting, and cost-based optimization. The fragmentation of these systems makes them consistent and accurate in handling multiple access requests by enabling data queries across fragmented structures.

On these platforms, scholars have developed improved frameworks that combine fragmentation, allocation, replication, and concurrency to enhance overall performance [17]. These mixed or hybrid models provide coordinated solutions that optimize cost, query response time, reliability, and load distribution. They also help reduce redundant data exchanges and improve the availability of high-availability systems.

In this paper, a thorough analysis of the novel fragmentation method, the Derived Horizontal Fragmentation Technique, will be conducted. This method has been implemented to define the (owner, member) relationship among the different distributed fragments. This technology has been developed to address fragmentation across horizontal, vertical, and hybrid approaches.

Materials and Methods:

In this paper, an MCRUD-based Derived Horizontal Fragmentation model is proposed for a distributed database system. The methodology comprises four phases: data analysis, fragment design, fragment allocation, and performance evaluation. A simulated distributed database architecture used to implement the proposed model comprised several database nodes connected via a network layer. Each node stores a subset of the database fragments, and queries are processed locally to reduce the number of remote data accesses. The experiment was conducted in a relational database, using structured records of financial transactions as the main dataset. The case study was done using a real client financing database. The data includes customer details, transactions, loan details, and payment plans. The dataset contains both frequently and rarely accessed attributes, so it is appropriate for determining attribute locality and access frequency.

The dataset was analyzed to identify before fragmentation:
Attribute access frequency
Query patterns
Distribution of transactions among nodes.

The Modified Create-Read-Update-Delete (MCRUD) model is a variation of the CRUD analysis model that has replaced the Attribute Local Precedence (ALP). The model uses the frequency of attribute access in query workloads to rank attributes by importance.

The steps followed in the fragmentation process are:
Examine query workload and identify the frequency of attribute usage.
Allot priority weights with ALP.
Create derived horizontal fragments depending on access conditions.
Fragment allocation to distributed nodes with minimum remote access.

Officially, horizontal fragmentation is given as:

$F_i = \sigma_{P_i}(R)$

where (P_i) is the predicate condition based on attribute locality and frequency of access.

Strategic fragmentation occurs through the allocation of fragments via the fragmentation strategy.

Fragment Allocation Strategy:

Workload proximity was used to distribute the fragments across nodes. The frequently accessed fragments were stored close to the nodes that generated the most queries. The method minimizes communication overhead and improves query execution time.

Metrics of Performance Evaluation:

The metrics that were used to evaluate the proposed method were as follows:
Query Response Time
Data Transfer Cost
Storage Overhead
Throughput
System Scalability

To achieve reproducibility, the experiment was performed in a fixed workflow:
Insert the data into the centralized database.
Use the conventional horizontal fragmentation.
Implement the suggested MCRUD-derived fragmentation.
Introduce the same load of queries.
Record performance metrics
Compare findings between models.

In Fig. 1, a distributed database design is shown to show horizontal data fragmentation of six networked locations to provide local query processing and less inter-location communication overhead

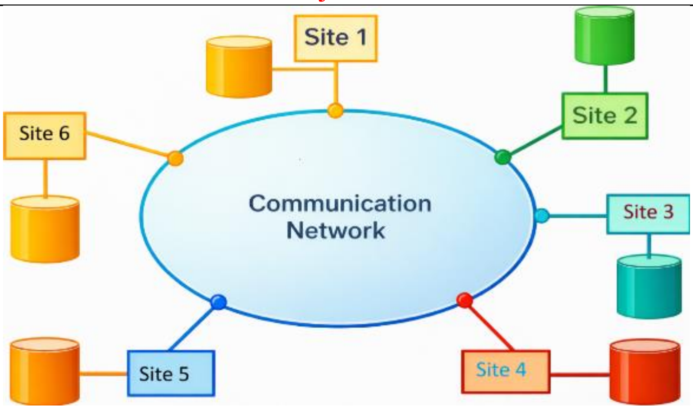

Figure 1. Distributed Database Management

Vertical Fragmentation Technique in Dbms:

The global fragment's available characteristics will be partitioned into several subsets during vertical fragmentation [18]. Decomposition is the most basic type of vertical fragmentation, in which each fragment may have a unique row input to guarantee and facilitate rebuilding via a join operation. Stated differently, this kind of fragmentation can be used to partition data into a few tables, with related attributes stored in each. A subset of a relation's attributes makes up vertical fragmentation. Each fragment in vertical fragmentation should include the table's primary key field(s) to preserve consistency, as shown in Fig. 2. One way to protect sensitive information or significant transactions is through vertical fragmentation. A relation E is supposed to contain vertical fragments as E1, E2, E3, etc., each possessing the attributes of the original set E [19]. The following projection operation associated with relational algebra is used to represent vertical fragmentation:

$$\begin{bmatrix} E1 \\ E2 \\ E3 \\ . \\ . \\ . \\ En \end{bmatrix} = \begin{bmatrix} \sigma C1D \\ \sigma C2D \\ \sigma C3D \\ . \\ . \\ . \\ \sigma CnD \end{bmatrix} \qquad (1)$$

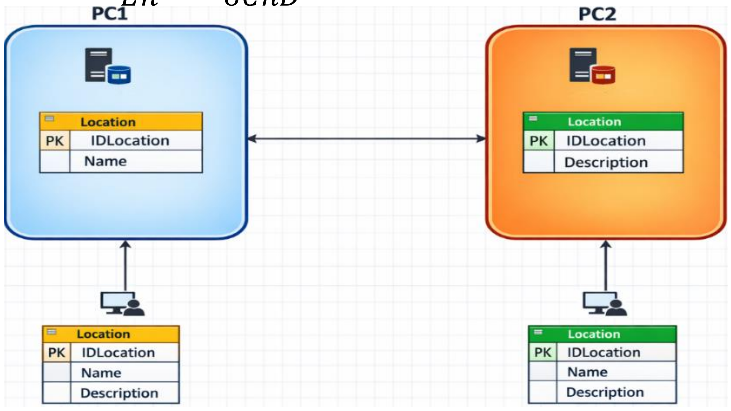

Figure 2. Horizontal vs. Vertical Fragmentation.

Hybrid of horizontal and vertical fragmentation techniques:

A horizontal fragment that is also vertically fragmented, or a vertical fragment that is also horizontally fragmented, makes up mixed or hybrid fragmentation. This is the most adaptable fragmentation method because it produces pieces with the least unnecessary information [20]. Because many applications and queries require more complex fragmentation than that, implementing a single fragmentation technique—horizontal or vertical—is insufficient in practice. This kind of fragmentation is achieved by first applying horizontal fragmentation, then performing vertical fragmentation on one or more previously generated horizontal fragments. Rebuilding the original table, however, is frequently a costly undertaking, as the rebuilding of hybrid fragmentation by FULL OUTER JOIN and UNION operations is not always an optimal approach. The following selection projection can represent hybrid fragmentation:

II _r (_B1, B2…., Bn(D)) (2)

The primary causes of relationship fragmentation are to improve data availability and dependability; increase the locality of reference for queries submitted to the database; minimize transmission costs between sites; balance storage capacities; and optimize system performance [21]. HF, VF, and MF's previous approaches share the issues that they base their fragmentation on the frequency of searches, the affinity of midterm predicates, or the attribute affinity matrix (AAM). These necessitate ample empirical data, which is typically unavailable at the beginning. To minimize complexity, most focus solely on the fragmentation problem, ignoring the allocation problem.

Illustration of 007 Benchmark System:

We have included the 007 Benchmark Database Model as an example to describe DBMS in detail. The process of assigning database fragments to locations in distributed networks is known as allocation. Once allocated, data can be kept as a single file or duplicated. Although fragment replication increases update costs, it improves read-only query performance and reliability. We introduce a novel method for horizontally partitioning relations in distributed databases. Without empirical data on query execution, this technique may make an appropriate fragmentation decision at the outset based on knowledge acquired during the requirements analysis phase [22]. Additionally, it can correctly allocate fragments across the DDBMS sites. The 007 Benchmark database, modified from, provides a consistent schema for performance evaluation, as shown in Fig. 3.

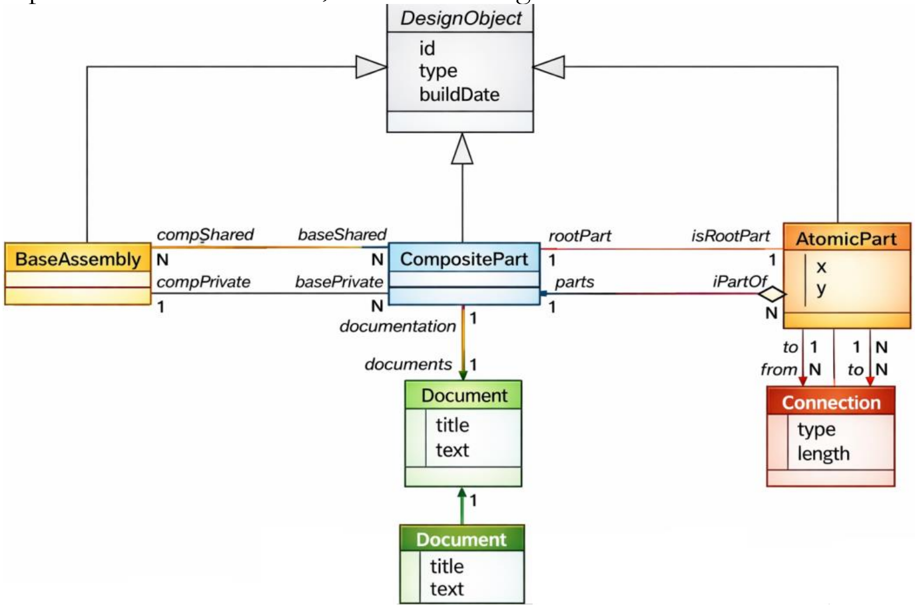

Figure 3. 007 Benchmark Database

A Proposed Derived Horizontal Model Implemented in Dbms:

The "owner and member" classification has historically served as the basis for most literary works in determining whether horizontal fragmentation is primary or derived. One of the most important factors to consider when building a distributed database is classification. In [23], the "owner-member" classification in the relational model was first introduced. The "owner and member" classification remains significant in the object model, although it is more complex than in its relational counterpart. This classification requires consideration of several object model difficulties, including the presence of complex objects, "part-of" relationships between classes, method calls, pseudo-classes (classes with no instances), and "a × b" relationships that lead to the sharing of versatile objects. The following section of the paper will describe the (owner, member) relationship, taking into account the quality and quantity of the data. This relationship can be interpreted in various ways to account for all query attributes.

We initially define the (owner, member) relation as follows to determine the owner and member classes. We suppose a set of pairs in the form (E, F) that make up the (owner, member) relation. The database schema requires that E and F be classes. An illustration of a relation is any pair (E, F) in the relation. An occurrence (E, F) indicates that class E (also known as the owner of F) will determine which class F will be chosen for derived horizontal fragmentation. Class E will be selected for the (owner, member) relation if, at the conclusion of its definition, there isn't another instance (B, E) in it—that is, if class E doesn't function as a "member" in any of the relation instances. Class E will be selected for the initial horizontal fragmentation process [22]. Denoting that class E will be selected for primary horizontal fragmentation even if no other class may be chosen for derived fragmentation according to E, it is also feasible to declare an occurrence of the type (E, null). The definition of the (owner, member) relation is also subject to certain limitations.

A pair of instances (E, B), (F, B), is not possible because it is not suitable to define derived horizontal fragmentation on a class B according to more than one primary class (E and F);

An occurrence of (E, E) is not possible because it is not recommended to define derived horizontal fragmentation on a class showing the same attributes as itself; since it is impossible to define derived horizontal fragmentation on another class B in accordance with more than one definition, it is also not conceivable to have appearances (E, B), (F, B) conforming to primary class (E and F).

The given database has been horizontally partitioned, and some operations, namely Transactions and Queries, are implemented on it to define (owner, member) relationships. All these activities were performed with some frequency, which facilitated their organisation, and these operations were observed for analysis. The analysis of one such Pi operation is given below:

Only one category or set named Y can be accessed via the Pi operation. We can evaluate the (owner, member) relationship as a (Y, null) instance. However, this situation may be rare and may not recur. This situation can be described as follows:

Example 1: selection of the Y from a fundamental Electronic Part where the build date can be taken as < 11/12/98, where we can define (owner, member) = (Electronic Part, null).

Example 2: selection of the Y class, Y build date from the Y in the Complex Part, where we can represent the following relationship: (owner, member) = (Complex Part, null).

Pi will navigate via a category pathway, which can be represented as A1->A2->…->An. In this example, we can take an (owner, member) relation that includes instances (Ai, Ai+1) for the category path, $1 \leq i \leq n-1$, for each of two categories (Ai, Ai+1). We do not take inheritance links into account, as they would degrade performance by producing redundant results. Fig. 4 shows the algorithm as we discussed above.

Proposed Model of the Fragmentation Based on Crud Matrix:

To overcome the drawbacks of classical methods, a new model was proposed based on the horizontal fragmentation approach. This method is based on the database's distribution by attribute order. ALP (Attribute Locality Precedence) is a system that assigns values to attributes based on their importance in determining the placement of database sites. A DBMS system is designed using the CRUD Matrix. This concept can be illustrated with the help of the following block diagram:

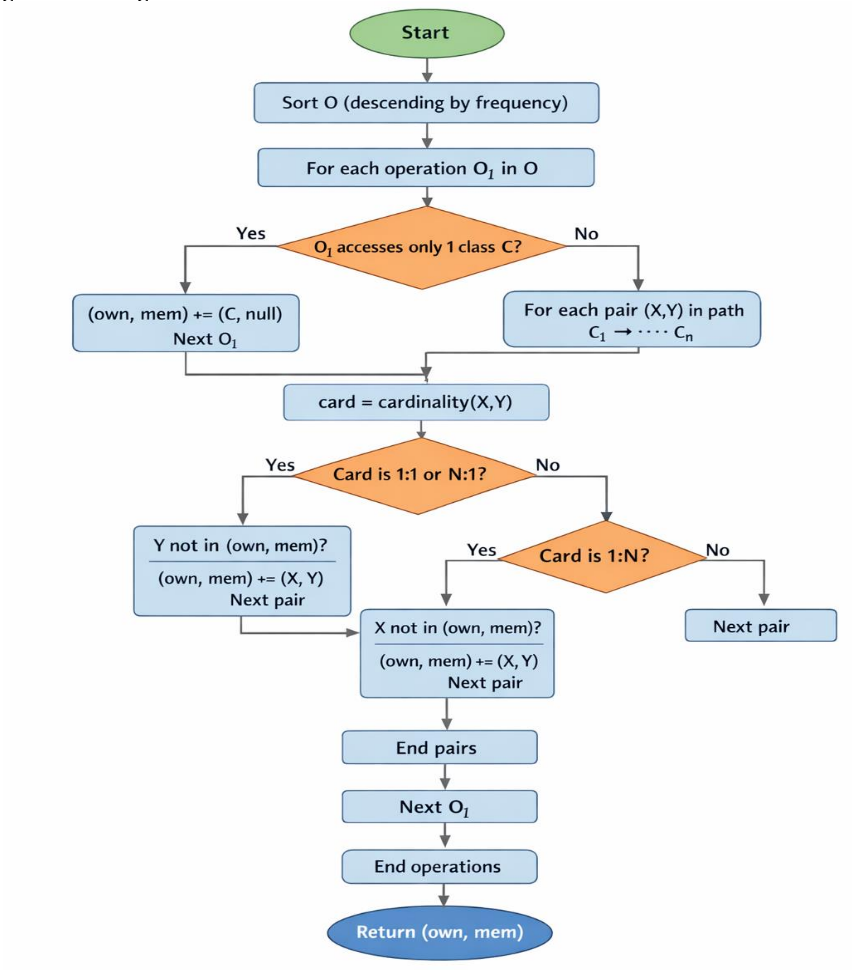

Figure 4. Algorithm defining (owner, member) relationship.

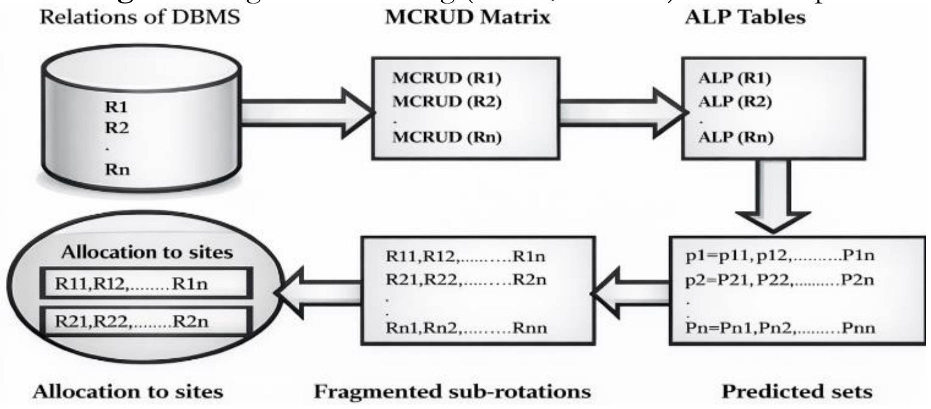

Figure 5. CRUD Matrix

Any relation in the database is defined by the attributes that specify it. However, when information is distributed across different sites, the specifications of the relation do not necessarily carry the same weight.

A CRUD matrix can be defined as a table whose rows represent the attributes of the relation, while its columns are intended to show different placements of implementations as shown in Fig. 5. When data is mapped, this structure is used by programmers and designers and serves as the most vital component of the analysis step in the system's development. We have attempted to modify the existing CRUD model into Modified CRUD (MCRUD) to meet current demands. We can also generate the ALP tables using this modified matrix. When this modified model was used to enhance the attributes of the relation associated with each fragmented site, a significant decrease in costs was observed. MCRUD is used to compute the attributes of a relation based on precedence. MCRUD was considered as the input, and the cost functions were implemented on it as follows:

$$\begin{bmatrix} Bm \\ n \\ o \\ p \end{bmatrix} = [fAA + FSS + fEE + fGG] \quad (3)$$

$$S_m = \sum_{p=1}^{m\,n\,o} C_{m,n,o,p} \qquad (4)$$

where P=m×n×o

$$S_{m,n,q} = \max O(S_{m,n,o}) \qquad (5)$$

$$[ALP_{m,\,n} = S_{m\,n,\,q} - \textstyle\sum_{o\neq m}^{m\,n\,o} s_{m,n,o}] \qquad (6)$$

$$[ALP_{m} = \textstyle\sum_{n=1}^{M} ALP_{m,\,n}] \qquad (7)$$

Here

fA = frequency representing the create operation

fS = frequency representing the read operation

fE = frequency representing the update operation

fG = frequency representing the delete operation

A = weight associated with the create operation

S = weight associated with the read operation

E = weight associated with the update operation

G = weight associated with the delete operation

Bm,n,o,p = cost showing predicate n of attribute m

Sm,n,o = sum of all implementations' costs of predicate n of attribute m

Sm,n,q = maximum cost for the sites associated with predicate n of attribute m

ALPm,n = actual cost associated with predicate n of attribute m

ALPm = sum of the costs related to attribute m (as the locality precedence)

Frequencies are assumed here to ensure the designer's robustness when applying the technique. Based on this assumption, the designer can make predictions about data processing using these values, and the cost will be evaluated as expected. Using this algorithm, a set R containing the attributes with the highest precedence will be generated. This predicate will serve as the standard, and the entire database, combined via a specific relationship, will be horizontally fragmented with respect to it.

Performing the fragmentation of the database:

The (owner, member) relation is fully instantiated once all database operations have been examined. Subsequently, classes from the database schema proceed to the fragmentation stage. Primary horizontal fragmentation is applied to owner classes that do not assume the member role in any (owner, member) relationship occurrence. In contrast, derived horizontal fragmentation is applied to member classes based on their owner. When a class E appears in both the (E, null) and the (F, E) forms of the (owner, member) relation, we have to decide which fragmentation technique to apply: primary according to set (E, null) or derived in accordance with instance (E, F). This decision is made with consideration for the processes that generated each instance.

The (owner, member) relation is cleared of the results produced by the action with a lesser frequency. Keep in mind that this could violate the class path followed by a navigation application (A), decreasing its performance; however, this will only occur if an operation (B) that accesses a class extension in the middle of the C path is more general. The most frequent operation will benefit from improved B performance if this class is chosen as the primary fragmentation class rather than a derived class. The example in the above sections provides a clear illustration of this scenario.

Implementation of the Primary Horizontal Fragmentation Technique:

An extension of the strategy in employed for the primary horizontal fragmentation. The algorithm receives input details about the applications that use the primary fragmented category, including their predicates and execution frequency, to calculate summations of associated objects that are likely to be accessed simultaneously by different programs. These sets of objects will form the class fragments. First, it creates a predicate affinity matrix from the basic predicates used in the applications. Next, it generates a predicate-affinity graph that includes cycles capturing class fragments. This two-step approach is used to carry out the fragmentation process. A predicate affinity matrix for the basic predicates used in the applications is first constructed, and then a predicate affinity graph with cycles that represent class fragments is built. This two-step approach is used to accomplish the fragmentation process.

Predicates are extracted from the applications and used to define the matrix dimensions (rows and columns) in the predicate affinity matrix. Each value in the predicate affinity matrix (ri, rj) denotes the sum of the frequencies of operations that simultaneously access predicates ri and rj. To reduce the number of defined class fragments, the predicate affinity matrix also preserves logical relationships between predicates, such as logical implications. Predicates are grouped using a graph-based model to generate the predicate affinity graph. A cycle may eventually arise in the graph due to graph associations. Each graph cycle will be regarded as a class fragment once a connected graph containing all of the predicates has been structured. Using the logical connectives ∧ and ∨, a class fragment is defined by a Boolean combination of the cycle's predicates.

Implementation of the Derived Horizontal Technique:

The implementation of the derived horizontal technique is based on the predefined (owner, member) relationship, which serves as the foundation for fragmentation. In this approach, horizontal fragments of a member class are created according to the fragmentation scheme already applied to its corresponding owner class. The distribution designer must define derived horizontal fragments for each member class (i.e., the class that serves as the "member") in the (owner, member) relation, grouping objects from various classes accessed via similar navigation operations into a single horizontal fragment.

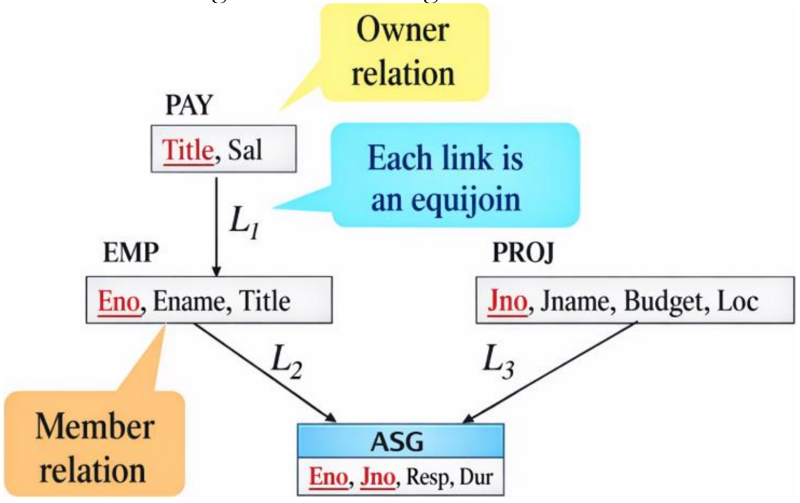

Figure 6. Derived Horizontal Fragmentation

To maintain dependence constraints in a distributed database, Figure 6 shows the horizontal fragmentation derived using equijoin linkages (L1–L3) between the owner and member relations.

Advantages and Disadvantages of the Derived Horizontal Fragmentation:

Derived horizontal fragmentation was engineered by keeping in mind all the drawbacks and disadvantages of previous techniques as follows:

Ability to Provide Enhanced Security: The derived horizontal fragmentation technique has succeeded in providing improved security against all access risks. Users' access to individual attributes no longer affects the operations of other distributed datasets.

Moderate Data Distribution: Derived horizontal fragmentation can be used for equal and moderate distribution of the database among different rows of the table. This technique is equipped with well-defined data distribution schemes, making it easy to fragment the database evenly across sections.

Enhanced Assistance for Queries: Query processing makes use of an attribute-based approach for efficient operation. All queries are treated consistently in this database fragmentation process, making the optimization more robust than in classical techniques.

Improved Storage of Data: Data is always distributed according to similar characteristics among different sets. Data can also be stored consistently for easier retrieval and storage using a derived horizontal approach.

Scalable Access: The nodal distribution of the datasets due to the advent of derived horizontal fragmentation has made the scalability of the system more accurate and faster compared to classical processes. Derived horizontal alignment of data facilitates efficient addition of more versatile databases.

Despite the benefits of the improved fragmentation technique, several drawbacks limit its effectiveness relative to modern paradigms. An attempt has been made to provide an overview of some disadvantages of this technique concerning essential parameters, as given below:

Enhanced Storage Overhead: When fragmenting a large amount of data into smaller parts, this process may result in increased storage overhead for the fragments. This makes the technique less efficient and costlier to use on large datasets.

Complex Startup Assembly: To make the technique more efficient, some modifications have to be made to its structure, which tends to make the overall design more complex to operate. This action also tends to increase the system's initialization time.

Greater Cost of Operation: As technology moves from traditional to advanced approaches, its infrastructure becomes more intricate with interface software systems. These factors not only increase the system's operational complexity but also raise costs.

Results and Discussion:

In this section, the performance of the suggested MCRUD-driven Derived Horizontal Fragmentation (DHF) method is compared to the existing fragmentation and machine-learning-based methods, such as Decision Tree Fragmentation (FTree), Variable Fragmentation, and RFO-Support Vector Machine (SVM).

The evaluation metrics include execution time, query throughput (flow rate), storage overhead, replication cost, scalability, and system availability as given in Table 1. All technique was tested with the same distributed database workloads to ensure fair comparison.

Table 1. Results Comparison

Techniques	Time Taken (ms)	Flow rate (q/s)	Overhead Storage (GB)	Cost of Replication (%)	Scalability (2X nodes)
Proposed MCRUD DHF	120	950	18	12	1.85

MCRUD DHF	160	760	20	15	1.5
(Decision tree) Ftree	145	820	22	18	1.6
Variable Fragmentation	135	870	19	14	1.7
RFO Support vector machine (SVM)	180	700	17	10	1.3

These findings indicate that the proposed MCRUD-DHF strategy consistently performs well across most metrics.

The suggested approach has the shortest execution time (120 ms) and maximum query throughput (950 queries/sec). This is enhanced by the fact that the placement of attributes, locality, and access frequency is taken into account when placing fragments, which minimises unnecessary disk scans and visits to remote nodes.

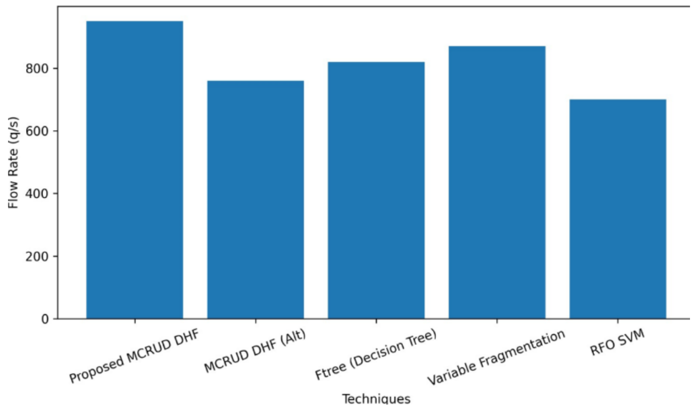

Figure 7. Different Techniques: Flow Rate Comparison.

Fig. 7 shows how well the suggested MCRUD-DHF and current fragmentation approaches perform in terms of flow rate.

Fig. 8 shows the computation (execution) time comparison between the baseline fragmentation technique and the suggested MCRUD-DHF.

These graphs confirm that the suggested approach is faster at processing queries and can handle more queries per second than competing models.

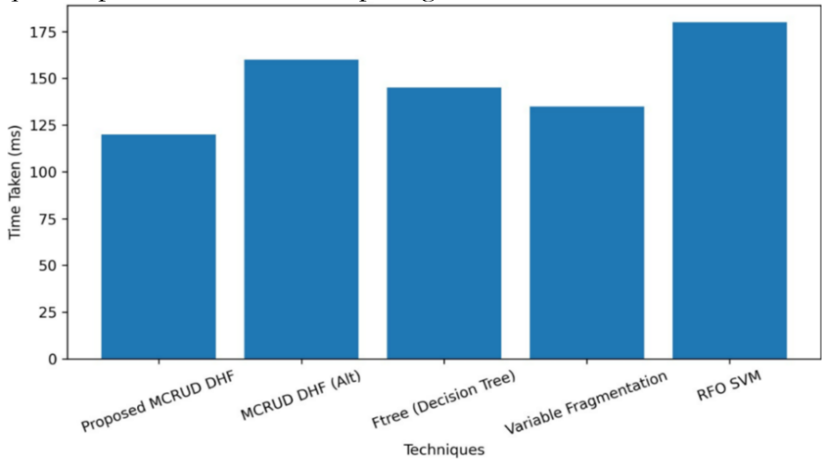

Figure 8. Computation Time Performance Analysis of Different Techniques.

Storage and Replication Efficiency:

Storage consumption and replication costs are also important in distributed systems because they affect memory usage and maintenance overhead.

The storage required by the proposed model is 18 GB, which is less than Decision Tree (22 GB) and Variable Fragmentation (19 GB), and slightly more than SVM (17 GB). SVM, however, does this reduction at the cost of greatly decreased performance.

More to the point, the cost of replicating the proposed system is only 12%, which is significantly lower than that of Decision Tree (18%) and Variable Fragmentation (14%). This means that the MCRUD technique manages the issue of redundancy and efficiency.

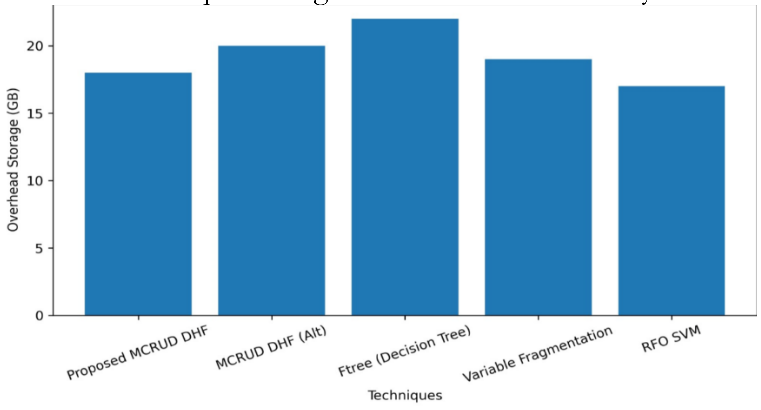

Figure 9. Overhead Storage Performance Analysis of Different Techniques.

In Fig. 9, a storage overhead comparison is performed between MCRUD-DHF and current fragmentation methods in a distributed database setting.

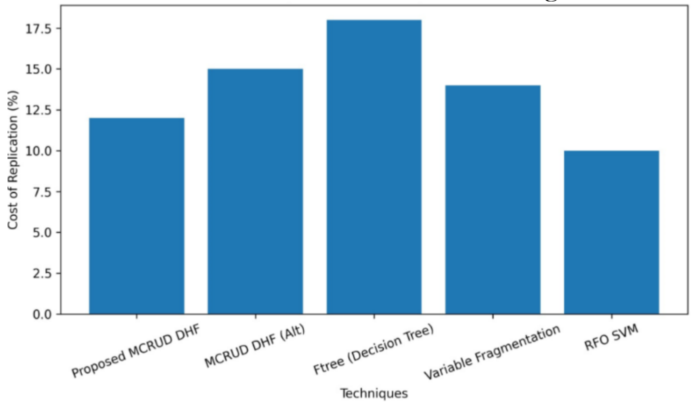

Figure 10. Replication Cost Analysis of Different Techniques.

Fig. 10 shows the relative replication costs of the baseline fragmentation technique and the suggested MCRUD-DHF in a distributed database setting.

The findings indicate that intelligent attribute-based fragmentation reduces redundant duplication without compromising data accessibility.

Scalability and Availability:

Scalability is the factor that determines how the database performs as the nodes are added. The proposed system has the best scalability factor of 1.85 by doubling the number of nodes, which is better than all the existing techniques.

Equally, the system availability is 99.2%, the best among all the models considered. This is because the fragment distribution is better and inter-node dependencies are minimized.

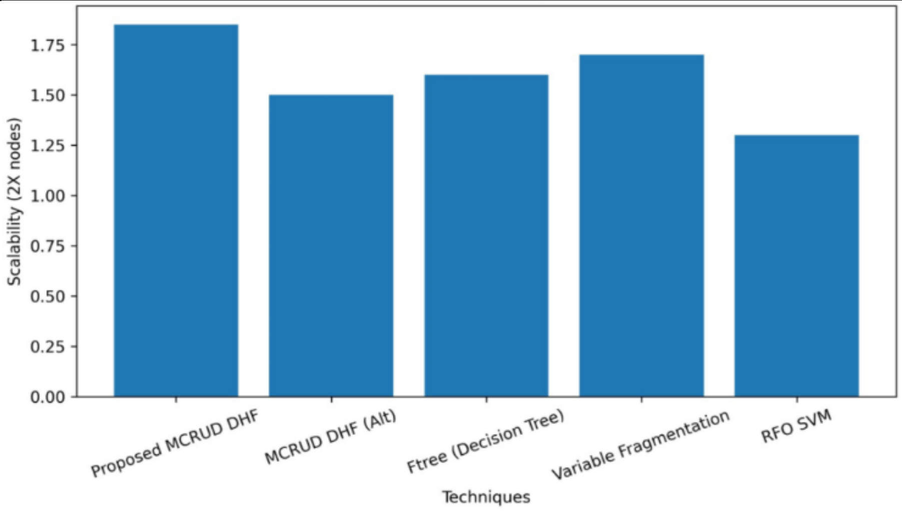

Figure 11. Scalability Performance Analysis of Different Techniques

Fig. 11 shows the performance increase when the number of distributed nodes is doubled (2×), demonstrating the proposed MCRUD-DHF's scalability compared to current fragmentation approaches.

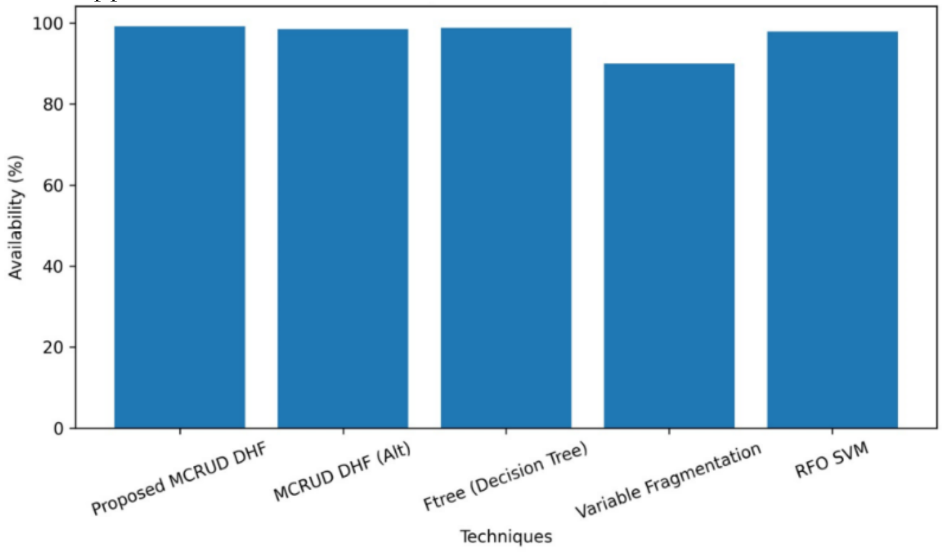

Figure 12. System Availability Performance Analysis of Different Techniques.

In Fig. 11, the proposed MCRUD-DHF and traditional fragmentation approaches' system availability during distributed database operation is compared. The findings validate that the suggested method can be effectively used in large-scale distributed settings where uptime and scalability are paramount.

Discussion:

The experimental results indicate that the proposed MCRUD-DHF method enhances the performance of distributed databases by optimizing fragment placement based on access behavior and attribute locality.

The proposed approach is also much faster than machine learning-based methods such as Decision Trees and SVMs, while maintaining higher throughput. Although SVM has the lowest storage usage, it has the worst response time and throughput, indicating it does not route queries efficiently.

The proposed model's increased scalability implies the system can effectively handle higher workloads without compromising performance. In addition, greater availability indicates that fragmentation minimizes the dependency on individual nodes and reduces the risk of system failure.

The model, however, can incur slight synchronization overhead when a query spans more than one fragment on the nodes. This is a weakness of distributed architectures, which is not compensated for by the high-performance gains.

In general, the suggested fragmentation method offers a trade-off among performance, storage efficiency, and reliability, which makes it applicable to practical distributed database systems, including financial transaction systems and cloud-based enterprise systems.

Conclusion:

This paper describes the Derived Horizontal Fragmentation (DHF) using MCRUD in order to enhance the efficiency of data distribution in a DBMS. The proposed method was intended to optimize fragment placement based on attribute locality and access frequency, thereby minimizing unnecessary data access and communication overhead across distributed nodes.

Evaluating the proposed technique experimentally revealed that it was consistently more effective than current fragmentation methods, including decision-tree-based, variable-based, and SVM-based RFO methods. The experimental results have shown that the system is outperforming in terms of execution time, query flow rate, storage overhead, scalability, and system availability. The above improvements suggest that the MCRUD-DHF strategy offers a more balanced allocation of data while maintaining the reliability and efficiency of operations in distributed database systems.

Despite these benefits, the study has some limitations. The experiments were executed on a controlled dataset and a fixed network setup, which may not accurately reflect highly heterogeneous or large-scale cloud settings. Also, the model currently lacks dynamic workload changes or real-time re-fragmentation.

Future studies will consider adopting adaptive learning to automatically scale fragment allocation in response to alternating workload patterns. The solution can be further applied to cloud and edge computing architecture and tested with real-world big data benchmarks to further scale and strengthen its scalability.

Acknowledgement:

We extend our heartfelt gratitude to Dr. Nisar Ahmed, who played an important role in improving the research article. The manuscript has not been published or submitted to other journals previously.

Author's Contribution:

M. Shoaib and M. U. Younus proposed the Hybrid Approach for Efficient Database Fragmentation, performed the experimental work, and wrote the manuscript, K. Safdar provided guidance during experiments, verified the results, and proofread the manuscript. A. B. Dogar helped in manuscript formatting and refinement.

Conflict of Interest:

The authors declare no conflict of interest.

References:

[1] Oscar Crescencio-Rico, Lisbeth Rodríguez-Mazahua, "Dynamic Hybrid Fragmentation Method for Multimedia Databases," *PÄDI Boletín Científico Ciencias Básicas e Ing. del ICBI*, vol. 11, pp. 47–54, 2023, [Online]. Available: https://www.researchgate.net/publication/373849225_Dynamic_Hybrid_Fragmentation_Method_for_Multimedia_Databases

[2] Masood Niazi Torshiz, Azadeh Salehi Esfaji, "Enhanced Schemes for Data Fragmentation, Allocation, and Replication in Distributed Database Systems," *Comput. Syst. Sci. Eng*, vol. 35, no. 2, 2020, [Online]. Available:

https://www.techscience.com/csse/v35n2/40082

[3] Naveen Kumar Jayakumar, "Engineering for Millions of Requests Per Second: Building Ultra-Low Latency, High-Availability Services at Scale," *J. Comput. Sci. Technol. Stud.*, vol. 8, 2026, [Online]. Available: https://al-kindipublisher.com/index.php/jcsts/article/view/11898

[4] Xinyi Wang, Chye Kiang Heng, Yu Wang, "From fragmentation to framework: A multi- and cross-scale review of property regimes in urban redevelopment," *Habitat Int.*, vol. 167, p. 103652, 2026, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0197397525003686

[5] M. Rajkumar, R. Radhika, "Efficient separation and allocation of dataset in structured and unstructured databases," *Mater. Today Proc.*, vol. 37, no. 2, pp. 2547–2552, 2021, [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S2214785320363744

[6] Felipe Castro-Medina, Lisbeth Rodríguez-Mazahua, "A New Method of Dynamic Horizontal Fragmentation for Multimedia Databases Contemplating Content-Based Queries," *Electronics*, vol. 11, no. 2, p. 288, 2022, [Online]. Available: https://www.mdpi.com/2079-9292/11/2/288

[7] Nidia Rodríguez-Mazahua, Lisbeth Rodríguez-Mazahua, "Decision-Tree-Based Horizontal Fragmentation Method for Data Warehouses," *Appl. Sci*, vol. 12, no. 21, p. 10942, 2022, [Online]. Available: https://www.mdpi.com/2076-3417/12/21/10942

[8] P. Karthikeyan and K. Brindha, "Smart data flow: a trust-driven hybrid fragmentation framework for optimizing data transmission in IoT-enabled edge-fog-cloud systems," *Computing*, vol. 107, no. 6, Jun. 2025, doi: 10.1007/s00607-025-01491-2.

[9] N. A. Sharifah Hafizah Sy Ahmad Ubaidillah, Julius Odili, "Fragmentation Techniques with Ideal Performance in Distributed Database -A Review," *Int. J. Comput. Syst. Softw. Eng.*, vol. 6, no. 1, pp. 18–24, 2020, [Online]. Available: https://www.researchgate.net/publication/344401409_Fragmentation_Techniques_with_Ideal_Performance_in_Distributed_Database_-A_Review

[10] D. Sahithi, J. Keziya Rani, "Efficient fragmentation and allocation on clustering in distributed environment (FACE)," *Meas. Sensors*, 2023, [Online]. Available: https://www.researchgate.net/publication/375470854_Efficient_fragmentation_and_allocation_on_clustering_in_distributed_environment_FACE

[11] Ali Amiri, "Maximizing data utility while preserving privacy through database fragmentation," *Expert Syst. Appl.*, vol. 273, p. 126873, 2025, [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0957417425004956

[12] P. J. Liu, C. P. Li, and H. Chen, "Enhancing Storage Efficiency and Performance: A Survey of Data Partitioning Techniques," *J. Comput. Sci. Technol. 2024 392*, vol. 39, no. 2, pp. 346–368, Jun. 2024, doi: 10.1007/s11390-024-3538-1.

[13] Yuankun Zhang, Zhaoxuan Zhang, "Adaptive DNN Partitioning for Edge-Cloud Systems with Meta-Reinforcement Learning," *Proc. 18th IEEE/ACM Int. Conf. Util. Cloud Comput. UCC 2025*, 2025, [Online]. Available: https://dl.acm.org/doi/10.1145/3773274.3774271

[14] Jing Jiang, Yushu Su, "Multi-Link Fragmentation-Aware Deep Reinforcement Learning RSA Algorithm in Elastic Optical Network," *Photonics*, vol. 12, no. 7, p. 634, 2025, [Online]. Available: https://www.mdpi.com/2304-6732/12/7/634

[15] "Data Fragmentation Explained: Causes and Architectural Solutions." Accessed: Feb. 16, 2026. [Online]. Available: https://estuary.dev/blog/data-fragmentation/

[16] shymaa hosny mahmoud, N. Badr, A. E. Abdelraouf, and M. I. Ali, "RL-Based Fragment Allocation and Replication for Distributed Heritage Multimedia Databases," *Int. J. Intell. Comput. Inf. Sci.*, vol. 25, no. 3, pp. 73–92, Sep. 2025, doi:

10.21608/ijicis.2025.411803.1419.

[17] Senhong Cai, Zhonghua Gou, "Towards energy-efficient data centers: A comprehensive review of passive and active cooling strategies," *Energy Built Environ.*, vol. 7, no. 1, pp. 206–226, 2026, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666123324000916

[18] Hamid Sarkheil, Taha Salahjou, Amirhossein Hashemi, "A robust hybrid machine learning framework for multidimensional ESG assessment in critical mineral supply chains: The case of cobalt mining," *Results Eng.*, vol. 29, p. 109501, 2024, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590123026005414

[19] Jin Li, "Distributed Data Processing and Real-Time Query Optimization in Microservice Architecture," *J. Comput. Signal, Syst. Res.*, vol. 2, no. 4, 2025, [Online]. Available: https://www.gbspress.com/index.php/JCSSR/article/view/321

[20] N. G. Totaro, G. Specchia, A. Corallo, and M. Gervasi, "Strategic management of human skills in Big Data initiatives: from SLR to skills taxonomy and human resource management framework," *Int. J. Data Sci. Anal. 2026 221*, vol. 22, no. 1, pp. 7-, Jan. 2026, doi: 10.1007/s41060-025-01004-6.

[21] Fernanda Baiao, Marta Mattoso, "A Knowledge-Based Perspective Of The Distributed Design Of Object Oriented Databases," *WIT Trans. Inf. Commun. Technol.*, vol. 22, no. 2, 2026, [Online]. Available: https://www.witpress.com/elibrary/wit-transactions-on-information-and-communication-technologies/22/6949

[22] Kassem Danach, Abdullah Hussein Khalaf, "Enhancing DDBMS Performance through RFO-SVM Optimized Data Fragmentation: A Strategic Approach to Machine Learning Enhanced Systems," *Appl. Sci.*, vol. 14, no. 14, p. 6093, 2024, [Online]. Available: https://www.mdpi.com/2076-3417/14/14/6093

[23] D. D. Scott T. Leutenegger, "A modeling study of the TPC-C benchmark," *ACM SIGMOD Rec.*, vol. 22, no. 2, p. 1, 1993, [Online]. Available: https://dl.acm.org/doi/abs/10.1145/170036.170042