Alqalam Journal of Medical and Applied Sciences. 2026;9(4):849-855
https://doi.org/10.54361/ajmas.269410

Original article

Adaptive Query-Driven Fragmentation for Performance Optimization in Distributed Databases

Zakia Ahmed*, **Anwar Alhenshiri**

Department of Computer Science, Faculty of Information Technology, Misurata University, Misurata, Libya
Corresponding E-mail. *z.ahmed@it.misuratau.edu.ly*

Abstract
Distributed databases improve scalability and availability but introduce challenges related to query performance and data management. Fragmentation is a key technique used to enhance efficiency by reducing irrelevant data access. This paper proposes an adaptive fragmentation approach based on real query patterns to optimize distributed database performance. The method applies horizontal and derived fragmentation techniques driven by frequently executed queries, followed by a genetic algorithm to reduce redundant predicates and minimize fragment growth. The approach was implemented and evaluated using the TPC-H benchmark on a PostgreSQL-based distributed environment. Experimental results demonstrate notable reductions in query response time and communication overhead while maintaining high query coverage. The findings confirm that query-driven adaptive fragmentation can effectively improve distributed database performance and resource utilization.
Keywords. Distributed Database, Horizontal Fragmentation, Query Pattern, Derived Fragmentation, Genetic Algorithm.

Introduction

The rapid growth of data-intensive applications and real-time information systems has created an increasing demand for scalable and efficient data management solutions. Traditional centralized database systems are often unable to manage massive volumes of data generated by modern applications, which leads to performance bottlenecks and limited scalability. Distributed databases (DDB) have emerged as a practical solution to these challenges by distributing data across multiple interconnected nodes, thereby improving availability, scalability, and fault tolerance [1]. Despite their advantages, distributed databases introduce several technical and operational challenges. Storing data across multiple sites can increase communication overhead, complicate transaction management, and negatively affect query response time. To address these issues, distributed database design relies on two fundamental processes: fragmentation and allocation [2]. Fragmentation divides database relations into smaller logical units so that queries access only relevant data, while allocation determines how these fragments are distributed among different nodes. Proper fragmentation can significantly reduce data transfer and improve query performance.

Fragmentation techniques are commonly classified into horizontal, vertical, hybrid, and derived fragmentation, where the effectiveness of any fragmentation strategy depends largely on workload characteristics and query patterns [3]. Most traditional fragmentation approaches assume static workloads and perform fragmentation at design time. However, real-world distributed systems are characterized by dynamic and continuously changing query patterns. Static fragmentation schemes often fail to adapt to such changes, which leads to inefficient resource utilization and degraded performance [4].

Evolutionary and meta-heuristic algorithms have also been widely used to address the combinatorial complexity of fragmentation design [5]. Although these approaches demonstrate promising results, many require extensive training data, complex modeling, or significant computational resources. However, many existing methods either rely on predefined workloads, focus on a single fragmentation technique, or do not address the problem of fragment explosion when large numbers of predicates are involved. These limitations highlight the need for a lightweight and adaptive fragmentation approach that directly utilizes real query execution patterns without requiring complex learning models. The proposed work addresses this gap by combining horizontal and derived fragmentation techniques based on actual query logs. To overcome the exponential growth of fragments caused by large predicate sets, a genetic algorithm is employed to select an optimal subset of predicates that preserves query coverage while minimizing storage and communication costs. This study proposes a query-pattern-driven adaptive fragmentation framework with the following main contributions:

1. A practical adaptive fragmentation model based on the analysis of frequently executed queries.
2. Integration of horizontal and derived fragmentation techniques to efficiently support join-intensive workloads.
3. Application of a genetic algorithm to reduce predicate sets and control fragment explosion.
4. Demonstration of significant improvements in query response time and overall system performance.

Early research primarily focused on static fragmentation strategies, where data partitioning decisions are made at design time based on predefined assumptions about database structure and expected workloads. However, with the increasing complexity and dynamism of modern applications, more recent studies have

Received*: 05-02-2026 -* ***Accepted****: 04-04-2026 -* ***Published****: 12-04-2026*

shifted toward adaptive and workload-aware fragmentation approaches. Castro-Medina et al. [3] reviewed various dynamic fragmentation methods and emphasized that the effectiveness of any fragmentation technique depends heavily on query patterns and application characteristics. Their study highlighted that static fragmentation methods often struggle to adapt to evolving workloads, which can lead to inefficient resource utilization in real-world distributed systems.

To overcome the limitations of static fragmentation, many earlier works proposed fragmentation methods that rely on design-time information. Khan and Hoque [6] introduced a horizontal fragmentation technique that determines fragments during the initial design stage of a distributed database based on requirements analysis. Similarly, Abdel-Raouf et al. [7] proposed a vertical fragmentation and allocation strategy applied at the early stages of database design. These approaches assume that query workloads remain relatively stable over time. Mehta et al. [8] focused on optimizing vertical fragmentation using the Differential Bond Energy Algorithm to determine optimal partition points. Their work demonstrated that optimization techniques can improve fragmentation quality when workload characteristics are known in advance. However, such static methods do not address the dynamic nature of modern distributed systems, where query patterns frequently change. Diaz et al. [9] experimentally evaluated workload-based partitioning and sharding strategies and showed that fragmentation guided by actual query execution patterns significantly improves scalability and performance in distributed environments. These findings reinforce the importance of query-pattern analysis as a foundation for effective fragmentation.

The present work proposes an adaptive fragmentation approach that applies horizontal and derived fragmentation techniques based on real workloads and employs a genetic algorithm to reduce predicate sets and control fragment explosion. This combination enables practical and lightweight adaptive fragmentation without requiring complex learning models or predefined workloads. The remainder of this paper is organized as follows. Section 2 presents the proposed methodology and system model. Section 3 discusses experimental results and performance evaluation. Finally, Section 4 concludes the paper and outlines future research directions.

Methodology

This section describes the proposed adaptive fragmentation framework, the dataset used for evaluation, the system architecture, and the optimization strategy based on a genetic algorithm.

Overview of the Proposed Framework

The proposed methodology is based on the idea that fragmentation decisions should be guided by real workload characteristics rather than static design-time assumptions. Instead of predefining fragments manually, the system continuously analyzes executed queries to determine the most relevant predicates and join operations. These extracted patterns are then used to generate horizontal and derived fragments dynamically.

The overall structure of the proposed adaptive fragmentation framework is presented in (Figure 1).

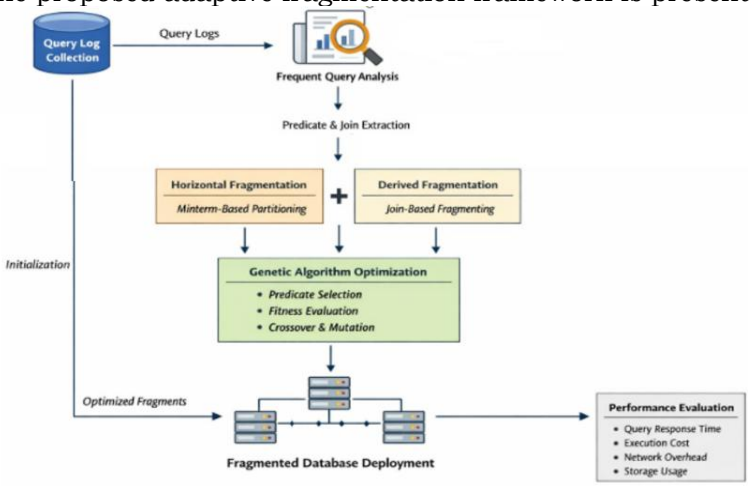

Figure 1. Proposed Adaptive Fragmentation Framework

The overall workflow of the proposed approach consists of the following main stages:

1. Construction of a distributed database environment.
2. Collection and analysis of executed query logs.
3. Extraction of predicates and join conditions from frequent queries.
4. Application of horizontal and derived fragmentation techniques.
5. Optimization of fragment generation using a genetic algorithm.
6. Evaluation of system performance before and after fragmentation.

Received: *05-02-2026* - ***Accepted***: *04-04-2026* - ***Published***: *12-04-2026*

This process enables the system to adapt to changing workloads and to continuously improve fragmentation quality based on observed query behavior.

Dataset

To evaluate the effectiveness of the proposed approach, the TPC-H benchmark dataset was used. TPC-H is a widely accepted benchmark developed by the Transaction Processing Performance Council (TPC) for evaluating decision-support systems and analytical query performance. It consists of eight normalized relational tables and a set of 22 complex SQL queries designed to simulate real-world business analysis workloads.

System Model

A distributed database environment was established to evaluate the effectiveness of the proposed fragmentation approach. The system consisted of multiple database nodes interconnected through a network, with each node configured using PostgreSQL as the underlying database management system. This setup was designed to simulate a realistic distributed architecture in which queries could be executed across several coordinated servers.

During query execution, detailed logs were collected from all nodes to capture real workload characteristics, including executed SQL statements, timestamps, node identifiers, and execution frequencies. The query log was periodically analyzed to identify the most frequently executed queries. In accordance with the Pareto principle, the top 20 percent of queries—representing approximately 80 percent of total workload—were selected for detailed analysis. These frequent queries formed the basis for extracting predicates and join relationships that subsequently guided the adaptive fragmentation process.

Fragmentation Model

The core component of the methodology is the adaptive fragmentation model. Two complementary fragmentation techniques were employed: horizontal fragmentation and derived fragmentation.

Minterm-Based Horizontal Fragmentation

This approach considers all possible logical combinations of the predicates extracted from frequent queries to generate highly selective fragments. For a relation containing n predicates, up to 2^n minterms can be produced, with each minterm representing a distinct fragment that corresponds to a specific combination of conditions. During the fragmentation process, only non-empty fragments are retained, which guarantees the essential properties of completeness and disjointness. While this technique significantly improves query selectivity and ensures that queries access only relevant data, it can also result in fragment explosion when many predicates are involved, thereby increasing storage and management overhead.

Derived Fragmentation

To efficiently support join-intensive queries, derived fragmentation was integrated into the framework. Derived fragmentation creates fragments of a child relation based on the fragmentation of a parent relation using semijoin operations. This technique collocates related tuples that participate in join operations on the same node, thereby minimizing inter-node communication. For example, if a primary relation is horizontally fragmented based on specific predicates, related tables involved in join queries are fragmented accordingly so that join operations can be executed locally.

Genetic Algorithm for Predicate Optimization

A major challenge in predicate-based fragmentation is the exponential growth of fragments when many predicates are extracted from queries. To address this issue, a genetic algorithm (GA) was incorporated to select an optimal subset of predicates that balances performance and storage cost.

Each chromosome in the GA population represents a candidate subset of predicates. The objective of the GA is to maximize query performance while minimizing fragment size and predicate count. The fitness function was defined as follows:

$$Fitness = 0.4 \times Fragment\ Size + 0.5 \times Query\ Coverage + 0.1 \times (1 - Predicate\ Ratio)$$

This function prioritizes high query coverage and smaller fragment sizes while discouraging excessive predicate use.

The genetic algorithm utilized standard evolutionary operations to search for an optimal subset of predicates. Tournament selection was applied to identify the most promising individuals from the population, while single-point crossover was used to combine candidate solutions and explore new configurations. To maintain diversity and avoid premature convergence, bit-flip mutation was incorporated into the evolutionary process. The algorithm was executed with a population size of 50 individuals and a maximum of 100 generations, and it terminated when no improvement in the best fitness value was observed for 15 consecutive generations.

Received: *05-02-2026* - ***Accepted***: *04-04-2026* - ***Published***: *12-04-2026*

The proposed approach was implemented using the following technologies: Python as a programming language, PostgreSQL as a database management system, Windows 11 Pro, and 8 GB RAM.

Evaluation Strategy

System performance was evaluated using a comprehensive set of quantitative metrics designed to capture different aspects of distributed database efficiency. These metrics included *query response time, query execution cost, number of processed rows, network communication overhead, and storage utilization*. Measurements were collected for selected TPC-H queries under two main conditions: before applying any fragmentation and after implementing the proposed adaptive fragmentation approach. This comparative evaluation enabled a systematic and objective assessment of the impact of the methodology on overall system performance.

Results

This section presents the experimental results obtained from evaluating the proposed adaptive fragmentation framework.

Baseline Results (Before Fragmentation)

To establish a reference point for evaluation, all experiments were first conducted on the distributed database system without applying any fragmentation techniques. In this baseline configuration, queries were executed directly on the complete dataset. For every evaluated query, several performance indicators were recorded, including total execution time, PostgreSQL optimizer plan cost, number of scanned tuples, and the overall amount of processed data.

The baseline measurements reflected the behavior of a conventional distributed database environment in which no workload-aware optimization is performed. Under this configuration, queries accessed the entire relations regardless of the actual amount of relevant data, resulting in relatively high processing costs and response times as illustrated in (Figure 2). These results served as the initial benchmark against which the impact of the proposed adaptive fragmentation approach was later compared.

Results After Adaptive Fragmentation

After applying the proposed adaptive fragmentation framework, the distributed database system was reevaluated using the same set of TPC-H queries and performance metrics. Horizontal fragmentation was first performed based on predicates extracted from frequently executed queries, followed by derived fragmentation to optimize relations involved in join operations. This reconfiguration modified the physical organization of the database so that queries could access only the relevant portions of data rather than scanning entire tables.

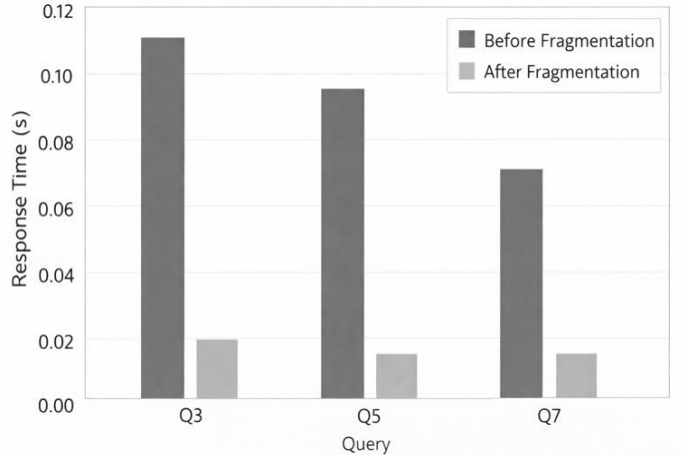

Figure 2. Query Response Time Comparison.

The measurements collected after fragmentation showed clear changes in system behavior. As illustrated in (Figure 3), the amount of scanned data was significantly reduced for all evaluated queries after adaptive fragmentation, confirming that queries accessed only the relevant fragments instead of entire relations.

Received*: 05-02-2026 -* ***Accepted****: 04-04-2026 -* ***Published****: 12-04-2026*

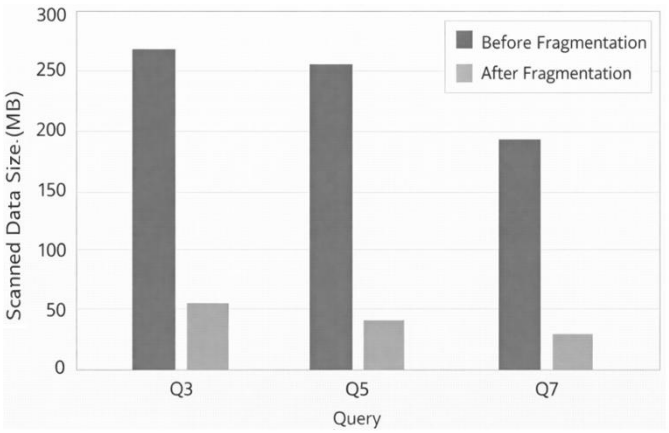

Figure 3. Reduction in the amount of Scanned Data Comparison

(Table 1) summarizes sample performance results for selected TPC-H queries before and after the application of adaptive fragmentation. These results provide a direct comparison between the baseline configuration and the fragmented database environment under identical workload conditions. By analyzing the system cost by measuring the execution time of query, the number of returned rows, storage usage and plan cost before and after this fragmentation approach, the finding demonstrated that sufficient improve of performance after applying the proposed approach by reducing the query execution time and plan cost that implies CPU resources and PostgreSQL query optimizer. There is increasing in storage size due to the metadata and the schema of created fragments.

Table 1. Performance Comparison Before and After Adaptive Fragmentation

Query	Configuration	Query Response Time (s)	Improvement	Plan Cost	Storage Used
Q3	Before Fragmentation	0.028	81.9%	5230	13.3 MB
	After Fragmentation	0.005	–	13582	14.5 MB
Q5	Before Fragmentation	0.049	99.5%	3.051	13.3 MB
	After Fragmentation	0.001	–	1.143	14.4 MB
Q7	Before Fragmentation	0.105	98.5%	3.349	4.5 MB
	After Fragmentation	0.002	–	2.694	5.0 MB

Results After Genetic Algorithm Optimization

In term of reducing the used storage, the genetic algorithm improved the fragmentation process by reducing the predicate set while preserving the query performance. Following the initial fragmentation stage, a genetic algorithm was applied to optimize the predicate set used in fragment generation. The objective of this optimization was to reduce the number of generated fragments while preserving the performance benefits achieved through adaptive fragmentation. As shown in (Figure 4), the genetic algorithm converged after approximately 40 generations, after which no significant improvement in fitness was observed.

Received: *05-02-2026* - ***Accepted***: *04-04-2026* - ***Published***: *12-04-2026*

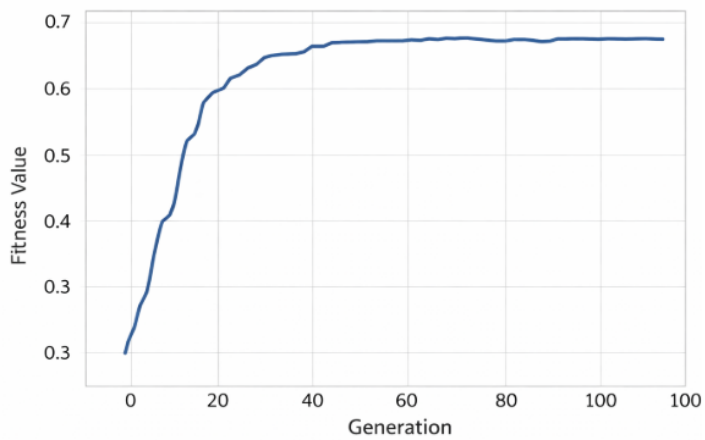

Figure 4. Genetic Algorithm Convergence over Generations

The optimization process resulted in a significant reduction in the number of predicates and corresponding fragments. More than half of the initially generated minterms were eliminated, while approximately 95 percent of the frequent query workload remained fully covered by the optimized fragment set. Storage usage was recalculated after optimization to measure the impact of fragment reduction, and query performance was reevaluated to confirm that the improvements achieved through fragmentation were maintained.
The application of the genetic algorithm led to a substantial decrease in the total number of fragments, reducing them from the initial set to a much smaller optimized subset. In addition to reducing the number of fragments, the optimization process also decreased the overall storage overhead associated with fragmentation, as in (Figure 5).

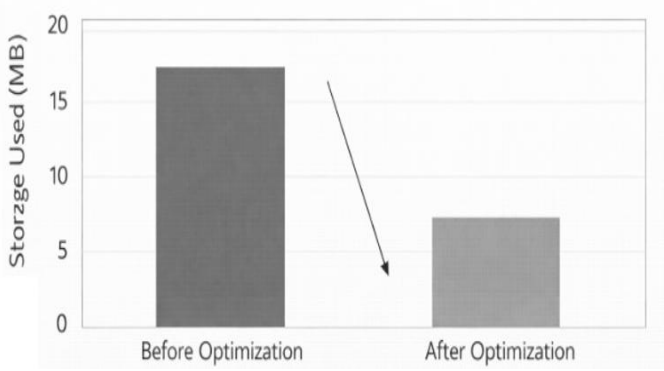

Figure 5. Storage Utilization Before and After Genetic Algorithm Optimization.

The outcomes of this stage demonstrate the effectiveness of the genetic algorithm in balancing fragment size, query coverage, and storage overhead. Performance metrics recorded after optimization were compared with both the baseline system and the post-fragmentation configuration to assess the overall contribution of the genetic algorithm to the proposed framework.

Discussion

The experimental results demonstrate the effectiveness of the proposed adaptive fragmentation framework in improving distributed database performance. By analyzing real query patterns and dynamically generating fragments accordingly, the system was able to significantly reduce query response time, data processing overhead, and network communication cost. The discussion that follows interprets these outcomes in relation to the research objectives.

Impact of Adaptive Fragmentation

The results presented in (Figures 2 and 3) clearly show that applying horizontal and derived fragmentation techniques had a substantial impact on system efficiency. Query response times were consistently reduced across all evaluated TPC-H queries. This improvement is primarily due to the fact that queries accessed only the relevant fragments instead of scanning entire relations, which minimized unnecessary data processing. The reduction in scanned data size, illustrated in (Figure 3), confirms that the fragmentation strategy effectively isolated frequently accessed tuples into smaller logical units. As a consequence, the PostgreSQL query optimizer was able to generate more efficient execution plans, leading to faster query processing. These findings indicate that workload-driven fragmentation is a practical approach for optimizing distributed databases that experience repetitive and predictable query patterns.

Received: *05-02-2026* - ***Accepted***: *04-04-2026* - ***Published***: *12-04-2026*

Effectiveness of Derived Fragmentation

Join operations represent one of the most expensive tasks in distributed database systems, especially when large relations must be transferred across network nodes. The use of derived fragmentation in the proposed framework played a crucial role in addressing this issue. By collocating related tuples that participate in join operations, many queries were executed locally without requiring inter-node communication.

The performance improvements observed for join-intensive queries, particularly Query 3 and Query 7, demonstrate the advantage of combining horizontal fragmentation with derived fragmentation. These results confirm that fragmentation strategies that consider join relationships can significantly reduce communication overhead and enhance overall system scalability.

Role of Genetic Algorithm Optimization

The optimization process resulted in a considerable reduction in fragment count. More than 50 percent of the initially generated fragments were eliminated while still maintaining approximately 95 percent coverage of frequent queries. This demonstrates that many predicates extracted from the workload were either redundant or contributed little to actual performance gains. Furthermore, (Figure 5) indicates that storage utilization decreased after optimization. Although initial fragmentation increased storage overhead due to metadata and fragment replication, the genetic algorithm successfully mitigated this issue by selecting a compact subset of effective predicates.

Comparison with Existing Approaches

The results obtained in this study are consistent with previous research that emphasizes the importance of workload-aware fragmentation. However, unlike machine-learning-based methods that require extensive training data, the proposed approach relies directly on executed query logs, making it simpler and more practical to implement. In addition, the combination of horizontal and derived fragmentation provides more comprehensive optimization than methods that focus on a single technique. Compared to static fragmentation strategies, the adaptive nature of the proposed framework allows it to respond to changing workloads without manual intervention. This flexibility represents a significant advantage for real-world distributed database environments where query patterns evolve over time.

Conclusion

This paper proposed an adaptive fragmentation framework for distributed databases based on real query execution patterns. The approach combines horizontal and derived fragmentation techniques to generate fragments dynamically according to frequently executed queries. Experimental evaluation demonstrated that the proposed method significantly improves system performance. Query response times and the amount of scanned data were substantially reduced after fragmentation, while derived fragmentation effectively minimized the cost of distributed join operations. Genetic algorithm optimization further decreased the number of fragments and overall storage utilization without negatively affecting query coverage. The experimental findings confirm that query-pattern-driven fragmentation provides an effective and practical mechanism for improving distributed database performance. Future work will extend this framework by integrating adaptive fragmentation with intelligent fragment allocation techniques and by evaluating the approach on larger-scale, real-world datasets with highly dynamic query patterns.

References

1. Cireşan DC, Meier U, Masci J, Gambardella LM, Schmidhuber J. High-performance neural networks for visual object classification. arXiv Prepr arXiv1102.0183. 2011.
2. Nashat D, Amer AA. A comprehensive taxonomy of fragmentation and allocation techniques in distributed database design. ACM Comput Surv. 2018;51(1):1–25.
3. Castro-Medina F, Rodríguez-Mazahua L, López-Chau A, Cervantes J, Alor-Hernández G, Machorro-Cano I. Application of dynamic fragmentation methods in multimedia databases: a review. Entropy. 2020;22(12):1352.
4. Malysheva M, Prymak I. Evaluation of Unsupervised and Reinforcement Learning approaches for Horizontal Fragmentation [dissertation]. 2019.
5. Danach K, Khalaf AH, Rammal A, Harb H. Enhancing DDBMS Performance through RFO-SVM Optimized Data Fragmentation: A Strategic Approach to Machine Learning Enhanced Systems. Appl Sci. 2024;14(14).
6. Khan SI, Hoque A. A new technique for database fragmentation in distributed systems. Int J Comput Appl. 2010;5(9):20–24.
7. Abdel-Raouf O, Abdel-Baset M, El-henawy I. A new hybrid flower pollination algorithm for solving constrained global optimization problems. Int J Appl Oper Res. 2014;4(2):1–13.
8. Mehta S, Agarwal P, Shrivastava P, Barlawala J. Differential bond energy algorithm for optimal vertical fragmentation of distributed databases. J King Saud Univ Inf Sci. 2022;34(1):1466–1471.
9. Diaz R, Gomez U, Adams T. Experimental Evaluation of Data Partitioning and Sharding Approaches for Scalability in Distributed Databases. J Innov Gov Bus Pract. 2025;1(1):118–147.

Received: *05-02-2026* - ***Accepted***: *04-04-2026* - ***Published***: *12-04-2026*