computers

MDPI

Article

A Hybrid Fragmentation Method for Multimedia Databases Considering Dynamic Access Patterns

Oscar Crescencio-Rico [1], Felipe Castro-Medina [1], Lisbeth Rodríguez-Mazahua [1,*], Jair Cervantes [2], Giner Alor-Hernández [1], Nidia Rodríguez-Mazahua [1] and Mario Leoncio Arrioja-Rodríguez [1]

[1] Tecnológico Nacional de México/I. T. Orizaba, Av. Oriente 9 852, Col. Emiliano Zapata, Orizaba 94320, Veracruz, Mexico; msc.ocrescencio@ito-depi.edu.mx (O.C.-R.); felipe.cm@orizaba.tecnm.mx (F.C.-M.); giner.ah@orizaba.tecnm.mx (G.A.-H.); dci.nrodriguez@ito-depi.edu.mx (N.R.-M.); mario.ar@orizaba.tecnm.mx (M.L.A.-R.)

[2] Centro Universitario UAEM Texcoco, Universidad Autónoma del Estado de México, Av. Jardín Zumpango, s/n, Fraccionamiento El Tejocote, Texcoco 56259, Estado de México, Mexico; jcervantesc@uaemex.mx

* Correspondence: lrodriguezm@ito-depi.edu.mx

Abstract

The volume of multimedia data generated by various applications is increasing constantly. These data are complex to process, hindering efficient information retrieval because database operations require analyzing an entire database during their execution, augmenting the query execution costs and response times. In a distributed environment, fragmentation is an effective technique that aims to improve query performance and increase the availability of information. Most methods address horizontal or vertical partitioning and are static because they cannot adapt the fragmentation scheme to new changes in database access patterns or elements (attributes and/or tuples). This work presents a dynamic hybrid fragmentation method for multimedia databases that includes an observer–fragmenter developed in the Java programming language and uses the NoSQL (Not only SQL) MongoDB database. To implement the proposed method, it is first necessary to carry out static hybrid fragmentation through the XAMANA Web application. Then, the observer–fragmenter is executed, which is capable of analyzing the fragment costs and monitoring the operation and performance thresholds without the intervention of the Database Administrator (DBA). When the cost of a fragment exceeds the thresholds, a new hybrid fragmentation is automatically triggered, and new fragments are allocated to the corresponding sites. Finally, to validate our proposed method, we conduct a case study using the HITO (History of the Technological Institute of Orizaba) database and compare our method with three state-of-the-art approaches. The results show that it is possible to reduce the execution cost of 77% of queries by accessing less remote data and requiring fewer joins.

Keywords: hybrid fragmentation; dynamic fragmentation; cost model; multimedia database

Check for updates

Academic Editor: George Angelos Papadopoulos

Received: 1 August 2026
Revised: 27 September 2026
Accepted: 28 September 2026
Published: 4 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Multimedia databases are of great importance in information systems, since they manage a large amount of data of different types. Therefore, it is necessary to develop techniques to efficiently access the information stored in these databases. Fragmentation is a technique that allows for the information in a database table or collection to be distributed into fragments, in such a way that the partitions are stored at the closest site of use [1]. There are three types of fragmentation: horizontal fragmentation divides a relation along

its tuples (rows), such that each fragment has a subset of the tuples in the relation; vertical fragmentation consists of dividing a relation into subsets of attributes (columns) and placing the primary key of the relation in each fragment to allow for its reconstruction; and, finally, hybrid fragmentation is the combination of horizontal–vertical partitioning or vice versa [1]. These methods are classified into static and dynamic; the former obtain a fragmentation scheme according to a workload, but if this changes, the scheme cannot adapt to new queries to remain efficient. In contrast, dynamic methods address this problem by continuously modifying the scheme according to changes in the workload [2]. Recently, several horizontal, vertical, and hybrid fragmentation methods for multimedia databases have been proposed [3–8]; however, most are static. Due to the dynamic nature of access patterns to multimedia databases and the advantages that hybrid fragmentation has over vertical and horizontal in terms of reducing irrelevant data both locally and remotely, it is necessary to develop dynamic hybrid fragmentation methods for this type of database.

This paper introduces a dynamic hybrid partitioning method for multimedia databases. It is divided into two parts: static fragmentation and dynamic fragmentation. In the first part, once the workload of the collection to be fragmented is available, the vertical and horizontal fragmentation schemes are obtained to subsequently carry out the hybrid fragmentation, where two schemes are generated: vertical–horizontal and horizontal–vertical. The first scheme undertakes the allocation of the fragments to the sites in the vertical scheme, and the second considers the allocations in the horizontal scheme. The costs of the two schemes are compared, and the less expensive scheme is determined so that the DBA can decide whether it will be materialized. The static hybrid fragmentation scheme is obtained through the XAMANA Web application [3] according to the data provided by the DBA (name of the database, table to be fragmented, and log file).

In contrast, dynamic hybrid fragmentation starts when the observer–fragmenter method is executed; it improves static hybrid fragmentation since it can analyze the costs of the fragments and monitor the operation and performance thresholds without the intervention of the DBA. When the number of new operations executed on a partition and the cost of a fragment reach the thresholds, a new partitioning is automatically triggered, and the new partitions produced are allocated to the corresponding sites. The operation threshold indicates the number of newly executed operations required to perform a cost analysis of the current scheme, and the performance threshold sets the cost needed to perform a new partitioning. When the values of the thresholds are high, new fragmentations are less likely to be performed; when they are lower, new fragmentations are frequently carried out.

Finally, we demonstrate the efficacy and efficiency of our fragmentation method using a case study where the obtained hybrid fragmentation schemes comply with the three rules of fragmentation: completeness, reconstruction, and disjunction. In addition, they have a lower query execution cost than the schemes found in other related works.

This paper makes three contributions: (1) It presents the development of the dynamic hybrid fragmentation method for multimedia databases proposed in [8]. Crescencio-Rico et al. [8] only introduced the method's workflow; in this paper, we include the complete cost model, the algorithms, and the evaluation of our approach. (2) In contrast to other previous works, where we addressed static and dynamic vertical [9], horizontal [10], and hybrid [11] fragmentation for multimedia databases, in this paper we also consider the allocation of the fragments to the sites of the network. (3) Although previous approaches [3,12,13] have focused on the multimedia database design problems of fragmentation and allocation, they only performed horizontal [3,12] or vertical [13] partitioning while, in this paper, we contemplate the dynamic hybrid fragmentation of multimedia databases.

This article is organized as follows: Section 2 defines the terms used in this research, while Section 3 presents related work on dynamic hybrid fragmentation. The method

proposed in this work is described in detail in Section 4. The evaluations and results are shown in Section 5. In Section 6, various aspects to consider when applying the presented method are discussed. Finally, Section 7 presents the conclusions and future work.

2. Background

The main terms used in this research are described below.

2.1. Multimedia Databases

Multimedia databases are a collection of data that, in addition to containing text, store and retrieve audio, images, and video. When including data whose main characteristic is a large size, optimal management is complicated because, as the multimedia content increases, the query response time rises due to the greater volume of irrelevant data accessed. Therefore, applying fragmentation techniques in these databases is necessary to improve query performance [2].

2.2. Fragmentation in Databases

Database fragmentation (or partitioning) is a process for reducing irrelevant data access by grouping frequently accessed data into dedicated segments, called fragments [14]; in this way, query response times are reduced, lowering the execution costs for each operation. The goal of fragmentation is to divide a relation into fragments so that most of the applications run in a single fragment. Fragmentation is used in centralized and Distributed Database Management Systems (DBMSs) and serves as a design tool that enables user queries to operate on smaller relations [1]. Three main fragmentation techniques are defined for relational databases:

1. Horizontal fragmentation (HF);
2. Vertical fragmentation (VF);
3. Hybrid or mixed fragmentation (MF).

2.3. Horizontal Fragmentation (Primary and Derived)

Horizontal partitioning splits a relation across its tuples. Each fragment is a subset of tuples in that relation; however, there are two versions of horizontal fragmentation: primary and derived [1].

1. **Primary horizontal fragmentation** is performed in a relation using the predicates that are defined in that relation.
2. **Derived horizontal fragmentation** is the partitioning of a relation as a result of predicates that are defined in another relation.

Horizontal fragmentation uses the select operator, where the predicates determine the fragmentation.

2.4. Vertical Fragmentation

Vertical fragmentation consists of dividing a relation into subsets of attributes and, in each fragment, placing the primary key of the relation to allow for its reconstruction. Vertical fragmentation is performed through the project operator [1].

2.5. Hybrid Fragmentation

In many cases, the horizontal or vertical fragmentation of a database schema is not sufficient to satisfy the application requirements. In these cases, vertical fragmentation is carried out after a horizontal one or vice versa, producing a structured tree of partitions. This technique is called hybrid fragmentation, mixed fragmentation, or nested fragmentation [1]. These two methods are described below:

1. **Horizontal–vertical hybrid:** First, it generates a set of horizontal fragments; vertical partitions are then obtained from one or more horizontal shards.
2. **Vertical–horizontal hybrid:** It produces a set of vertical fragments first; horizontal shards are then generated based on one or more vertical partitions.

2.6. Fragmentation Rules

Three fragmentation rules ensure that databases do not experience semantic changes during fragmentation [1].

1. **Completeness:** If a relation instance R is divided into fragments $F_R = \{R_1, R_2, \ldots, R_n\}$, then every element that is in R is also found in one or more fragments. This property is identical to the lossless decomposition property of normalization, and it is also important in fragmentation, as it ensures that the data in a global relation is mapped onto fragments without any loss. Note that in the case of horizontal partitioning, the "element" generally refers to a tuple, while in the case of vertical fragmentation, it refers to an attribute.
2. **Reconstruction:** If a relation R is split into fragments $F_R = \{R_1, R_2, \ldots, R_n\}$, it is possible to define a relational operator ∇ such that $R = \nabla R_i, \forall R_i \in F_R$.

The operator ∇ will be different for distinct forms of fragmentation; i.e., for horizontal fragmentation this operator is the union, and in vertical partitioning it is the join operator. Reconstructing the relation from its fragments ensures that the constraints defined for the data in the form of dependencies are preserved.

3. **Disjointness:** If a relation R is horizontally decomposed into fragments $F_R = \{R_1, R_2, \ldots, R_n\}$ and the data d_i is in R_j, then it is not in any other partition R_k ($k \neq j$). This criterion ensures that the horizontal fragments are disjoint. If the relation R is vertically decomposed, its primary key attributes are typically repeated across all its fragments (for reconstruction). Therefore, in the case of vertical fragmentation, disjunction is defined only on the non-primary key attributes of a relation.

2.7. Static Fragmentation vs. Dynamic Fragmentation

Most of the existing fragmentation techniques are static [4,6,7,15–24]. Static partitioning techniques are inefficient for large or highly dynamic systems because they only allow a database to be fragmented based on the workload. However, if it changes, these approaches cannot adapt the partitioning scheme to the new data access patterns. On the other hand, dynamic partitioning allows the fragmentation scheme to be updated over time according to changes in the workload [1]. Nevertheless, few works address dynamic hybrid fragmentation [8,25–29]. Other approaches are dynamic, but consider only horizontal or vertical partitioning [3,5,30–32].

3. Related Work

This section provides a comparative analysis of 24 works published between 2020 and 2026 on fragmentation methods for databases. The reviewed works are presented below, grouped into three categories: the first subsection includes the works that consider static fragmentation, then we discuss the approaches for dynamic fragmentation, and the last subsection encompasses multimedia data fragmentation methods.

3.1. Static Fragmentation

Zhou et al. [15] proposed a Graph Learning-Based Database Partitioning System (Grep) to perform horizontal fragmentation. In Grep, first, the graph model is used to encode the data and query features. Where the vertices are columns, the edges represent column relations, and the column weights are computed based on the graph structures in which

they are located. Subsequently, graph neural networks are considered to capture the data and query correlations, and a key selection model is then used to choose the appropriate partitioning keys based on the graph model.

The use of High-Performance Computing (HPC) applications has caused a growing interest in data fragmentation techniques and strategies; nevertheless, it is necessary to find the right size for the data blocks to accelerate intensive parallel applications and increase scalability. To solve this problem, in [16], a method for data block size estimation in HPC applications, based on supervised machine learning techniques, was proposed. This approach can determine a suitable way to partition a given data set, thus enabling the efficient execution of data-parallel applications in high-performance environments. However, this work only considered horizontal fragmentation.

The efficiency of Log-Structured Merge-tree (LSM-tree)-based systems depends mainly on the chosen compaction strategy, whether level- or stack-based. However, these approaches rely either on no fragmentation or on simplistic partitioning techniques, which limit parallelism and the feasibility of trivial moves. To solve this problem, Mao & Hristidis [17] focused on two horizontal partitioning algorithms: Local Range and Global Range, applicable to any stack-based strategy. These were compared under different workloads, and the results showed that Global Range significantly improved compaction performance with minimal additional overhead.

To ensure scalability, availability, and performance in a distributed database, Muhammad et al. [18] developed a new derived horizontal fragmentation method using a Modified Create, Read, Update, and Delete (MCRUD) matrix to optimize partition placement based on attribute locality and access frequency, thereby minimizing unnecessary data access and communication overhead between distributed nodes.

The previous works in this subsection focused on static horizontal fragmentation; in contrast, the following consider static hybrid partitioning. First, Badran et al. [19] proposed a hybrid fragmentation method based on data sensitivity to improve the security and performance of current partitioning techniques. Moreover, in their other work [21], they implemented a vertical–horizontal static hybrid fragmentation technique for a relational DBMS, such as Microsoft SQL, that protects sensitive data and improves query performance.

Then, Noraziah et al. [20] addressed the problems that schema design presents regarding wasting storage space and increasing the transaction response time when applying replication in databases with the Binary Vote Assignment on Grid Quorum with Association Rule (BVAGQ-AR) algorithm, which manages synchronous replication and transactions and can preserve the consistency of the fragmented schema in distributed databases.

Likewise, Azila et al. [22] provided an approach to address the problem that a distributed database has with preserving data reliability with low communication cost when there are partition replicas, because this data will be distributed evenly among the replica servers instead of storing all data on all servers.

In contrast, Ahmed & Alluhaibi [23] presented Genetic Killer Whale Optimization-based Clustering, which solved the data loss and imperfect clustering of large databases using horizontal–vertical fragmentation and an optimization-based clustering model. In addition, it included a cost model that allowed for the determination of the fragment attributes and instances to reduce computing and communication costs.

Schreiner et al. [24] proposed a hybrid fragmentation technique for NewSQL databases based on an extended heat graph, fulfilling two objectives: first, it partitions the graph while maintaining well-balanced graph partitions (local transactions equally distributed), and second, it minimize the edges between fragments (distributed transactions).

3.2. Dynamic Fragmentation

Ahmed & Alhenshiri [30] proposed an adaptive fragmentation framework based on query patterns to optimize distributed database performance. This approach considered horizontal and derived partition techniques according to frequently executed queries, followed by a genetic algorithm to reduce redundant predicates and minimize fragment growth. They used the Transaction Processing Performance Council-H (TPC-H) benchmark on PostgreSQL to implement the framework, and the results showed that the derived partition minimized the cost of distributed join operations and genetic algorithms, further reducing the number of fragments.

Singh & Choudhary [31] proposed improving the Security-awarE Data placement mechanism for cLOUd storage Systems (SEDuLOUS) [33], which provides a heuristic algorithm for data location in a cloud environment. They also addressed cloud data storage services and associated security issues, and analyzed SEDuLOUS to improve it by specifying the minimum fragments required to ensure partitioning of all files and assigning a hash to each partition to identify storage nodes.

Since vertical fragmentation techniques are commonly based on heuristic rules set by experts, they fail to allow for different fragment optimizations given the characteristics of a workload. Liu et al. [32] solved this problem with a method called spectral clustering-based vertical partitioning (SCVP). First, SCVP reduces the solution space by increasing the number of constraints, i.e., by generating initial partitions via spectral clustering. Subsequently, SCVP designs an algorithm to search the solution space; i.e., the initial partitions are optimized by combining frequent feature set extraction with greedy search. In addition, they developed a new method called special clustering-based vertical partitioning redesign (SCVP-R), which improved SCVP.

The three works discussed above contemplated dynamic horizontal or vertical partitioning; the rest of this subsection focuses on dynamic hybrid fragmentation approaches. In the case of Heterogeneous Hybrid Transactional Analytical Processing (H^2TAP), database systems are hard to construct and complex to administer; therefore, Pinnecke et al. [25] proposed GRIDTABLE, a highly flexible and adaptive data structure that makes it possible to physically organize dispersed-but-structured records in the context of H^2TAP for mixed query workloads, which provides vertical–horizontal partitioning.

Kulba & Somov [26] analyzed dynamic fragment allocation in distributed data processing systems. A heuristic algorithm was presented to place fragments as a function of each system's parameters over time. Two main methods of data fragmentation were used: horizontal and vertical fragmentation. The authors presented a method for dynamically redistributing table fragments among the nodes of a distributed system, taking into account the current values of system parameters, which change over time.

On the other hand, Kang et al. [27] solved the problem caused when queries repeatedly read a partition due to its irregular shape using Jigsaw, a technique for creating irregular partitions, which recursively splits a table until finer partitioning does not transfer less data for a given query workload, and then combines smaller segments into larger irregular partitions. In this way, partitions are accessed fewer times during query execution.

Cui et al. [28] focused on the design of a Workload-Aware Data Hybrid Partitioning Advisor with Deep Reinforcement Learning (WorthyPar) for Online Analytical Processing (OLAP) workloads, which considers flexibility and fine-grained partitioning based on adaptability. They demonstrated that the hybrid partitioning problem is NP-hard and formulated it as a Markov Decision Process (MDP) to train an automatic partitioning advisor. Their other work [29] is an extended WorthyPar, a data partition method for cloud–edge–device architectures. They developed FinePar, a fine-grained, dynamic hybrid (horizontal–vertical) partitioning framework based on Deep Reinforcement Learning that considers data allocation, aiming to reduce the data transfer volume and shorten the execution time.

3.3. Multimedia Data Fragmentation

Castro-Medina et al. [3] focused on the management of multimedia data. They proposed a dynamic horizontal method for content-based queries that can adapt to access patterns of multimedia databases to reduce costs and improve query response time performance.

To maintain robust performance in Distributed Multimedia Database Systems (DMDBSs), Mahmoud et al. [4] considered data allocation, replication, and clustering in their vertical fragmentation technique, using the silhouette analysis method and the K-means++ clustering algorithm. Their two-layer architecture includes a User Interface Client Layer, which provides a set of interfaces for the DBA to access the database from any location with minimal technological resources, and a Distributed Database System Layer responsible for the storage, retrieval, and processing of different types of multimedia data, which consists of seven methods. The results demonstrated that the approach outperforms both the K-means with refinement method and the Differential Bond Energy Algorithm, achieving higher fragmentation quality, better data locality, and lower execution time. Furthermore, Mahmoud et al. [5] presented the Reinforcement Learning-Based Fragment Allocation and Replication for Workload-Aware Multimedia Systems (RL-FAWM) architecture, a Q-learning framework that models fragment placement as an MDP. The architecture is designed to dynamically minimize the execution cost based on the query patterns and system constraints, and to jointly consider the read, manipulation, join, and capacity constraints to learn effective data placement strategies adaptively. The results demonstrated that RL-FAWM reduces replication costs by over 93% and allocation costs by over 30% compared to the Vertical Fragment Allocation and Replication heuristic.

In contrast to the previous works in this subsection that only considered horizontal or vertical partitioning, the following works contemplated hybrid fragmentation of multimedia databases. First, the authors of [6] proposed a text-based medical image indexing technique that uses the attributes of medical images and fragments them in a hybrid (horizontal–vertical) manner, allowing for these attribute fragments to be reshaped into a hierarchy and building a multidimensional index. Alternatively, Patel et al. [7] presented a workload- and storage-aware cost-based fragmentation technique. The cost model extracts each attribute from the raw data file and loads it into the database system to determine the extraction and load times and the attribute size.

In contrast, in our previous work [8] we showed the results of an exhaustive analysis of 34 hybrid fragmentation methods following the proposed research methodology. In [8] we also designed a cost model-based hybrid fragmentation method that adapts the fragmentation scheme according to changes in the workload to maintain efficient retrieval of multimedia data; therefore, that work motivated the objective of this new research, as the proposed hybrid fragmentation method is now developed.

Table 1 shows the works related to data fragmentation. We aim to provide a hybrid fragmentation method considering nine aspects: (1) horizontal fragmentation, (2) vertical partitioning, (3) completeness, (4) ease of implementation, (5) cost model, (6) dynamic fragmentation, (7) multimedia database, (8) query performance, and (9) allocation.

Table 1. Comparison of related works.

Article	1	2	3	4	5	6	7	8	9
Castro-Medina et al. [3]	X		X	X	X	X	X	X	X
Mahmoud et al. [4]		X	X	X			X	X	X
Mahmoud et al. [5]		X	X		X	X	X	X	X
Safaei [6]	X	X	X	X			X	X	
Patel et al. [7]	X	X	X	X	X		X	X	
Crescencio-Rico et al. [8]	X	X		X	X	X	X	X	X

Table 1. *Cont.*

Article	1	2	3	4	5	6	7	8	9
Zhou et al. [15]	X		X	X	X			X	
Cantini et al. [16]	X							X	
Mao & Hristidis [17]	X		X						
Muhammad et al. [18]	X			X	X			X	X
Badran et al. [19]	X	X							
Noraziah et al. [20]	X	X	X					X	
Badran et al. [21]	X	X	X	X					
Azila et al. [22]	X	X		X				X	
Ahmed & Alluhaibi [23]	X	X	X		X			X	
Schreiner et al. [24]	X	X	X		X			X	
Pinnecke et al. [25]	X	X			X	X		X	
Kulba & Somov [26]	X	X			X	X		X	X
Kang et al. [27]	X	X	X		X	X		X	
Cui et al. [28]	X	X	X			X		X	
Cui et al. [29]	X	X	X			X		X	X
Ahmed & Alhenshiri [30]	X			X		X		X	
Singh & Choudhary [31]		X			X	X			X
Liu et al. [32]		X	X	X	X	X		X	
This work	X	X	X	X	X	X	X	X	X

(1) Horizontal fragmentation, (2) Vertical fragmentation, (3) Completeness, (4) Ease of implementation, (5) Cost model, (6) Dynamic fragmentation, (7) Multimedia database, (8) Query performance, and (9) Allocation.

As shown in Table 1, 58% of the works focus on hybrid fragmentation [6–8,19–29]. Nevertheless, only six hybrid fragmentation approaches consider dynamic fragmentation [8,25–29], and just one of them takes multimedia data into account [8], but it only presents the design of its hybrid fragmentation technique. In contrast, Mahmoud et al. [4], Safaei [6], and Patel et al. [7] also contemplate multimedia data, but they do not perform dynamic fragmentation. Moreover, Castro-Medina et al. [3] and Mahmoud et al. [5] consider multimedia data and dynamic fragmentation, but not hybrid partitioning.

In this paper, we extend our previous work proposed in [8], now presenting a complete and easy-to-implement hybrid fragmentation method. That is, the article presents all the necessary information to implement the method, i.e., a process diagram and the physical database diagram, algorithms, and settings (database, sites, computer equipment, and workloads); and this approach does not require specialized knowledge (e.g., machine learning) for its application. Additionally, it is developed for multimedia databases, based on a cost model to determine the partitioning schemes that reduce the execution cost of database operations, and it is capable of adapting the fragmentation scheme according to changes in database access patterns. Finally, our method allocates fragments to sites; therefore, in contrast to the related works in Table 1, it complies with the nine comparison criteria.

4. Dynamic Hybrid Fragmentation for Multimedia Databases

The proposed method has two phases: static hybrid fragmentation and dynamic hybrid fragmentation.

In the first phase, a hybrid fragmentation scheme is obtained through the XAMANA web application [3]. This web application was developed using the Java programming language with the JavaServer Faces framework [34,35]. XAMANA allows the three types of fragmentation to be applied—horizontal, vertical, and hybrid—considering Content-Based Image Retrieval (CBIR), including MySQL, PostgreSQL, Postgres-XL, and MongoDB database management systems.

The information the DBA enters into XAMANA is stored in its database because it is required to perform dynamic hybrid fragmentation. The XAMANA database is relational and was developed with the MySQL DBMS. It comprises 10 tables (Database (DB), Table, Attribute, Fragment, CRUD, Site, Predicate, CRUD_Attribute, Predicate_Table, and CRUD_Predicate). Figure 1 shows the process diagram for static hybrid fragmentation. The physical scheme of XAMANA is presented in Figure 2, and Table 2 includes the explanation of each relation in Figure 2.

Table 2. Description of each table in the physical diagram of the XAMANA database.

Table	Description
Table	It stores the data of the fragmented table, such as its name, the type of fragmentation performed, and its operation and performance thresholds.
DB	It contains the information (database name, IP address, port, DBMS, user, and password) on the connection to the database on which dynamic hybrid partitioning is performed. The password is optional if the DBMS allows disabling authentication.
Attribute	It includes the name, type, and size of the fragmented table attributes.
Fragment	When fragmentation is applied, it saves the information from the fragments resulting from either the static or dynamic approach; primarily, the number of fragment operations and performance are recorded to determine the operation and performance thresholds.
Site	It contains the IP address of all network sites used in the log file.
CRUD	It stores any logged operation in the database query history file. This table contains the type of operation (read, create, update, or delete), the cost of the operation, and whether it is considered CBIR.
Predicate	It includes all predicates of each operation related to the fragmented and allocated table. The predicates are largely occupied with horizontal fragmentation. With this table, the MCRUD matrix is constructed.
CRUD_Predicate	It is the relationship between the predicate and CRUD (Create, Read, Update, and Delete) tables, since an operation can have many predicates and a predicate sometimes appears in many operations.
CRUD_Attribute	It relates all the attributes used in each CRUD operation; in this way, a single attribute can be used in many CRUD operations, and one CRUD operation can use many attributes.
Predicate_Table	It relates the operation predicates to the site and table identifiers.

The first action in XAMANA begins with a form where the DBA enters the DBMS to be used, as well as the database IP address, DB port, user, password, log file, table to fragment, operation and performance thresholds, and the type of fragmentation (horizontal, vertical, and hybrid). The method proposed in this research is implemented in the MongoDB DBMS. In this DBMS, size limits are not established for attributes, as occurs in relational databases; it only defines the maximum size of each document (16 MB). However, in vertical and hybrid fragmentation, the use of attributes in operations is considered; therefore, it is essential to assign the maximum size to each of them.

Once the DBA has finished capturing the data, the next action is the connection test. If the data is correct, as a second step the application obtains the vertical fragmentation scheme based on the method proposed by Castro-Medina et al. [36] and the horizontal scheme according to the technique of Castro-Medina et al. [3], with their respective allocation of fragments to the network sites. Subsequently, XAMANA produces two hybrid fragmentation schemes: (1) vertical–horizontal, which considers the vertical allocations, and (2) horizontal–vertical, which contemplates the horizontal allocations. Both hybrid schemes are compared, and the lowest-cost scheme is provided to the DBA, who can decide if XAMANA materializes the fragmentation and allocation scheme.

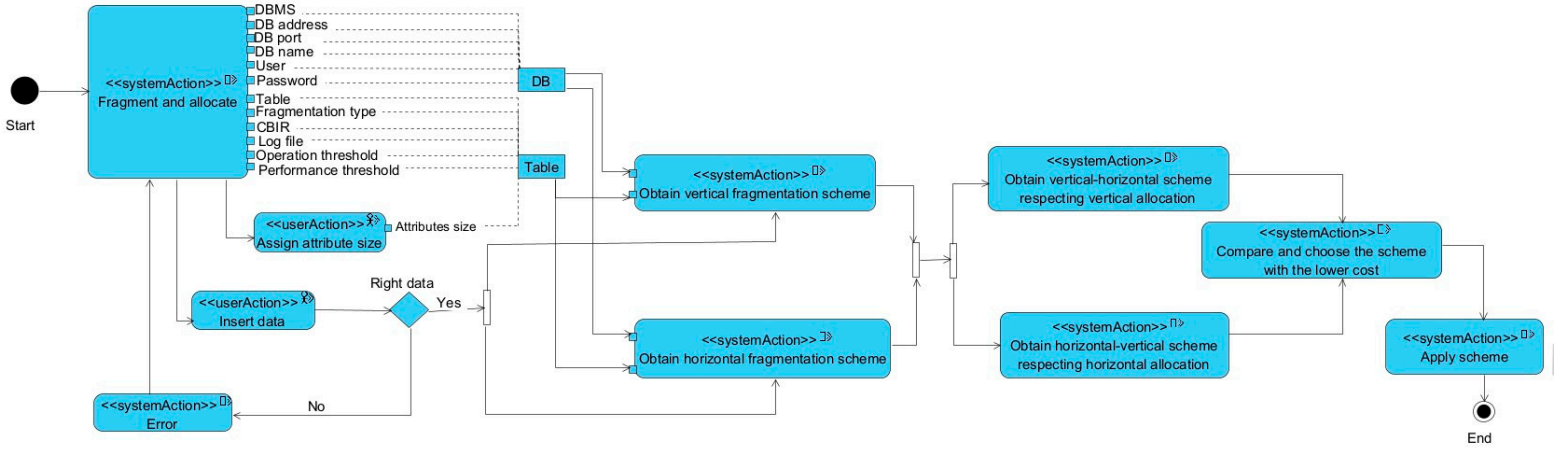


Figure 1. Process diagram of static hybrid fragmentation.

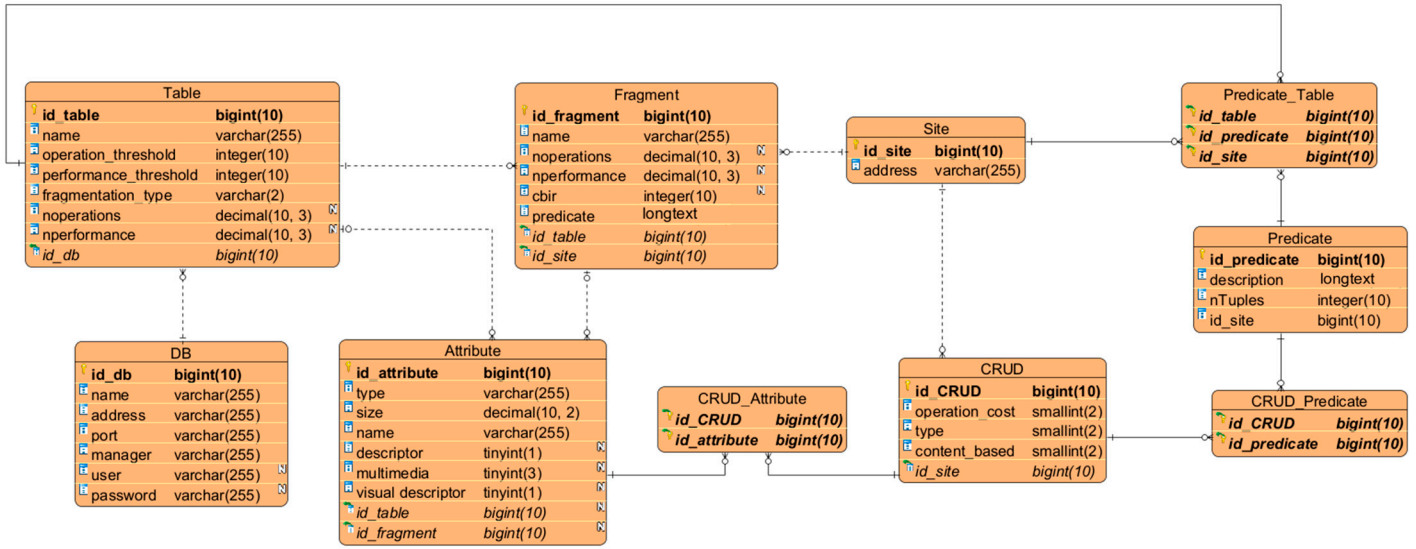


Figure 2. Physical diagram of the XAMANA database.

During horizontal and vertical fragmentation, according to the type of operation, a representative value is assigned, as presented in Table 3. These values are used to calculate the cost of an operation in the cost analysis. Likewise, it is important to consider the site where the operation is executed. If the operation is performed on the same site where all its required data is located, the *RV* (Remote Value) is 1, and it is 2 when it accesses remote data.

Table 3. Operation assigned values (OAVs).

Type of Operation	Value
Create	2
Delete	2
Update	3
Read	1

Equation (1) is used to obtain the vertical cost (VER) of an operation q_j, where *JC* is the join cost, which takes the value of *JN* + 1, where *JN* is the number of joins required to execute the operation. In addition, AS is the size in MB of each attribute used in the operation. Therefore, the cost of a vertical fragment (VFC) vfr_k is calculated by adding the costs of the attributes (ACs) located in that fragment, as seen in Equations (2) and (3). *N* is the number of operations, *M* is the number of attributes, and *MQ* is the number of attributes used in the operation q_j.

$$VER\ (q_j) = OAV_j \times JC_j \times RV_j \times \sum_{i=1}^{MQ} AS_i \tag{1}$$

$$AC\ (a_i) = \sum_{j=1}^{N} (OAV_j \times JC_j \times RV_j) \times AS_i \tag{2}$$

$$VFC\ (vfr_k) = \sum_{i=1|a_i\ \in vfr_k}^{M} AC(a_i) \tag{3}$$

Equation (4) determines the horizontal cost (HOR) of an operation q_j, where SEL is the selectivity of its predicate. While in horizontal fragmentation, the Attribute Locality Precedence (ALP) [37] of an attribute a_i is determined by the sum of the horizontal cost (HOR) of the operations that use that attribute, as shown in (5).

$$HOR(q_j) = OAV_j \times SEL_j \times RV_j \tag{4}$$

$$ALP(a_i) = \sum_{j=1|\ use(q_j,a_i)=1}^{N} HOR(q_j) \tag{5}$$

The cost of a horizontal fragment (HFC) hfr_l is obtained by adding the horizontal cost of the operations with the same predicate p_j that defines the fragment, as observed in Equation (6).

$$HFC\ (hfr_l) = \sum_{j=1|p_j=p_l}^{N} HOR(q_j) \tag{6}$$

Equation (7) indicates that the hybrid cost (HYB) of an operation q_j is calculated by adding its vertical cost plus its horizontal cost.

$$HYB(q_j) = VER(q_j) + HOR(q_j) \tag{7}$$

The cost of a hybrid fragment $hyfr_t$ is determined by taking into account the allocation of the vertical and horizontal schemes; therefore, there are two costs: *V-H* (8) and *H-V* (9), which are obtained by adding the costs of the operations that are executed in that fragment. VER_v and VER_h refer to the vertical cost of the operation based on the allocation of the vertical or horizontal scheme, respectively, while HOR_v and HOR_h indicate the horizontal cost of the operation considering each allocation (vertical or horizontal).

$$V - H\ (hyfr_t) = \sum_{j=1|use\ (q_j,\ hyfr_t)=1}^{N} VER_v(q_j) + HOR_v(q_j) \tag{8}$$

$$H - V\ (hyfr_t) = \sum_{j=1|use\ (q_j,\ hyfr_t)=1}^{N} VER_h(q_j) + HOR_h(q_j) \tag{9}$$

Likewise, to calculate the cost of the hybrid fragmentation scheme *HYFS*, the allocation of the vertical and horizontal schemes must be considered; therefore, there are two final costs: the vertical–horizontal (V-H) and the horizontal–vertical (H-V) costs. These are obtained with Equations (10) and (11).

$$V - H\ (HYFS) = \sum_{j=1}^{N} VER_v(q_j) + HOR_v(q_j) \tag{10}$$

$$H - V\ (HYFS) = \sum_{j=1}^{N} VER_h(q_j) + HOR_h(q_j) \tag{11}$$

The cost of performing a new fragmentation (NF) of the HYFS is determined by using Equation (12). The number of documents (ND) transported from the original site to a remote site in each fragment is multiplied by the size of the attributes in each document. *F* is the number of fragments in the HYFS, and *A* is the number of attributes in each document.

$$NF(HYFS) = \sum_{j=1}^{F} ND_j \times \sum_{i=1}^{A} AS_i \tag{12}$$

As mentioned at the beginning of this work, static fragmentation is not capable of adapting a scheme to new workloads; therefore, the approach in this research mainly focuses on dynamic hybrid fragmentation using the observer–fragmenter, which is an application developed in the Java programming language. Next, the proposed technique is explained through the process diagram in Figure 3.

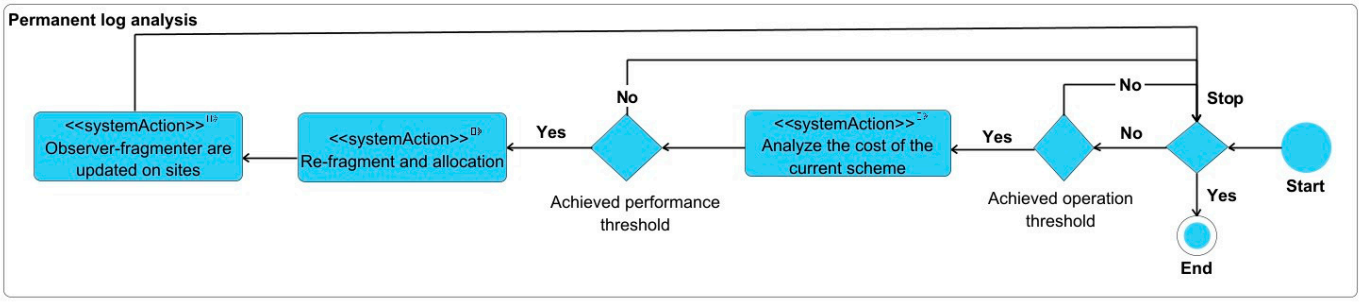


Figure 3. Process diagram of the dynamic hybrid fragmentation.

Once XAMANA materializes the hybrid fragmentation scheme, it provides a token (the identifier of the database that the DBA wants to partition). The DBA executes the observer–fragmenter at each site and enters the token, so that the observer–fragmenter of

each site starts with the permanent log analysis, which stops when the DBA considers it necessary. For this, the following steps are performed.

1. The operation threshold is evaluated with an initial value that the DBA considers appropriate. If the threshold is not reached, no changes are made; otherwise, step two begins.
2. The necessary operations to obtain the costs of the current scheme are carried out, and the observer–fragmenter continues with step three.
3. The performance threshold is evaluated; if this threshold is not achieved by the results obtained from the cost analysis, no change is made; otherwise, step four is initiated.
4. A new fragmentation and allocation of fragments are performed.
5. Finally, the observer–fragmenter of each site is updated with the new scheme, and the first step begins again.

The observer–fragmenter is based on Algorithm 1, the DHYF (dynamic hybrid fragmentation) observer, and works as follows: The algorithm receives, as its parameters, the token returned by the XAMANA Web application when hybrid static fragmentation is executed and the DBMS log files. In addition, it returns *DHYFS[]*, which is the set of hybrid fragmentation schemes, one for each fragment that surpasses both the operation and performance thresholds. In lines 7–8, the number of operations required to trigger a new performance analysis (nOperations) is calculated based on the number of operations recorded on the fragment at the time of its last iteration (initial_operations) and the operation threshold (the percentage set by the DBA in XAMANA during static hybrid partitioning); then, in lines 9–10, the fragments that do not exceed the operation threshold are discarded from the list of fragments to analyze.

Algorithm 1 uses Algorithm 2, *performance_analysis*, which receives as its parameters all the fragments, the workload, the fragments that achieve the operation threshold (*thresholdF*[]), and the size of the attributes (*AS*[]), to subsequently obtain the hybrid cost of each operation. It returns the list of the fragments that exceed the performance threshold (*thresholdP[]*).

Algorithm 2 starts by looping through each line of the *log* file to calculate the hybrid cost (Equation (7)) of each operation performed on the fragments, where *current* is the position of the line being read in the *log* file (line 4). For each operation, the IP address of the site (IpLine) where it was executed and its OAV are obtained (lines 5 and 6). Subsequently, the hybrid cost of each operation that accesses the fragment that achieved the operation threshold is obtained. For this purpose, the attributes of the operation being analyzed are recovered and stored in the *attributesOpe[]* list. To obtain the join cost (JC), two lists are required: the fragment lists containing the attributes of the operation (*fragmentsResolveAttributes*[]) and those that resolve the predicates of the same (*fragmentsResolvePredicates[]*) (lines 7–31).

The remote value (RV) is obtained by comparing the IP address of the operation in the *log* file with those of the sites allocating the fragments that solve its predicate to determine if the operation is local or remote. Next, the vertical (VER), horizontal (HOR), and hybrid (HYB) costs of the operations are calculated according to Equations (1), (4) and (7) (lines 31–44).

Finally, the hybrid costs (hybridCost) are assigned to the fragments, the value necessary to achieve the performance threshold (performance) is calculated, and the fragments that meet this criterion (*thresholdP[]*) are returned to Algorithm 1 (lines 45–63).

For each fragment that achieves the performance threshold, Algorithm 1 invokes Algorithm 3, *dynamic_hybrid_fragmentation* (line 15). The latter receives as its parameters a fragment and the workload (*log*[]). Subsequently, in lines 3 and 4, the vertical and horizontal fragmentation schemes are produced according to the techniques proposed by

Castro Medina et al. [3,36]. From the vertical and horizontal partition schemes, the hybrid scheme is produced (*HYFS[]*), assigning to each hybrid fragment its number (nFragment), name, predicate, attributes, and horizontal (hSite) and vertical sites (vSite), as presented in lines 7–18. Afterward, Algorithm 6 (hybrid performance cost) from Castro Medina et al. [36] is used to obtain the costs of the hybrid fragmentation scheme and return the new hybrid scheme generated from each fragment to Algorithm 1.

Algorithm 1 DHYF Observer

```
1  input: token, log[]
2  output: DHYFS[]
3  fragments[] ← token.get_fragments
4  AS[]        ← token.get_attribute_sizes
5  thresholdF[] ← fragments[]
6  for i from 0 to size of fragments[]-1 do {
7      nOperations ← (fragments[i].operations_threshold *
8                     fragments[i].initial_operations)/100
9      if fragments[i].new_operations < nOperations then
10         thresholdF[]← thresholdF[] - fragments[i]
11 }
12 thresholdP[]← performance_analysis(fragments[], log[], thresholdF[], AS[])
13 DHYFS[]← empty list
14 for i from 0 to size of thresholdP[]-1 do {
15     DHYFS[i]← dynamic_hybrid_fragmentation(thresholdP[i], log[])
16 }
17 return DHYFS[]
```

Algorithm 2 performance_analysis

```
1  input: fragments[], log[], thresholdF[], AS[]
2  output: thresholdP[]
3  for i from 0 to size of log[]-1 do {
4      current ← log[i]
5      IpLine ← get the site IP where the current operation was executed
6      OAV ← 1 if it is a read, 2 if it is a creation or deletion, and 3 if it is an update
7      if current contains a fragment name of thresholdF[] then {
8          attributesOpe[] ← find the attributes used in the current operation
9          fragmentsResolveAttributes[] ← empty list
10         for each fragment ← fragments[] do {
11             use ← false
12             for each l ← attributes of fragment do {
13                 if attributesOpe[] contains l then
14                     use← true
15             }
16             if use = true then
17                 add fragment to fragmentsResolveAttributes[]
18         }
19         predicate ← get predicate of the current operation
20         SEL ← get the selectivity of the predicate
21         fragmentsResolvePredicates[] ← empty list
22         if SEL > 0 then {
```

Algorithm 2 *Cont.*

```
                    for each fragment ← fragmentsResolveAttributes[] do {
                        tuples ← number of documents satisfying both the operation
                            predicate and fragment predicate
                        if tuples > 0 then
                            add the fragment to fragmentsResolvePredicates[]
                    }
                }
                JC ← size of fragmentsResolvePredicates[]
                RV ← 1
                exist ← false
                for each fragment ← fragmentsResolvePredicates[] do {
                    if the site of the fragment != IpLine then
                        exist ← true
                }
                if exist = true then
                    RV ← RV * 2
                VER ← 0
                for each a ← attributesOpe[] do {
                    VER ← VER + (OAV * JC * RV * AS[a])
                }
                HOR ← OAV * SEL * RV
                HYB ← HOR + VER
                for each fragment ← thresholdF[] do {
                    b ← false
                    for each frag ← fragmentsResolvePredicates[] do {
                        if fragment.name = frag.name then
                            b ← true
                    }
                    if b = true then
                        fragment.hybridCost ← fragment.hybridCost + HYB
                }
            }
        }
        thresholdP[] ← thresholdF[]
        for j from 0 to size of thresholdF[]-1 do {
            performance ← (thresholdF[j].performance_threshold *
                thresholdF[j].initial_performance)/100
            if thresholdF[j].hybridCost < performance then
                thresholdP[] ← thresholdP[]-thresholdF[j]
        }
        return thresholdP[]
```

Algorithm 3 dynamic_hybrid_fragmentation

```
input: fragment, log[]
output: HYFS[]
VFS[] ← vertical_fragmentation (fragment, log[])
HFS[] ← horizontal_fragmentation (fragment, log[])
HYFS[] ← empty list
```

Algorithm 3 *Cont.*

```
nFragment ← 1
        for each hFragment ← HFS[] do {
            for each vFragment ← VFS[] do {
                hybridFragment ← object fragment
                hybridFragment.name ← hFragment.name + nFragment
                hybridFragment.predicate ← hFragment.predicate
                hybridFragment.attributes[] ← vFragment.attributes
                hybridFragment.hSite ← hFragment.site
                hybridFragment.vSite ← vFragment.site
                add hybridFragment to HYFS[]
                nFragment ← nFragment + 1
            }
        }
        HYFS[] ← Hybrid performance cost (log[], HYFS[])
return HYFS[]
```

5. Results

In this section, we present how the hybrid fragmentation process is carried out and demonstrate the correct operation of our proposed approach through two case studies. The first case describes the process for static hybrid fragmentation, and the second details the execution of the observer–fragmenter (dynamic hybrid fragmentation). The database, sites, and computer equipment are presented below.

The HITO database [38] is used to apply static and dynamic hybrid fragmentation. HITO is a centralized database developed in MongoDB. HITO stores the history of the Orizaba Technological Institute through images and videos of its different events, students, teaching staff, buildings, and equipment. Currently, it consists of 19 collections with a total of 2.53 GB of storage.

HITO contains the *multimedia_records* collection with 1870 documents; it stores images of events, buildings, equipment, and personnel of the Orizaba Technological Institute. As seen in Table 4, each document has a unique identifier, image name, description, image, whether it is valid, the SURF (Speeded Up Robust Features) descriptor of the image, and its type. *multimedia_records* is used to apply the three types of fragmentation (horizontal, vertical, and hybrid), as well as to measure the performance of our proposed approach. Vertical fragmentation considers attributes; therefore, Table 4 details each attribute and its size in the *multimedia_records* collection.

Table 4. Attributes of the *multimedia_records* table.

Name of Attribute	Datatype	Size in MB (AS)
_id	String	0.01
nombre	String	0.01
descripcion	String	0.01
imagen	String	0.05
validado	String	0.01
descriptor	Array	0.03
tipo	String	0.01

Additionally, two sites are used: the first site (192.168.1.63) is where HITO is located, with 19 collections including *multimedia_records*; it has the Windows 11 Home Single Language operating system, an AMD Ryzen 5 5600H processor with Radeon Graphics,

3.30 GHz frequency, and 8 GB of RAM. In addition, the IP address 192.168.1.75 is assigned to the second site, which has a Windows 10 Pro operating system, an AMD A9-9425 RADEON R5 processor, 4 GB of RAM, and a frequency of 3.10 GHz.

5.1. Case Study of Hybrid Fragmentation

The workloads used in the case studies are obtained through SimulQUERY, an operations simulator developed by Castro-Medina et al. [36] for MongoDB. Table 5 shows the workload for *multimedia_records*, with ten operations used in the first case study, which includes the four types of operations (*find*, *delete*, *insert*, and *update*). In the "Site" column, 1 or 2 indicates that the operation comes from the site with IP 192.168.1.63 or the site with IP 192.168.1.75, respectively.

Table 5. Operations performed on the centralized multimedia database.

Query	Operation	Site
q_1	db.multimedia_records.insertOne({"nombre":"conf2.1"})	1
q_2	db.multimedia_records.find({"tipo":{$eq:"personal"}}, {{"_id":1,"nombre":1})	1
q_3	db.multimedia_records.deleteMany({"tipo":{$eq:"personal"}})	1
q_4	db.multimedia_records.find({"tipo":{$ne:"personal"}}, {"validado":1, "_id":1, "nombre":1, "tipo":1, "descripcion":1})	2
q_5	db.multimedia_records.updateMany({"tipo":{$eq:"event"}},{"$set":{"tipo":"personal"}})	2
q_6	db.multimedia_records.insertOne({"nombre":"LabIng06"})	2
q_7	db.multimedia_records.updateMany({"nombre":{$eq:"MATGUTMA"}},{"$set":{"nombre":{$eq:"MATGUTMA"}},{"$set":{"nombre":"VEGHERGE"}})	1
q_8	db.multimedia_records.deleteMany({"validado":{$eq:null}})	2
q_9	db.multimedia_records.find({"nombre":{$ne:"13MAYO95"}},{"descripcion":1})	2
q_{10}	db.multimedia_records.deleteMany({"nombre":{$eq:"DN198xxxxx-129(JMH)"}})	1

When running XAMANA, MongoDB is selected as the DBMS, and the database information (*DB address*, *DB port*, *DB name*, *user*, and *password*) is entered as shown in Figure 1; the username and password are optional if the administrator has disabled login authentication. When the *DB name* is entered, the collections found in it are automatically displayed; therefore, the DBA selects the table to fragment. Subsequently, the configuration for vertical fragmentation is carried out.

The vertical and hybrid fragmentation cost model requires the size of attributes in MB (AS); therefore, the DBA must assign the maximum size for each field in the collection. The size of each attribute for *multimedia_records* is presented in Table 4.

Following the process diagram in Figure 1, the next option is *CBIR*; this alternative is only considered when the table or collection receives content-based queries through multimedia data descriptors. This work does not consider CBIR. Then, the workload file is uploaded (*log*); subsequently, in XAMANA, the values of the operation and performance thresholds (ranging from 0 to 100) are assigned by the DBA; the smaller these values are, the more frequent the fragmentations are. Then, the DBA proceeds to obtain the scheme.

The vertical fragmentation scheme is obtained based on Algorithm 4 and Algorithm 5 of Castro Medina et al. [36]; therefore, the costs of the database operations (Table 5) are first calculated based on Equation (1), as shown in Table 6. For instance, in the case of query q_4, the operation is a *find* type and uses the attributes: *descripcion*, *tipo*, *validado*, *_id*, *and nombre*. Therefore, the sum of their sizes is obtained; the sizes of each attribute are taken from Table 4. The remote value (RV_4) is 2 because the operation is executed at site 2 and

the database is located at site 1. Also, the join cost (JC_4) takes the value of 1 because all attributes are present in the collection, so no join is required for this operation.

Table 6. Vertical cost for each operation in the workload.

Query	OAV_j	JC_j	RV_j	Attributes	AS_i	$VER(q_j)$
q_1	2	1	1	*nombre*	0.01	$2 \times 1 \times 1 \times 0.01 = 0.02$
q_2	1	1	1	*tipo, _id, nombre*	0.03	$1 \times 1 \times 1 \times 0.03 = 0.03$
q_3	2	1	1	*tipo*	0.01	$2 \times 1 \times 1 \times 0.01 = 0.02$
q_4	1	1	2	*descripcion, tipo, validado, _id, nombre*	0.05	$1 \times 1 \times 2 \times 0.05 = 0.1$
q_5	3	1	2	*tipo*	0.01	$3 \times 1 \times 2 \times 0.01 = 0.06$
q_6	2	1	2	*nombre*	0.01	$2 \times 1 \times 2 \times 0.01 = 0.04$
q_7	3	1	1	*nombre*	0.01	$3 \times 1 \times 1 \times 0.01 = 0.03$
q_8	2	1	2	*validado*	0.01	$2 \times 1 \times 2 \times 0.01 = 0.04$
q_9	1	1	2	*descripcion, nombre*	0.02	$1 \times 1 \times 2 \times 0.02 = 0.04$
q_{10}	2	1	1	*nombre*	0.01	$2 \times 1 \times 1 \times 0.01 = 0.02$

The operations carried out in the database use sets of attributes; therefore, attribute affinity must be considered, which indicates the relationship among table columns regarding how they are accessed together by the operations. Therefore, according to Algorithm 5 (Affinity Algorithm) [36], the second step of vertical fragmentation is the creation of the attribute cost per site matrix, where each set of attributes used in each workload operation forms a candidate fragment (CF).

To obtain the candidate fragment costs, Equation (13) is used. Table 7 presents the candidate fragments of each operation, which form the attribute cost per site matrix.

$$VFCC\ (vfr_k) = \begin{cases} \sum_{i=1 | a_i \in\ vfr_k}^{M} if\ AC(a_i) > 0 \\ 0 otherwise \end{cases} \quad (13)$$

Table 7. Attributes cost per site matrix.

CF	Query	Attributes	Site 1 Cost	Site 2 Cost	Total Cost
1	q_1	*nombre, _id*	0.09	0.1	0.19
2	q_2	*tipo, _id, nombre*	0.12	0.18	0.3
3	q_3	*tipo, _id*	0.04	0.1	0.14
4	q_4	*descripcion, tipo, validado, _id, nombre*	0	0.28	0.28
5	q_5	*tipo, _id*	0.04	0.1	0.14
6	q_6	*nombre, _id*	0.09	0.1	0.19
7	q_7	*nombre, _id*	0.09	0.1	0.19
8	q_8	*validado, _id*	0	0.08	0.08
9	q_9	*descripcion, nombre, _id*	0	0.14	0.14
10	q_{10}	*nombre, _id*	0.09	0.1	0.19

The candidate fragments of the queries q_1, q_3, *and* $q_5 - q_{10}$ do not have a document identifier, and so it is added. Furthermore, as seen in Table 7, the total cost is obtained by adding the cost per site for each candidate fragment. For example, candidate fragment 2 requires the attributes *tipo*, *_id*, and *nombre*; therefore, the execution cost of the vertical fragment is obtained using Equation (13) ($VFCC\ (vfr_2)$), adding the costs of the attributes present in that candidate fragment: $VFCC\ (vfr_2) = AC(tipo) + AC(_id) + AC(nombre)$. These attributes are accessed at site 1 as follows: *tipo* in operations q_2 and q_3; *_id* in q_2, and *nombre*

in q_1, q_2, q_7, and q_{10}. Thus, $AC(tipo) = (OAV_2 \times JC_2 \times RV_2) + (OAV_3 \times JC_3 \times RV_3) \times AS_{tipo}$ = [(1 × 1 × 1) + (2 × 1 × 1)] × 0.01 = 0.03; $AC(_id) = 0.01$; and $AC(nombre) = 0.08$. Thus, for candidate fragment 2 at site 1, $VFCC$ (vfr_2) = 0.03 + 0.01 + 0.08 = 0.12, and at site 2, $VFCC$ (vfr_2) = 0.08 + 0.02 + 0.08 = 0.18.

In the case of candidate fragment 4, $VFCC$ (vfr_4) = $AC(descripcion)$ + $AC(tipo)$ + $AC(validado)$ + $AC(_id)$ + $AC(nombre)$. Since $AC(descripcion)$ = 0 at site 1, the cost of this candidate fragment is also 0 at site 1 according to Equation (13); this is done to avoid assigning a higher cost to fragments that are not used together at a site.

Once the attribute cost per site matrix (Table 7) is obtained, the candidate fragments are arranged in descending order based on their final cost. The candidate fragments that contain repeated attributes are removed, and a candidate fragment is added with the attributes that were not used in the workload or not located in any candidate fragment, thus forming the final candidate fragments, as seen in Table 8. Finally, the vertical fragmentation scheme obtained with XAMANA is presented in Table 9.

Table 8. Candidate fragment.

CF	Attributes	$VFCC$ (vfr_k)
2	*tipo, _id, nombre*	0.3
8	*validado, _id*	0.08
11	*_id, descripcion, imagen, descriptor*	0.07

Table 9. Vertical fragmentation scheme for the case study.

Fragment Name	Site	Attributes	$VFCC$ (vfr_k)
multimedia_records_1	192.168.1.75	*descripcion, imagen, descriptor, _id*	0.07
multimedia_records_2	192.168.1.75	*validado, _id*	0.08
multimedia_records_3	192.168.1.75	*tipo, nombre, _id*	0.3

In the case of horizontal fragmentation, the database operations are first analyzed considering their predicates, calculating the horizontal cost of each query to create the MCRUD. The horizontal cost of an operation is obtained using Equation (4). For example, the operation q_3 is an elimination; RV_3 = 1 because it is carried out at the site where HITO is located. According to the values assigned in Table 3, OAV_3 = 2 and the selectivity of the predicate {"tipo":{$eq:"personal"}} is 874 documents found over the entire collection "*multimedia_records*" in HITO; therefore, $HOR(q_3) = OAV_3 \times SEL_3 \times RV_3 = 2 \times 874 \times 1 = 1748$. Table 10 shows the result of the MCRUD with the rest of the operations.

Table 10. MCRUD obtained with XAMANA.

Query	Predicate	Site	OAV	SEL	$HOR(q_j)$
q_1	{"nombre":"conf2.1"}	192.168.1.63	2	1	2
q_2	{"tipo":{"$eq":"personal"}}	192.168.1.63	1	874	874
q_3	{"tipo":{"$eq":"personal"}}	192.168.1.63	2	874	1748
q_4	{"tipo":{"$ne":"personal"}}	192.168.1.75	1	996	1992
q_5	{"tipo":{"$eq":"event"}}	192.168.1.75	3	724	4344
q_6	{"nombre":"LabIng06"}	192.168.1.75	2	1	4
q_7	{"nombre":{"$eq":"MATGUTMA"}}	192.168.1.63	3	2	6
q_8	{"validado":{"$eq":null}}	192.168.1.75	2	1843	7372
q_9	{"nombre":{"$ne":"13MAYO95"}}	192.168.1.75	1	1869	3738
q_{10}	{"nombre":{"$eq":"DN198xxxxx-129(JMH)"}}	192.168.1.63	2	1	2

As a second step, the ALP table is created using Equation (5). The locality precedence of an attribute is the sum of the horizontal costs of queries that use that attribute, $HOR(q_j)$. In the ten queries carried out, the attributes present in the predicates are *tipo*, *nombre*, and *validado*. The attribute *tipo* is used in four operations (q_2, q_3, q_4, *and* q_5), so $ALP(tipo) = HOR(q_2) + HOR(q_3) + HOR(q_4) + HOR(q_5) = 874 + 1748 + 1992 + 4344 = 8958$. The costs of *tipo*, *validado*, and *nombre* are presented in Table 11.

Table 11. ALP table obtained with XAMANA.

Attribute	ALP
tipo	8958
validado	7372
nombre	3752

Finally, the horizontal fragmentation scheme is obtained. For this purpose, the attribute with the highest cost in the ALP table is considered, which is *tipo*. This attribute defines the predicates of the horizontal fragments; therefore, four operations have *tipo* in their predicate: q_2, q_3, q_4, and q_5. q_2 and q_3 have the same predicate; therefore, the horizontal fragmentation scheme has three partitions. Table 12 presents the fragments with their respective costs.

Table 12. Horizontal fragmentation scheme for the case study.

Fragment Name	Predicate	Predicate Overlap	Site	Number of Tuples	Performance Value
multimedia_records_1	{“tipo”:{“$eq”:”event”}}	{$and:[{“tipo”: {“$eq”:”event”}}]}	192.168.1.75	724	4344
multimedia_records_2	{“tipo”: {“$eq”:”personal”}}	{$and:[{$or:[{“tipo”: {$not:{“$eq”:”event”}}}]}, {“tipo”: {“$eq”:”personal”}}]}	192.168.1.63	874	2622
multimedia_records_3	{“tipo”: {“$ne”:”personal”}}	{$and:[{$or:[{“tipo”: {$not: {“$eq”:”personal”}}}]}, {$or:[{“tipo”:{$not: {“$eq”:”event”}}}]}, {“tipo”: {“$ne”:”personal”}}]}	192.168.1.75	272	1992

To obtain the cost of each fragment, Equation (6) is applied; *HFC* (hfr_l) is the cost of each horizontal fragment defined according to its predicate. As an example, for the predicate {“tipo”:{$eq:”personal”}}, *HFC* (*multimedia_records_2*) = $HOR(q_2) + HOR(q_3) = 874 + 1748 = 2622$. The fragments are created based on their costs. First, the most expensive fragment is created, which is *multimedia_records_1*; the subsequent fragments contain the number of tuples that satisfy their predicates minus the tuples already located in previous fragments.

Once XAMANA has vertical and horizontal fragmentation schemes, it proceeds to obtain the hybrid one. For this task, XAMANA begins with the operation analysis to calculate the hybrid cost for each database operation; according to the type of allocation (whether the horizontal or vertical scheme allocation is respected), it will be the remote value to be considered in the hybrid cost.

Table 13 presents the results of the hybrid costs for each query in the operation analysis. For example, in Table 5 the operation q_2 is issued at site 1 and has the predicate {“tipo”:{$eq:”personal”}}; according to the horizontal scheme presented in Table 12, the operation is resolved with the fragment *multimedia_records_2*, also allocated to site 1. Therefore, the remote value considering the horizontal scheme (RV_h) is 1 because the operation

is performed locally; furthermore, q_2 considers the attributes: *tipo*, *_id*, and *nombre*. *multimedia_records_3* is the vertical fragment that contains these three attributes and it is allocated to site 2 as shown in Table 9; therefore, the remote value respecting the vertical scheme (RV_v) is 2.

Table 13. Hybrid costs of operations.

Query	RV_h	RV_v	JC_j	VER_v	HOR_v	VER_h	HOR_h
q_1	2	2	1	0.04	4	0.04	4
q_2	1	2	3	0.18	1748	0.09	874
q_3	1	2	3	0.12	3496	0.06	1748
q_4	1	1	6	0.30	996	0.30	996
q_5	1	1	1	0.03	2172	0.03	2172
q_6	1	1	1	0.02	2	0.02	2
q_7	1	2	1	0.06	12	0.03	6
q_8	2	1	9	0.18	3686	0.36	7372
q_9	2	1	3	0.06	1869	0.12	3738
q_{10}	2	2	3	0.12	4	0.12	4
				V-H:	13,990.11	**H-V:**	16,917.17

In this way, in operation q_2, the vertical costs respecting the horizontal (VER_h) and vertical allocations (VER_v) are obtained with the vertical fragmentation Equation (1); the only change to consider is the value of RV taken according to the allocation: $VER_h(q_2) = OAV_2 \times JC_2 \times RV_2 \times \sum_{i=1}^{3} AS_i = 1 \times 3 \times 1 \times 0.03 = 0.09$; and $VER_V = 1 \times 3 \times 2 \times 0.03 = 0.18$. The horizontal costs respecting the horizontal (HOR_h) and vertical allocations (HOR_v) for q_2 are calculated based on Equation (4) as follows: $HOR_h(q_2) = OAV_2 \times SEL_2 \times RV_2 = 1 \times 874 \times 1 = 874$; and $HOR_v = 1 \times 874 \times 2 = 1748$.

Once the hybrid costs of all operations are available, the final horizontal–vertical (H-V) and vertical–horizontal (V-H) costs of the hybrid fragmentation scheme are calculated using Equations (10) and (11); therefore, $V - H\ (HYFS) = \sum_{j=1}^{10} VER_v(q_j) + HOR_v(q_j) = 13,990.11$ and $H - V\ (HYFS) = \sum_{j=1}^{10} VER_h(q_j) + HOR_h(q_j) = 16,917.17$.

Table 14 shows the result of the static hybrid fragmentation; the token generated by XAMANA once the scheme materialized is 226. Nine fragments are obtained with their respective attributes, predicates with overlap, costs, and site allocations. In addition, the costs of each hybrid scheme, horizontal–vertical or vertical–horizontal, are compared, and the less expensive materialized scheme is the latter.

Table 14. Scheme of static hybrid fragmentation.

Fragment Name	Predicate Overlap	Attributes	Horizontal Site	Vertical Site	Horizontal Cost	Vertical Cost
multimedia_records_1	{“$and”:[{“tipo”: {“$eq”:”personal”}}]}	*tipo, _id, nombre*	192.168.1.63	192.168.1.75	8942.54	10,000.54
multimedia_records_2	{“$and”:[{“tipo”: {“$eq”:”personal”}}]}	*descripcion, imagen, descriptor, _id*	192.168.1.63	192.168.1.75	10,799.54	13,732.63
multimedia_records_3	{“$and”:[{“tipo”: {“$eq”:”personal”}}]}	*validado, _id*	192.168.1.63	192.168.1.75	8930.48	9994.51
multimedia_records_4	{“$and”:[{$or:[{“tipo”: {$not: {“$eq”:”personal”}}}]}, {“tipo”: {“$eq”:”event”}}]}	*tipo, _id, nombre*	192.168.1.75	192.168.1.75	6858.55	10,544.73

Table 14. *Cont.*

Fragment Name	Predicate Overlap	Attributes	Horizontal Site	Vertical Site	Horizontal Cost	Vertical Cost
multimedia_records_5	{“$and”:[{$or:[{“tipo”: {$not: {“$eq”:”personal”}}}]}, {“tipo”: {“$eq”:”event”}}]}	*descripcion, imagen, descriptor, _id*	192.168.1.75	192.168.1.75	6551.54	12,106.78
multimedia_records_6	{“$and”:[{$or:[{“tipo”: {$not: {“$eq”:”personal”}}}]}, {“tipo”: {“$eq”:”event”}}]}	*validado, _id*	192.168.1.75	192.168.1.75	4682.48	8368.66
multimedia_records_7	{“$and”:[{$or:[{“tipo”: {$not: {“$eq”:”event”}}}]}, {$or:[{“tipo”:{$not: {“$eq”:”personal”}}}]}, {“tipo”: {“$ne”:”personal”}}]}	*tipo, _id, nombre*	192.168.1.75	192.168.1.75	4688.62	8374.80
multimedia_records_8	{“$and”:[{$or:[{“tipo”: {$not: {“$eq”:”event”}}}]}, {$or:[{“tipo”:{$not: {“$eq”:”personal”}}}]}, {“tipo”: {“$ne”:”personal”}}]}	*descripcion, imagen, descriptor, _id*	192.168.1.75	192.168.1.75	6555.66	12,110.90
multimedia_records_9	{“$and”:[{$or:[{“tipo”: {$not: {“$eq”:”event”}}}]}, {$or:[{“tipo”:{$not: {“$eq”:”personal”}}}]}, {“tipo”: {“$ne”:”personal”}}]}	*validado, _id*	192.168.1.75	192.168.1.75	4686.60	8372.78
				H-V cost:	16,917.17	
				V-H cost:	13,990.11	

Table 15 presents the queries resolved with the first two fragments of the static hybrid fragmentation scheme shown in Table 14. For example, operations q_2, q_3, q_8, and q_9 are resolved with *multimedia_records_2*; this is because its attributes and the documents that satisfy its predicate with overlap are used in the queries mentioned above. Equations (8) and (9) are used to calculate the cost of a hybrid fragment. Therefore, $V-H$ (*multimedia_records_2*) = $VER_v(q_2)$ + $HOR_v(q_2)$ + $VER_v(q_3)$ + $HOR_v(q_3)$ + $VER_v(q_8)$ + $HOR_v(q_8)$ + $VER_v(q_9)$ + HOR_v (q_9) = 0.18 + 1748 + 0.12 + 3496 + 0.18 + 3686 + 0.06 + 1869 = 10,799.54 and $H-V$ (*multimedia_records_2*) = $VER_h(q_2)$ + $HOR_h(q_2)$ + $VER_h(q_3)$ + $HOR_h(q_3)$ + $VER_h(q_8)$ + $HOR_h(q_8)$ + $VER_h(q_9)$ + $HOR_h(q_9)$ = 0.09 + 874 + 0.06 + 1748 + 0.36 + 7372 + 0.12 + 3738 = 13,732.63.

Table 15. V-H and H-V costs by fragment.

Fragment	Query	VER_h	VER_v	HOR_h	HOR_v	$V-H$ ($hyfr_t$)	$H-V$ ($hyfr_t$)
multimedia_records_1	q_2	0.09	0.18	874	1748	8942.54	10,000.54
	q_3	0.06	0.12	1748	3496		
	q_7	0.03	0.06	6	12		
	q_8	0.36	0.18	7372	3686		
multimedia_records_2	q_2	0.09	0.18	874	1748	10,799.54	13,732.63
	q_3	0.06	0.12	1748	3496		
	q_8	0.36	0.18	7372	3686		
	q_9	0.12	0.06	3738	1869		

5.2. Case Study of Dynamic Hybrid Fragmentation

Next, dynamic hybrid fragmentation is performed. To apply dynamic hybrid fragmentation, the partitioning scheme in Table 14 is used.

A workload of 50 queries is generated from the first five fragments (10 queries for each fragment). This workload is structured as follows: 12 *insert*, 11 *find*, 12 *delete*, and 15 *update* types, of which 24 operations are issued at the first site (192.168.1.63) and 26 at the second site (192.168.1.75). The observer–fragmenter must be at each site along with its respective workload; this way, it will only fragment the partitions located at its site.

If the data is correct, by executing the observer–fragmenter and entering the token obtained from the static hybrid fragmentation plus the path where the workload is located, the information (name and IP) for the database and the name of the collection fragmented are displayed, as well as the performance and operation threshold percentages (in this case, 2% is assigned to both thresholds), the total number of queries in the log (10 operations), and the partitions at the site where the observer–fragmenter is located, as shown in Figure 4. If the data entered by the DBA is incorrect, the observer–fragmenter stops. The observer–fragmenter finds nine fragments. Subsequently, it begins by analyzing the operation threshold concerning the workload, considering all the fragments of the scheme. The first five fragments reach the operation threshold and move to the next phase (cost analysis of the current scheme).

```
run:
Enter token obtained from the Web application to initiate the method (The token is only an integer)
226
Enter database log location: C:\LOGS
C:\Workload_of_50_operations
-----------------------------------------
Data found and information obtained
-----------------------------------------
Database name             HITO
Database IP               192.168.1.63
Fragmented table          multimedia_records
Operation threshold       2.0
Performance threshold     2.0
Total operation log       10.0
-----------------------------------------

Fragments finds
Fragment name: multimedia_records_1
Fragment name: multimedia_records_2
Fragment name: multimedia_records_3
Fragment name: multimedia_records_4
Fragment name: multimedia_records_5
Fragment name: multimedia_records_6
Fragment name: multimedia_records_7
Fragment name: multimedia_records_8
Fragment name: multimedia_records_9
```

Figure 4. Execution of the observer–fragmenter.

Figure 5 shows the execution cost of the new workload for *"multimedia_records_3"*, which is 7372.12. As observed, the fragment's initial performance is 8930.48. The value required to reach the performance threshold is obtained as follows: $(8930.48 \times 2)/100 = 178.6096$. Therefore, the fragment cost is much higher than this value; therefore, a new fragmentation is performed for this partition.

```
----------------------------------------------------------------------
Current cost information for hybrid fragmentation (second analysis):
----------------------------------------------------------------------
Cost in multimedia_records_3:   7372.12
Performance threshold:          2.0
Initial performance:            8930.48
Achieve performance:            178.6096
----------------------------------------------------------------------
This fragment is not CBIR

 -----------CREATE HYBRID SCHEME-------------
```

Figure 5. Cost of multimedia_records_3.

Figure 6 shows the resulting partitioning for the fragment *multimedia_records_3*, which consists of 874 tuples with the attributes *_id* and *validado*. After applying the new fragmentation, the first fragment is now *multimedia_records_3_1* with 871 tuples, which corresponds to queries with the predicate {“$and”:[{“validado”:{$eq:null}}]}, and the second partition is *multimedia_records_3_2*, with only three documents that satisfy the predicate {$or:[{“validado”:{$not:{$eq:null}}}]}.

```
-----------------------------------------
Hybrid Scheme
-----------------------------------------
Total tuples of the original fragment: 874.0
Total tuples of the new fragment: 871
Fragment name: multimedia_records_3_1
Predicate with overlap: {"$and":[{"validado":{$eq:null}}]}
Attribute: validado
Attribute: _id

Total tuples of the original fragment: 874.0
Total tuples of the new fragment: 3
Fragment name: multimedia_records_3_2
Predicate with overlap: {$or:[{"validado":{$not:{$eq:null}}}]}
Attribute: validado
Attribute: _id

Total tuples of the original fragment: 874.0
Total tuples of the new fragment: 0
The fragment is empty
H-V cost:  6982.200000000002
V-H cost:  6982.200000000002
-----------------------------------------

 Both schemes have equal cost but the vertical-horizontal scheme will be applied
```

Figure 6. New scheme for *multimedia_records_3*.

Table 16 presents the schemes of *multimedia_records_2*, *multimedia_records_4*, and *multimedia_records_5*. All five fragments reach the performance threshold; therefore, they move on to obtaining new hybrid schemes. Although *multimedia_records_1* surpasses both thresholds, its fragments and the sites where they are allocated remain the same, so there is no need to materialize a new fragmentation scheme. The partition *multimedia_records_3_1* is located in site 2 (192.168.1.75) and *multimedia_records_3_2* in site 1 (192.168.1.63).

Table 16. The new fragments obtained from *multimedia_records_2*, *multimedia_records_4*, and *multimedia_records_5* through dynamic hybrid fragmentation.

Fragment Name	Attributes	Predicate with Overlap	Tuples	Site	Hybrid Cost
multimedia_records_2_1	*imagen, descriptor, _id*	{"$and":[{"_id":{"$ne": ObjectId("615e99b 8f0f0e30bc46133a5")}}]}	873	192.168.1.63	**H-V:** 9467.34 **V-H:** 9467.34
multimedia_records_2_2	*descripcion, _id*	{"$and":[{"_id":{"$ne": ObjectId("615e99b 8f0f0e30bc46133a5")}}]}	873	192.168.1.63	
multimedia_records_2_3	*imagen, descriptor, _id*	{"$and":[{$or:[{"_id": {$not:{"$ne": ObjectId("615e99b8f0f0 e30bc46133a5")}}}]}, {"_id":{"$ne":ObjectId ("615e9922f0f0e30bc4613237")}}]}	1	192.168.1.63	
multimedia_records_2_4	*descripcion, _id*	{"$and":[{$or:[{"_id": {$not:{"$ne": ObjectId("615e99b8f0f0 e30bc46133a5")}}}]}, {"_id":{"$ne":ObjectId ("615e9922f0f0e30bc4613237")}}]}	1	192.168.1.63	
multimedia_records_4_1	*tipo, _id, nombre*	{"$and":[{"_id":{"$ne": "IMG1598"}}]}	723	192.168.1.63	**H-V:** 5074.26 **V-H:** 5074.26
multimedia_records_4_2	*tipo, _id, nombre*	{"$and":[{$or:[{"_id":{$not:{"$ne": "IMG1598"}}}]}]}	1	192.168.1.63	
multimedia_records_5_1	*imagen, descriptor, _id*	{"$and":[{"descripcion": {"$eq":"."}}]}	561	192.168.1.75	**H-V:** 10,286.61 **V-H:** 15,890.61
multimedia_records_5_2	*descripcion, _id*	{"$and":[{"descripcion": {"$eq":"."}}]}	561	192.168.1.75	
multimedia_records_5_3	*imagen, descriptor, _id*	{"$and":[{$or: [{"descripcion": {$not:{"$eq":"."}}}]}, {"descripcion": {"$ne":"."}}]}	163	192.168.1.63	
multimedia_records_5_4	*descripcion, _id*	{"$and":[{$or: [{"descripcion": {$not:{"$eq":"."}}}]}, {"descripcion": {"$ne":"."}}]}	163	192.168.1.63	

5.3. Performance Evaluation of Our Approach

In this section, we present the results obtained when comparing the proposed method with the horizontal and vertical fragmentation of Castro-Medina et al. [3,36] and with the static hybrid fragmentation of Rodríguez-Mazahua et al. [11], showing the effectiveness and efficiency of the developed technique.

To achieve the comparison and apply static hybrid fragmentation, a workload of 100 operations is obtained on *multimedia_records*, which is structured as follows: 23 *insert*, 25 *find*, 30 *update*, and 22 *delete* types, where 50 queries are issued at the first site (192.168.1.63) and the other 50 at the second site (192.168.1.75). As it is an extensive workload, only the 20 most relevant queries are shown in Table 17.

Table 17. Workload with twenty operations out of one hundred shown.

Query	Operation
q_1	db.multimedia_records.updateMany({“validado”:{$eq:null}},{“$set”:{“validado”:null}})
q_3	db.multimedia_records.deleteMany({“descripcion”:{$eq:”.”}})
q_5	db.multimedia_records.insertOne({“nombre”:”LabIng06”})
q_9	db.multimedia_records.deleteMany({“validado”:{$eq:null}})
q_{17}	db.multimedia_records.find({“tipo”:{$eq:”personal”}},{“_id”:1,”tipo”:1,”validado”:1,”nombre”:1})
q_{28}	db.multimedia_records.updateMany({“descripcion”:{$eq: “profesor1”}}, {“$set”:{“descripcion”:”.”}})
q_{32}	db.multimedia_records.find({“tipo”: {$ne:”building”} },{“_id”:1})
q_{39}	db.multimedia_records.find({“tipo”:{$ne:”event”}},{“descripcion”:1,”tipo”:1,”_id”:1,”nombre”:1,”validado”:1})
q_{45}	db.multimedia_records.deleteMany({“descripcion”:{$ne:”...”}})
q_{48}	db.multimedia_records.updateMany({“validado”:{$ne:null}},{“$set”:{“validado”:null}})
q_{55}	db.multimedia_records.deleteMany({“nombre”:{$ne:”BAZMENMM”}})
q_{61}	db.multimedia_records.find({“tipo”:{$ne:”personal”}},{“nombre”:1,”descripcion”:1,”validado”:1, “tipo”:1, “_id”:1})
q_{68}	db.multimedia_records.updateMany({“descripcion”:{$ne:”profesor2”}},{“$set”:{“descripcion”:”Caricatura a mano”}})
q_{74}	db.multimedia_records.updateMany({“tipo”:{$eq:”equipment”}},{“$set”:{“tipo”:”event”}})
q_{77}	db.multimedia_records.deleteMany({“tipo”:{$eq:”event”}})
q_{83}	db.multimedia_records.find({“validado”:{$ne:null} },{“_id”:1,”nombre”:1})
q_{86}	db.multimedia_records.deleteMany({“validado”:{$eq:null}})
q_{87}	db.multimedia_records.insertOne({“descripcion”:”profesor3”})
q_{90}	db.multimedia_records.updateMany({“tipo”:{$ne:”event”}},{“$set”:{“tipo”:”event”}})
q_{99}	db.multimedia_records.deleteMany({“tipo”:{$ne:”event”}})

First, the vertical fragmentation scheme is obtained. As seen in Table 18, the scheme has two fragments: the first with the attributes *imagen*, *descriptor*, and *_id*, and the second with the rest of the attributes: *descripcion*, *tipo*, *validado*, *nombre,* and *_id*. Both fragments have 1870 documents.

Table 18. Vertical fragmentation scheme for the second experiment.

Fragment Name	Site	Attributes	Cost
multimedia_records_1	192.168.1.75	*imagen, descriptor,_id*	0.94
multimedia_records_2	192.168.1.75	*descripcion, tipo, validado, nombre, _id*	3.90

Subsequently, the horizontal fragmentation scheme is shown in Table 19. *multimedia_records_2* is not considered in the scheme because it is a partition without documents. The mixture of the vertical and horizontal schemes (Tables 18 and 19) results in the hybrid fragmentation scheme presented in Table 20.

Table 19. Horizontal fragmentation scheme for the second experiment.

Fragment Name	Predicate	Predicate Overlap	Site	Number of Tuples	Performance Value
multimedia_records_1	{“validado”: {“$eq”:null}}	{$and:[{“validado”: {“$eq”:null}}]}	192.168.1.75	1843	46,075
multimedia_records_3	{“validado”: {“$ne”:null}}	{$and:[{$or: [{“validado”:{$not: {“$eq”:null}}}]}, {“validado”: {“$ne”:null}}]}	192.168.1.75	27	729

Table 20. Hybrid fragmentation scheme for the second experiment.

Fragment Name	Predicate Overlap	Attributes	Horizontal Site	Vertical Site	Horizontal Cost	Vertical Cost
multimedia_records_1	{“$and”:[{“validado”: {“$eq”:null}}]}	*descripcion, tipo, validado, _id*	192.168.1.75	192.168.1.75	206,101.20	206,101.20
multimedia_records_2	{“$and”:[{“validado”: {“$eq”:null}}]}	*imagen, descriptor, _id*	192.168.1.75	192.168.1.75	107,983.08	107,983.08
multimedia_records_3	{“$and”:[{$or: [{“validado”:{$not: {“$eq”:null}}}]}, {“validado”: {“$ne”:null}}]}	*descripcion, tipo, validado, _id*	192.168.1.75	192.168.1.75	142,422.48	142,422.48
multimedia_records_4	{“$and”:[{$or: [{“validado”:{$not: {“$eq”:null}}}]}, {“validado”: {“$ne”:null}}]}	*imagen, descriptor, _id*	192.168.1.75	192.168.1.75	91,224.98	91,224.98
				H-V cost:	206,831.07	
				V-H cost	206,831.07	

Based on the static hybrid fragmentation scheme in Table 20, dynamic hybrid fragmentation is applied with a workload of 400 operations with 102 insertions, 95 queries, 115 updates, and 88 deletions. One hundred operations are performed on each fragment and 200 at each site. The observer–fragmenter is executed at each site with token 227 and the workload. The observer–fragmenter recovers the information for the fragments at each site, as presented in Figure 7. Subsequently, it analyzes which fragments meet the operation threshold; those that do not are removed. However, the four fragments meet this criterion, and their performance costs are obtained.

```
run:
Enter token obtained from the Web application to initiate the method (The token is only an integer)
227
Enter database log location: C:\LOGS
C:\Workload_of_400_operations


-----------------------------------------
Data found and information obtained
-----------------------------------------
Database name             HITO
Database IP               192.168.1.63
Fragmented table          multimedia_records
Operation threshold       2.0
Performance threshold     2.0
Total operation log      100.0
-----------------------------------------

Fragments finds
Fragment name: multimedia_records_1
Fragment name: multimedia_records_2
Fragment name: multimedia_records_3
Fragment name: multimedia_records_4
```

Figure 7. Execution of the observer–fragmenter for the second experiment.

Figure 8 shows the performance cost for *multimedia_records_1*, and Table 21 contains the costs of the other fragments. Since all fragments reach the value established by the performance threshold (achieve performance), a new hybrid fragmentation is performed on them.

```
-----------------------------------------------------------------
Current cost information for hybrid fragmentation (second analysis):
-----------------------------------------------------------------
Cost in multimedia_records_1:    374036.4799999996
Performance threshold:           2.0
Initial performance:             206101.203
Achieve performance:             4122.024060000001
-----------------------------------------------------------------
This fragment is not CBIR

----------CREATE HYBRID SCHEME-------------
```

Figure 8. Cost of *multimedia_records_1*.

Table 21. Cost per fragment of dynamic hybrid fragmentation.

Fragment Name	Cost	Performance Threshold	Initial Performance	Achieve Performance
multimedia_records_2	48,072.119	2	107,983.078	2159.661
multimedia_records_3	295,674.679		142,422.484	2848.449
multimedia_records_4	48,072.119		91,224.977	1824.499

Figure 9 presents the scheme for the first fragment (*multimedia_records_1*), from which two new partitions are obtained, *multimedia_records_1_1* and *multimedia_records_1_2*. Both have the same attributes (*descripcion, tipo, validado, nombre,* and *_id*); the difference lies in the predicate with overlap for each partition. A total of 1842 documents satisfy the predicate of *multimedia_records_1_1*. Meanwhile, in *multimedia_records_1_2*, there is only one document. Finally, Table 22 presents the new scheme of the fragments *multimedia_records_2*, *multimedia_records_3*, and *multimedia_records_4*.

```
----------------------------------------
Hybrid Scheme
----------------------------------------
Total tuples of the original fragment: 1843.0
Total tuples of the new fragment: 1842
Fragment name: multimedia_records_1_1
Predicate with overlap: {"$and":[{"_id":{"$ne":ObjectId("615e99abf0f0e30bc461334b")}}]}
Attribute: descripcion
Attribute: tipo
Attribute: validado
Attribute: nombre
Attribute: _id

Total tuples of the original fragment: 1843.0
Total tuples of the new fragment: 1
Fragment name: multimedia_records_1_2
Predicate with overlap: {"$and":[{$or:[{"_id":{$not:{"$ne":ObjectId("615e99abf0f0e30bc461334b")}}}]},
                        {"_id":{"$ne":ObjectId("615e9803f0f0e30bc461317a")}}]}
Attribute: descripcion
Attribute: tipo
Attribute: validado
Attribute: nombre
Attribute: _id

Total tuples of the original fragment: 1843.0
Total tuples of the new fragment: 0
The fragment is empty
H-V cost:  245108.5299999999
V-H cost:  245108.5299999999
----------------------------------------

Both schemes have equal cost but the vertical-horizontal scheme will be applied
```

Figure 9. New scheme for *multimedia_records_1*.

Table 22. Scheme of dynamic hybrid fragmentation.

Fragment Name	Attributes	Predicate with Overlap	Tuples	Site	Hybrid Cost
multimedia_records_2_1	*imagen, descriptor, _id*	{“$and”:[{“_id”:{“$ne”: ObjectId(“615e997af0f0e 30bc46132ad”)}}]}	1842	192.168.1.63	**H-V:** 217,480.85 **V-H:** 217,480.85
multimedia_records_2_2	*imagen, descriptor, _id*	{“$and”:[{$or:[{“_id”:{$not: {“$ne”:ObjectId (“615e997af0f0e30bc46 132ad”)}}}]},{“_id”: {“$ne”:”IMG768”}}]}	1	192.168.1.63	
multimedia_records_3_1	*descripcion, tipo, validado, nombre, _id*	{“$and”:[{“_id”:{“$ne”: “IMG841”}}]}	26	192.168.1.63	**H-V:** 2654.83 **V-H:** 2654.83
multimedia_records_3_2	*descripcion, tipo, validado, nombre, _id*	{“$and”:[{$or:[{“_id”:{$not: {“$ne”:”IMG841”}}}]}, {“_id”:{“$ne”:”IMG1930”}}]}	1	192.168.1.63	
multimedia_records_4_1	*imagen, descriptor, _id*	{“$and”:[{“_id”:{“$ne”: ObjectId(“615e951b92 bd5f6ac041f477”)}}]}	26	192.168.1.63	**H-V:** 2871.87 **V-H:** 2628.51
multimedia_records_4_2	*imagen, descriptor, _id*	{“$and”:[{$or:[{“_id”:{$not: {“$ne”: ObjectId(“615e951 b92bd5f6ac041f477”)}}}]}, {“_id”:{“$ne”:ObjectId (“615e94e492bd5f6ac041 f438”)}}]}	1	192.168.1.63	

Table 23 compares the costs of the first ten operations in Table 17 obtained with the vertical (VFS), horizontal (HFS), and hybrid (HYFS) fragmentation schemes of Tables 18–20 and using Equations (1), (4) and (7). In this table, ID refers to the irrelevant documents accessed by each query, while IAT is the size of the irrelevant attributes in bytes retrieved by the queries.

Table 23. Comparison of the cost for each operation with vertical, horizontal, and hybrid fragmentation schemes.

Query	VFS			HFS			HYFS		
	VER	ID	IAT	HOR	ID	IAT	HYB	ID	IAT
q_1	60,000	27	**40,000**	**11,058**	**0**	120,000	71,058	**0**	**40,000**
q_3	80,000	**565**	**40,000**	**5220**	**565**	120,000	85,220	**565**	**40,000**
q_5	40,000	**0**	**0**	**4**	**0**	**0**	40,004	**0**	**0**
q_9	80,000	27	**40,000**	**7372**	**0**	120,000	87,372	**0**	**40,000**
q_{17}	80,000	**996**	**10,000**	**1748**	**996**	90,000	81,748	**996**	**10,000**
q_{28}	60,000	1869	**40,000**	**6**	**1842**	120,000	60,006	**1842**	**40,000**
q_{32}	40,000	**168**	**30,000**	**3404**	**168**	110,000	43,404	**168**	**30,000**
q_{39}	100,000	**724**	**0**	**2292**	**724**	80,000	102,292	**724**	**0**
q_{45}	80,000	**131**	**40,000**	**6956**	**131**	120,000	86,956	**131**	**40,000**
q_{48}	60,000	1843	**40,000**	**162**	**0**	120,000	60,162	**0**	**40,000**

Note: Bold values indicate the lowest execution cost obtained for each operation according to vertical, horizontal, and hybrid fragmentation schemes.

Based on the results obtained in the first comparison (Table 23), in the case of vertical fragmentation, irrelevant attribute access is avoided because only the attributes that the operations require are considered. Therefore, the queries access a greater number of irrelevant documents. On the other hand, in the horizontal scheme, the operations have

a lower cost because the size of the attributes is not considered; however, it has a greater amount of access to irrelevant attributes. In contrast, hybrid fragmentation, by mixing the vertical and horizontal schemes, considers both predicates and attributes, so it has better performance in terms of tuples and irrelevant attributes.

Next, we present the comparison of the method proposed in this work with the approach of Rodríguez-Mazahua et al. [11]. The result obtained according to the queries in Table 17 is a static hybrid fragmentation with eight fragments, as visualized in Table 24. All fragments are assigned to the first site (192.168.1.63). Table 25 compares the hybrid fragmentation scheme obtained with the method proposed in this work and that of Rodríguez-Mazahua et al. [11] based on the last ten queries in Table 17.

Table 24. Hybrid fragmentation scheme obtained by Rodríguez-Mazahua et al. [11].

Fragment Name	Attributes	Predicate
multimedia_records_1	*descripcion, tipo, validado, _id, nombre*	{"tipo":{$eq:"personal"}}
multimedia_records_2	*descripcion, tipo,validado, _id, nombre*	{"tipo":{$eq:"equipment"}}
multimedia_records_3	*descripcion, tipo, validado, _id, nombre*	{"tipo":{$eq:"event"}}
multimedia_records_4	*descripcion, tipo, validado, _id, nombre*	{"tipo":{$eq:"building"}}
multimedia_records_5	*imagen, descriptor,_id*	{"tipo":{$eq:"personal"}}
multimedia_records_6	*imagen, descriptor, _id*	{"tipo":{$eq:"equipment"}}
multimedia_records_7	*imagen, descriptor,_id*	{"tipo":{$eq:"event"}}
multimedia_records_8	*imagen, descriptor,_id*	{"tipo":{$eq:"building"}}

Table 25. Result of the comparison with the work of Rodríguez-Mazahua et al. [11].

Query	Our Scheme			Rodríguez-Mazahua et al. [11]		
	HYB	***RV***	***JC***	**HYB**	***RV***	***JC***
q_{55}	**21,869**	**1**	**2**	83,738	2	4
q_{61}	**25,498**	**1**	**1**	200,996	2	4
q_{68}	**17,804**	**1**	**1**	125,607	2	4
q_{74}	**15,156**	**1**	**1**	30,312	2	**1**
q_{77}	**20,724**	**1**	2	21,448	2	**1**
q_{83}	**15,014**	**1**	**1**	120,027	2	4
q_{86}	**21,483**	**1**	**2**	83,686	2	4
q_{87}	**5001**	**1**	**2**	83,686	2	4
q_{90}	**16,719**	**1**	**1**	123,438	2	4
q_{99}	**21,146**	**1**	**2**	82,292	2	4

The hybrid scheme obtained with the proposed method has a lower cost than the technique presented by Rodríguez-Mazahua et al. [11] for 77% of the 100 operations in the workload. This is because most operations require fewer joins in the scheme obtained with the method presented in this work. Also, the remote value in the scheme of Rodríguez-Mazahua et al. [11] is greater in most operations since it only performs fragmentation and does not consider fragment allocation; for this reason, all fragments are allocated to site 1, increasing the cost of operations issued at site 2.

We expanded our experimental evaluation by meeting three objectives: First, estimating the query response time, taking into account the workloads used in this section (100 and 400 operations), each executed five times and considering the total, average, and 95th percentile time of each operation. Second, comparing our approach vs. other dynamic fragmentation methods, considering the operating and performance thresholds at 2% and 90%. Third, by analyzing the sensitivity of our cost model. We compare the response time of the 100 operations considering the initial hybrid fragmentation scheme (Table 20) obtained

with our method (static hybrid fragmentation, SHYBF) vs. the centralized database (CDB) and the vertical (static vertical fragmentation, SVERF) and horizontal (static horizontal fragmentation, SHORF) fragmentation schemes (Tables 18 and 19) produced by the methods of Castro-Medina et al. [3,36]. The response time of the remaining 400 operations is compared to that of the partitioning scheme generated by our hybrid fragmentation method (dynamic hybrid fragmentation, DHYBF), with operation and performance thresholds of 2% (Figure 9 and Table 22) and 90%, vs. the partitioning schemes obtained by the dynamic horizontal fragmentation technique proposed by Castillo-García et al. [12] (dynamic horizontal fragmentation, DHORF) and the dynamic vertical fragmentation of Castro-Medina et al. [36] (dynamic vertical fragmentation, DVERF).

Table 26 shows that horizontal fragments *multimedia_records_1* and *multimedia_records_3* surpass both performance thresholds (2% and 90%). Nevertheless, only the second vertical fragment exceeds them. Therefore, Tables 27 and 28 illustrate the new partitioning schemes (DVERF and DHORF). By applying dynamic horizontal and vertical fragmentation with operation and performance thresholds of 2% and 90%, we obtain the same fragmentation schemes (Tables 27 and 28). Meanwhile, when applying our hybrid fragmentation method (DHYBF) with thresholds of 90%, only *multimedia_records_1* and *multimedia_records_3* are fragmented, as shown in Figure 9 and Table 22.

Table 26. Cost per fragment of dynamic vertical and horizontal fragmentation.

Type of Fragmentation	Fragment Name	Cost	Performance Threshold	Initial Performance	Achieve Performance
Vertical	*multimedia_records_1*	0	2 90	0.940	0.0188 0.846
	multimedia_records_2	14.560	2 90	3.900	0.078 3.510
Horizontal	*multimedia_records_1*	101,235	0 90	46,075	921.500 41,467.500
	multimedia_records_3	123,636	0 90	729	14.580 656.100

Table 27. Dynamic vertical fragmentation scheme obtained by Castro-Medina et al. [36].

Fragment Name	Site	Attribute	Cost
multimedia_records_1	192.168.1.75	*imagen, descriptor,_id*	0
multimedia_records_2_1	192.168.1.63	*descripcion,validado,_id*	5.38
multimedia_records_2_2	192.168.1.75	*tipo,nombre,_id*	5.40

Table 28. Dynamic horizontal fragmentation scheme obtained by Castillo-García et al. [12].

Fragment Name	Predicate	Predicate with Overlap	Site	Number of Tuples	Performance Value
multimedia_records_1	{“validado”:{“$eq”:null}}	{$and:[{"validado": {"$eq":null}}]}	192.168.1.63	1843	5625
multimedia_records_3_1	{“_id”:{“$ne”:”IMG1930”}}	{$and:[{$or:[{"validado": {$not:{"$eq":null}}}]},{"_id": {"$ne":"IMG1930"}}]}	192.168.1.63	26	11,232
multimedia_records_3_2	{“_id”:{“$ne”:”IMG1791”}}	{$and:[{$or:[{"_id":{$not: {"$ne":"IMG1930"}}}]},{$or: [{"validado":{$not: {"$eq":null}}}]},{"_id": {"$ne":"IMG1791"}}]}	192.168.1.63	1	3740

Figure 10 graphically depicts the query response time, and Table 29 presents the results in greater detail. According to the results obtained, for static fragmentation, the vertical fragmentation method outperforms the initial fragmentation scheme obtained by our technique; nevertheless, when dynamic fragmentation is performed considering thresholds at 2%, the scheme obtained by our method achieves the best results. In contrast, when the dynamic hybrid fragmentation thresholds are increased to 90%, the query response times increase, as shown in Table 29.

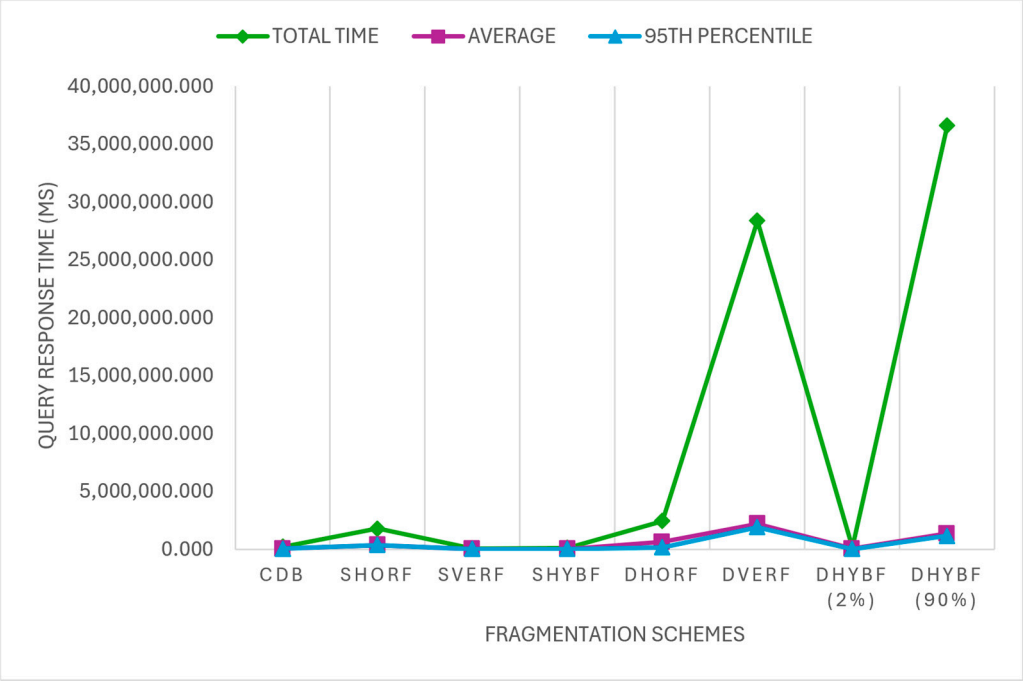


Figure 10. Graphical representation of query response times.

Table 29. Comparison of the execution time in milliseconds for vertical, horizontal, and hybrid fragmentation schemes.

Fragmentation Schemes	Total Time	Average Time	95th Percentile
CDB	203,562.942	40,712.588	35,737.946
SHORF	1,772,698.062	354,539.612	352,866.634
SVERF	**69,300.132**	**13,873.751**	**13,930.288**
SHYBF	112,926.766	22,585.353	8062.185
DHORF	2,442,030.843	627,605.325	158,954344
DVERF	28,391,330.490	2,198,506.147	1,909,949.769
DHYBF (2%)	**150,975.382**	**30,195.076**	**27,474.493**
DHYBF (90%)	36,595,530.060	1,339,579.979	1,152,206.491

We executed a Friedman test followed by a Durbin–Conover test, Holm adjusted, to compare the dynamic fragmentation schemes vs. the partitioning scheme obtained by our approach with the operation and performance thresholds at 2% (DHYBF_2%). Figure 11 shows that two pairwise comparisons are very significant (**): the DHYBF_2 - DVERF ($p.$ adj. = 0.002) and DHYBF_2 - DHYBF_90 ($p.$ adj. = 0.004) fragmentation schemes, considering the total, average, and 95th query response times.

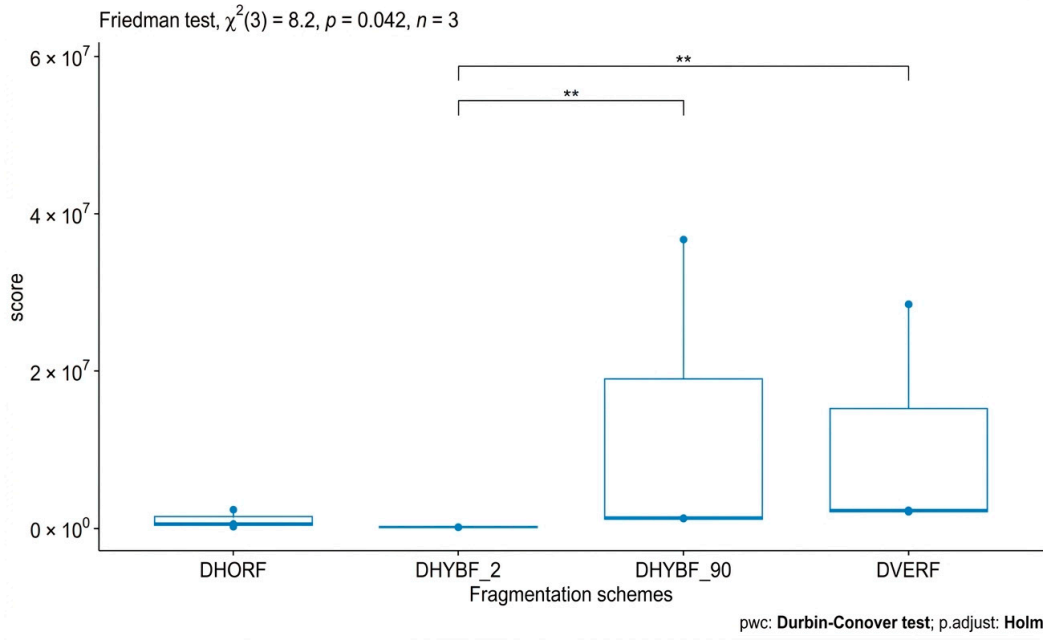


Figure 11. Friedman test of fragmentation schemes.

6. Discussion

Multimedia databases are of great importance in information systems, as they allow for the storage of large amounts of data of various types, such as images, audio, and video. However, multimedia data, due to their large size, represent a challenge for efficient information retrieval, as they increase the data transmission costs and response times. To address these problems, we developed a partitioning method for multimedia data that considers horizontal and vertical fragmentation, based on cost and affinity. In addition, it adapts the fragmentation scheme according to changes in the workload in a distributed environment.

The presented method complies with the three fragmentation rules. (1) Completeness: All tuples and attributes of the collection are mapped onto fragments without any loss. (2) Reconstruction: The collection is rebuilt from the obtained fragments, that is, for horizontal fragmentation through the union operator and for vertical fragmentation through the join. (3) Disjointness: The fragments must not overlap. In the case of horizontal fragmentation, the horizontal fragments are disjoint, whereas in vertical fragmentation the primary key attributes are normally repeated in all its fragments (for reconstruction purposes); that is, disjointness is defined only for attributes that are not part of the primary key of a relation.

The proposed cost model enables comparison of hybrid fragmentation schemes based on workload and, subsequently, the selection of the less costly fragmentation and allocation scheme (horizontal–vertical or vertical–horizontal). Likewise, the proposed approach operates under these characteristics. (1) The observer–fragmenter must be present at each of the sites considered, along with the respective workloads. This way, it will only re-fragment the partitions located at its site (server), and each observer–fragmenter will evaluate whether it is necessary to update the information for the fragments it supervises at each site. (2) Before re-fragmenting and allocating, the following must be considered: (a) When one of the hybrid fragments has the same number of tuples and attributes as the original fragment, partitioning is not performed again; however, if the site where it should be allocated changes, then it is moved from its place of origin—that is, it is allocated to the destination site and kept as is, retaining its name. (b) If a hybrid fragment is empty (contains no tuples or attributes), then it is neither created nor named. (c) When

a fragment is allocated to the site where the database is located, it is added to it; but if the site is different and the database does not exist, in the case of MongoDB, the database and the new fragments are created. (3) The thresholds are values (from 0 to 100) that the DBA assigns when applying static hybrid fragmentation. The smaller these values, the more sensitive and frequent the new fragmentations are. The operations threshold is the percentage of operations required to perform a new fragment cost analysis, and the performance threshold is the percentage of fragment cost set by the DBA that triggers a new fragmentation. Therefore, in dynamic hybrid fragmentation, these thresholds allow for determining when it is appropriate to carry out a new re-fragmentation.

To improve the explanation of the configuration of the operation and performance thresholds, Figure 12 graphically shows the different combinations of thresholds (operation and performance) with respect to 10%, 50%, and 90%. If the DBA sets the operation threshold at 10% and the performance threshold at 50%, the probability of performing a new fragmentation is medium, whereas if both thresholds are set at 90%, the probability is low; that is, it is less likely that a new fragmentation will be triggered because a greater number of queries and a higher performance value are required. The DBA will decide the correct values for these thresholds according to the characteristics of the multimedia database; this decision can also be made according to the cost of a new fragmentation using Equation (12). When this cost is greater than the cost of the V-H or the H-V obtained with Equations (10) and (11), then a new fragmentation is not recommended; therefore, the operation and performance thresholds must increase. Additionally, we take the first hybrid fragment from the second experiment as a reference; the number of operations and the performance cost that must be reached for a new fragmentation are calculated (Table 30) considering 10%, 50%, and 90% for both thresholds. One hundred new operations are executed, and the cost is now greater than all the thresholds; therefore, a new fragmentation is triggered.

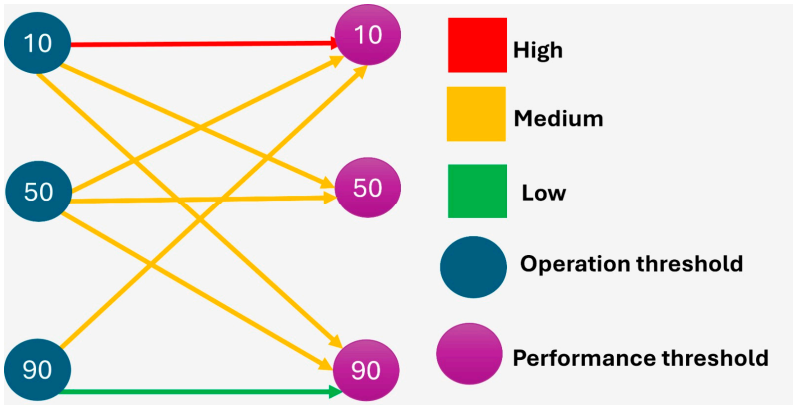


Figure 12. Graphical representation of operation and performance thresholds.

Table 30. Costs obtained for *multimedia_records_1* for a new fragmentation.

Initial Operations	Initial Performance	Operation Threshold	Achieve Operations	Performance Threshold	Achieve Performance	Cost
100	206,101.203	10	10	10	20,610.120	374,036.479
		50	50	50	103,050.601	
		90	90	90	185,491.083	

7. Conclusions, Limitations, and Directions for Future Research

A dynamic hybrid fragmentation method for multimedia databases is presented that can adapt the fragment scheme to changes in the workload and considers a cost

model. The proposed cost model allows for comparison of hybrid fragmentation schemes based on the workload to obtain and subsequently apply the least expensive scheme. The observer–fragmenter can monitor the operation and performance thresholds without the intervention of the DBA. When the cost of the fragment reaches the thresholds, a new fragmentation is automatically triggered, and the new partitions produced are allocated to the corresponding sites.

We compare our approach with another state-of-the-art method and show that it is possible to reduce the execution cost of 77% of queries by accessing a smaller amount of remote data and requiring fewer joins. Therefore, the selection of the type of fragmentation (vertical, horizontal, or hybrid) to apply will depend on the DBA optimization goals. If access to fewer irrelevant documents by operations is needed, then horizontal fragmentation is most appropriate; conversely, if reducing access to irrelevant attributes is required, vertical partitioning is recommended. When operations involve subsets of documents and attributes, hybrid fragmentation is the best alternative solution.

Despite the aforementioned benefits, our method has limitations: it was developed for fragmentation in MongoDB; nevertheless, it was designed to be adapted to another NoSQL DBMS, such as Cassandra. On the other hand, the application and evaluation of hybrid fragmentation were done in a controlled environment; that is, the workloads were generated using a query generator, only two sites with a fixed network were considered, and the maximum number of operations was 400. Nevertheless, the method presented can analyze new operations and detect new sites. For this reason, as future work, the experimental analysis will be expanded using other databases, datasets, and benchmarks, including larger data volumes and diverse data distributions. Additionally, the configuration will consider more network sites, and the impact of thresholds (operation and performance) will also be studied to verify how different configurations affect the system's performance and stability.

Due to the increasing volume of data generated by applications and/or devices connected to the internet, especially through the Internet of Things (IoT), and since Big Data refers to data that is complex to process because of the volume generated when stored, another area for future research is to focus on developing new methods of horizontal, vertical, and hybrid fragmentation for Big Data based on graphs that optimize operations in massive data repositories. We intend another line of research to consider CBIR and CBVR (Content-Based Video Retrieval) within the proposed method, and to contemplate other types of NoSQL databases, such as graph-oriented databases.

Author Contributions: Conceptualization, L.R.-M. and F.C.-M.; methodology, L.R.-M., F.C.-M. and O.C.-R.; software, L.R.-M., F.C.-M. and O.C.-R.; validation, G.A.-H. and J.C.; formal analysis, L.R.-M. and F.C.-M.; investigation, L.R.-M., F.C.-M. and O.C.-R.; resources, L.R.-M. and G.A.-H.; data curation, M.L.A.-R., L.R.-M. and F.C.-M.; writing—original draft preparation, O.C.-R., L.R.-M. and F.C.-M.; writing—review and editing, L.R.-M., G.A.-H., J.C., N.R.-M. and M.L.A.-R.; visualization, F.C.-M., O.C.-R. and L.R.-M.; supervision, F.C.-M., L.R.-M. and G.A.-H.; project administration, L.R.-M. and F.C.-M.; funding acquisition, L.R.-M., G.A.-H., N.R.-M., and M.L.A.-R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Secretariat of Science, Humanities, Technology and Innovation (SECIHTI) through a doctoral scholarship provided to the first author and a postdoctoral research fellowship given to the second and sixth authors (Application numbers: 8177740 and 10618225).

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors are very grateful to the National Technological of Mexico (TecNM) for supporting this work. This research paper was sponsored by the Secretariat of Science, Humanities, Technology and Innovation (SECIHTI), and the Secretariat of Public Education (SEP).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Özsu, M.T.; Valduriez, P. *Principles of Distributed Database Systems*, 4th ed.; Springer International Publishing: Cham, Switzerland, 2020.
2. 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*, 1352. [CrossRef] [PubMed]
3. Castro-Medina, F.; Rodríguez-Mazahua, L.; López-Chau, A.; Cervantes, J.; Alor-Hernández, G.; Machorro-Cano, I.; Arrioja-Rodríguez, M.L. A New Method of Dynamic Horizontal Fragmentation for Multimedia Databases Contemplating Content-Based Queries. *Electronics* **2022**, *11*, 288. [CrossRef]
4. Mahmoud, S.H.; Abdelraouf, A.E.; Badr, N.L.; Ali, M.I. Optimizing Vertical Distribution Techniques For Heritage Multimedia Databases: Challenges And Solutions. *J. Theor. Appl. Inf. Technol.* **2025**, *103*, 366–374.
5. Mahmoud, S.H.; Badr, N.; Abdelraouf, A.E.; Ali, M.I. RL-Based Fragment Allocation and Replication for Distributed Heritage Multimedia Databases. *Int. J. Intell. Comput. Inf. Sci.* **2025**, *25*, 73–92. [CrossRef]
6. Safaei, A.A. Hybrid Fragmentation of Medical Images' Attributes for Multidimensional Indexing. *Clust. Comput.* **2022**, *25*, 215–230. [CrossRef]
7. Patel, M.; Yadav, N.; Bhise, M. Workload Aware Cost-Based Partial Loading of Raw Data for Limited Storage Resources. In *Futuristic Trends in Networks and Computing Technologies*; Springer: Singapore, 2022; Volume 936, pp. 1035–1048.
8. Crescencio-Rico, O.; Rodríguez-Mazahua, L.; Castro-Medina, F.; Alor-Hernández, G.; Sánchez-Cervantes, J.L. Dynamic Hybrid Fragmentation Method for Multimedia Databases. *Pädi Bol. Cienti Cienc. Básicas Ing. ICBI* **2023**, *11*, 47–54. [CrossRef]
9. Rodríguez-Mazahua, L.; Alor-Hernández, G.; Li, X.; Cervantes, J.; López-Chau, A. Active Rule Base Development for Dynamic Vertical Partitioning of Multimedia Databases. *J. Intell. Inf. Syst.* **2017**, *48*, 421–451. [CrossRef]
10. Rodríguez-Mazahua, L.; Alor-Hernández, G.; Abud-Figueroa, M.A.; Peláez-Camarena, S.G. Horizontal Partitioning of Multimedia Databases Using Hierarchical Agglomerative Clustering. In *Nature-Inspired Computation and Machine Learning*; Gelbukh, A., Espinoza, F.C., Galicia-Haro, S.N., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 296–309.
11. Rodríguez-Mazahua, L.; Alor-Hernández, G.; Cervantes, J.; López-Chau, A.; Sánchez-Cervantes, J.L. A Hybrid Partitioning Method for Multimedia Databases. *Dyna* **2016**, *83*, 59–67. [CrossRef]
12. Castillo-García, A.; Rodríguez-Mazahua, L.; Castro-Medina, F.; Arrioja-Rodríguez, M.L.; Sánchez-Cervantes, J.L.; Rodríguez-Mazahua, N. Design of a Dynamic Horizontal Fragmentation Method for Multimedia Databases. In *Intelligent Systems Reference Library*; Springer Science and Business Media Deutschland GmbH: Cham, Switzerland, 2023; Volume 226, pp. 71–91.
13. Ortiz-Ballona, A.O.; Rodríguez-Mazahua, L.; López-Chau, A.; Castro-Medina, F.; Abud-Figueroa, M.A.; Rodríguez-Mazahua, N. A Vertical Fragmentation Method for Multimedia Databases Considering Content-Based Queries. In *Intelligent Systems Reference Library*; Springer Science and Business Media Deutschland GmbH: Cham, Switzerland, 2023; Volume 226, pp. 3–23.
14. Saad, S.; Tekli, J.; Chbeir, R.; Yetongnon, K. Towards Multimedia Fragmentation. In *Advances in Databases and Information Systems*; Lecture Notes in Computer Science; Manolopoulos, Y., Pokorny, J., Sellis, T.K., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4152, pp. 415–429.
15. Zhou, X.; Li, G.; Feng, J.; Liu, L.; Guo, W. Grep: A Graph Learning Based Database Partitioning System. *Proc. ACM Manag. Data* **2023**, *1*, 1–24. [CrossRef]
16. Cantini, R.; Marozzo, F.; Orsino, A.; Talia, D.; Trunfio, P.; Badia, R.M.; Ejarque, J.; Vázquez-Novoa, F. Block Size Estimation for Data Partitioning in HPC Applications Using Machine Learning Techniques. *J. Big Data* **2024**, *11*, 19. [CrossRef]
17. Mao, Q.; Hristidis, V. Enhancing LSM Trees Merge Efficiency via Coordinated Sorted Runs Partitioning. *J. Big Data* **2025**, *12*, 252. [CrossRef]
18. Shoaib, M.; Younus, M.U.; Safdar, K.; Dogar, A.B. A Hybrid Approach for Efficient Database Fragmentation in Distributed Systems. *Int. J. Innov. Sci. Technol.* **2026**, *8*, 162–178. [CrossRef] [PubMed]
19. Badran, S.; Arman, N.; Farajallah, M. Towards a Hybrid Data Partitioning Technique for Secure Data Outsourcing. In *Proceedings of the 2020 21st International Arab Conference on Information Technology, ACIT 2020*; Institute of Electrical and Electronics Engineers Inc.: New York, NY, USA, 2020.
20. Noraziah, A.; Fauzi, A.A.C.; Ubaidillah, S.H.S.A.; Alkazemi, B.; Odili, J.B. BVAGQ-AR for Fragmented Database Replication Management. *IEEE Access* **2021**, *9*, 56168–56177. [CrossRef]

21. Badran, S.; Arman, N.; Farajallah, M. An Efficient Approach for Secure Data Outsourcing Using Hybrid Data Partitioning. In *Proceedings of the 2021 International Conference on Information Technology, ICIT 2021—Proceedings*; Institute of Electrical and Electronics Engineers Inc.: New York, NY, USA, 2021; pp. 418–423.
22. Azila, A.; Fauzi, C.; Fariza, W.; Rahman, W.A.; Fauzi, A.; Weigelt, F. Managing Fragmented Database in Distributed Database Environment. *J. Math. Comput. Sci.* **2021**, *7*, 8–14.
23. Ahmed, Z.J.; Alluhaibi, S.T. Hybrid Data Fragmentation Using Genetic Killer Whale Optimization-Based Clustering Model. *J. Pharm. Negat. Results* **2022**, *13*, 290–298. [CrossRef]
24. Schreiner, G.A.; De Santiago, R.; Duarte, D.; Dos Santos Mello, R. An Autonomous Hybrid Data Partitioning Approach for NewSQL Databases. *J. Braz. Comput. Soc.* **2026**, *32*, 55–72. [CrossRef]
25. Pinnecke, M.; Campero Durand, G.; Broneske, D.; Zoun, R.; Saake, G. GridTables: A One-Size-Fits-Most H2TAP Data Store. *Datenbank-Spektrum* **2020**, *20*, 43–56. [CrossRef]
26. Kulba, V.; Somov, S. Dynamic Fragment Allocation in Distributed System with Time-Varying Parameters of Its Operation. In *Proceedings of the 2020 13th International Conference "Management of Large-Scale System Development" (MLSD)*; IEEE: New York, NY, USA, 2020; pp. 1–4.
27. Kang, D.; Jiang, R.; Blanas, S. Jigsaw: A Data Storage and Query Processing Engine for Irregular Table Partitioning. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 898–911.
28. Cui, S.; Wang, H.; Lin, J.; Ding, X.; Yang, D. WorthyPar: A Workload-Aware Data Hybrid Partitioning Advisor with Deep Reinforcement Learning. In *Proceedings of the Lecture Notes in Computer Science*; LNCS; Springer Science and Business Media Deutschland GmbH: Berlin/Heidelberg, Germany, 2026; Volume 15989, pp. 105–122.
29. Cui, S.; Wang, H.; Lin, J.; Ding, X.; Yang, D. FinePar: A Fine-Grained Data Partitioning Framework for Cloud-Edge-Device Architectures. *IEEE Trans. Knowl. Data Eng.* **2026**, *38*, 2262–2276. [CrossRef]
30. Ahmed, Z.; Alhenshiri, A. Adaptive Query-Driven Fragmentation for Performance Optimization in Distributed Databases. *Alqalam J. Med. Appl. Sci.* **2026**, *9*, 849–855. [CrossRef]
31. Singh, A.P.; Choudhary, A. Approach for Ensuring Fragmentation and Integrity of Data in SEDuLOUS. In *Proceedings of Second International Conference on Computing, Communications, and Cyber-Security*; Springer: Singapore, 2021; Volume 203, pp. 857–869.
32. Liu, P.J.; Li, H.Y.; Wang, T.Y.; Liu, H.; Sun, L.M.; Ren, Y.F.; Li, C.P.; Chen, H. Multi-Stage Method for Online Vertical Data Partitioning Based on Spectral Clustering. *Ruan Jian Xue Bao/J. Softw.* **2023**, *34*, 2804–2832. [CrossRef]
33. Kang, S.; Veeravalli, B.; Aung, K.M.M. A Security-Aware Data Placement Mechanism for Big Data Cloud Storage Systems. In *Proceedings of the 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*; IEEE: New York, NY, USA, 2016; pp. 327–332.
34. JavaServer Faces Technology. Available online: https://www.oracle.com/java/technologies/javaserverfaces.html (accessed on 28 July 2026).
35. Jakarta Faces Specifications. Available online: https://jakarta.ee/specifications/faces/ (accessed on 20 July 2026).
36. Castro Medina, F.; Rodríguez Mazahua, L.; López Chau, A. *Desarrollo de Nuevos Métodos de Fragmentación Dinámica Para Bases de Datos Multimedia*; Instituto Tecnológico de Orizaba: Orizaba, Mexico, 2024.
37. Abdel Raouf, A.E.; Badr, N.L.; Tolba, M.F. Distributed Database System (DSS) Design over a Cloud Environment. In *Multimedia Forensics and Security*; Springer: Cham, Switzerland, 2017; Volume 115, pp. 97–116.
38. Joaquín Rodríguez-Arauz, M.; Rodríguez-Mazahua, L.; Leoncio Arrioja-Rodríguez, M.; Antonieta Abud-Figueroa, M.; Gustavo Peláez-Camarena, S.; Del Carmen Martínez-Méndez, L. Design of a Multimedia Data Management System That Uses Horizontal Fragmentation to Optimize Content-Based Queries. In Proceedings of the Tenth International Conference on Advances in Information Mining and Management, Lisbon, Portugal, 27 September–1 October 2020.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.