electronics

Article

A New Method of Dynamic Horizontal Fragmentation for Multimedia Databases Contemplating Content-Based Queries

Felipe Castro-Medina [1], Lisbeth Rodríguez-Mazahua [1,*], Asdrúbal López-Chau [2], Jair Cervantes [3], Giner Alor-Hernández [1], Isaac Machorro-Cano [4] and Mario Leoncio Arrioja-Rodríguez [1]

1 Division of Research and Postgraduate Studies, Tecnológico Nacional de México/I. T. Orizaba, Av. Oriente 9 852, Col. Emiliano Zapata, Orizaba C.P. 94320, Veracruz, Mexico; dci.fcastro@ito-depi.edu.mx (F.C.-M.); giner.ah@orizaba.tecnm.mx (G.A.-H.); mario.ar@orizaba.tecnm.mx (M.L.A.-R.)
2 Centro Universitario UAEM Zumpango, Universidad Autónoma del Estado de México, Camino Viejo a Jilotzingo Continuación Calle Rayón, Valle Hermoso, Zumpango C.P. 55600, Estado de México, Mexico; alchau@uaemex.mx
3 Centro Universitario UAEM Texcoco, Universidad Autónoma del Estado de México, Av. Jardín Zumpango, s/n, Fraccionamiento El Tejocote, Texcoco C.P. 56259, Estado de México, Mexico; jcervantesc@uaemex.mx
4 Faculty of Engineering, Universidad del Papaloapan, Circuito Central #200, Colonia Parque Industrial, Tuxtepec C.P. 68301, Oaxaca, Mexico; imachorro@unpa.edu.mx
* Correspondence: lrodriguezm@ito-depi.edu.mx

Abstract: The proper storage and management of multimedia data is a topic of great interest to industry and academia. Database fragmentation plays a fundamental role as a mechanism to guarantee cost reduction and improve response time performance in distributed data management environments. Multimedia database access patterns are constantly changing; due to this, it is important that the partitioning schemes also adapt to these changes. Dynamic fragmentation techniques offer this advantage and represent a reduction of the tasks that an administrator must perform and the complete autonomy to determine when to carry out a new fragmentation based on a cost model. This work proposes a new method of dynamic horizontal fragmentation for multimedia databases, including a way to contemplate content-based queries in the creation of new fragments. The use of content-based queries is on the rise, as multimedia elements are often presented within databases, and for this reason new fragmentation strategies must include this aspect to provide better-performing schemas. The method included in this research is placed within a current web application called XAMANA. We performed some experiments to demonstrate the effectiveness of our approach.

Keywords: dynamic fragmentation; CBIR; access patterns

check for updates

Citation: 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. https://doi.org/10.3390/electronics11020288

Academic Editor: George A. Tsihrintzis

Received: 29 November 2021
Accepted: 13 January 2022
Published: 17 January 2022

1. Introduction

Data fragmentation arises as a strategy to provide a new and adequate distribution unit in terms of performance. Chunks in a Distributed Database Management System (DDBMS) allow data to be stored close to the points of use, and thus each site manages only a portion of the database, reducing I/O requests. In addition, in a DDBMS scheme the location of the data reduces delays in remote access [1].

Relational tables can be partitioned either horizontally or vertically. The basis of horizontal fragmentation is the select operator where the selection predicates determine the fragmentation, while vertical fragmentation is performed utilizing the project operator [1].

This research uses horizontal fragmentation to divide a relation along its tuples, considering a cost model that uses predicates to obtain each fragment and assign it to the site where it produces the lowest cost under the current workload.

Database fragmentation can be performed statically or dynamically [2]. When a workload changes, databases in a distributed environment require redesigning their schema. When these changes in the workload occur, dynamic fragmentation can be carried out from the starting point; however, for highly dynamic systems this causes an overhead when redesigning. The best approach is to redesign the schema incrementally each time changes are detected or periodically at regular intervals of time [1]. Rodríguez et al. [2] show different disadvantages of the static approach and mention the dynamic approach as a solution to all of them.

This work applies horizontal fragmentation dynamically to determine when to perform new fragmentations over time in a distributed database, thus avoiding the significant amount of time spent by the database administrator observing the system to collect the frequencies of the operations performed, guaranteeing the use of real trends to reduce query execution time, adapting the scheme to changes in access patterns and making refragmentation a trivial, autonomous, and executed on-the-fly task.

Large amounts of complex data (e.g., image, sound, and video) are collected and organized in databases every day. These complex data come in different formats and cannot be sorted by content in the same way as conventional data (those represented as strings or numbers). This scenario, combined with the huge size and considerable growth of some complex data collections, poses new challenges for information retrieval [3]. The consideration of multimedia databases and CBIR (Content-Based Image Retrieval) makes this work a valuable study to face challenges and use fragmentation as an approach to improve the performance of this type of database. This article stands out for contemplating all the characteristics mentioned above in the new and complete fragmentation method applied to the database management systems that are most used in the literature [4].

This article is organized as follows: Section 2 shows the works related to dynamic horizontal fragmentation. The method proposed by this work is described in detail in Section 3. The evaluations and results are shown in Section 4. Section 5 mentions different aspects to consider in the application of the proposed method. Finally, Section 6 provides the conclusions and future research plans.

2. Related Work

To introduce this section, the work that laid the foundations for the development of the objectives of this work is presented. Our previous work in [4] showed an in-depth analysis of the approaches related to dynamic fragmentation methods in multimedia databases. The most current works related to the topics of interest were selected through a research methodology. The conclusion in [4] produced the objective of this new research since the platform where the horizontal fragmentation method was implemented was developed. In [4], three works [5–7] stood out for presenting a simple cost model for their implementation. Researchers in [5,6] carried out a vertical and horizontal fragmentation, respectively. Rodriguez-Mazahua et al. [7] presented a hybrid fragmentation method for multimedia databases. The analysis in [4] mentions that these works are excellent ways to perform fragmentation and it is concluded that the research performed helps researchers and practitioners with an overview of the state of the art of this type of work.

Pinto and Torres [8] provided a solution to the problem of dynamic data reassignment in a partitionable distributed database with changing access patterns. They used a public company subsidized by the Mexican government, CFE (Comision Federal de Electricidad, Federal Electricity Commission), as a case study. CFE has a distributed database in continuous operation, which causes data fragmentation that cannot be determined with the server turned off. Therefore, the authors performed a horizontal fragmentation with partial replication to achieve improved performance through access patterns. It was mentioned as a conclusion that the tests showed a good behavior of the method under a distributed structure of 16 computers.

Saad et al. [9] proposed a multimedia primary horizontal fragmentation approach and achieved this by analyzing the implications between predicates creating minterms to use in k-NN (K-Nearest Neighbor) and range query methods. A prototype was produced, implemented in C#, and named Multimedia Implication Identifier. The experiments satisfactorily obtained the expected result in a computational time that grows in a quadratic way for the analysis of the implications of similarity.

Hauglid et al. [10] showed how DYFRAM performs dynamic table fragmentation and replication based on recent historical access to maximize the number of local accesses and minimize remote ones. DYFRAM is a decentralized approach that takes place in a distributed database environment. This work achieves its objective by observing the access patterns of the sites to the tables. The evaluation presented began by obtaining the results through a workload simulator under two sites; later, a new scenario was presented simulating two highly dynamic workloads involving more sites. Finally, DYFRAM was implemented in a distributed database system (DASCOSA-DB).

Abdalla and Amer [11] presented a fragmentation strategy that sometimes uses partial replication to allocate fragments. The approach presented by the authors applies horizontal fragmentation in a relational database environment. To achieve this objective, a heuristic technique is used, since it evaluates the allocation of fragments to sites through a cost model. Finally, the heuristic model assigns the fragment to where it represents the highest cost. It is concluded that the proposed technique significantly improves the performance of distributed databases by eliminating frequent remote accesses and high data transfer costs between sites.

Bellatreche et al. [12] oriented their work to the inclusion of optimal fragmentation schemes of large relational data warehouses. Different schemes are statically selected using a genetic algorithm. It presents the primary choice for horizontal incremental data fragmentation and adapts to the operations and evolution of data warehouse workloads. The conclusion mentioned that the adaptation produced an incremental approach, which was compared with two other strategies and was shown to give a better optimization with the queries, reducing the maintenance cost.

Derrar and Ahmed-Nacer [13] carried out an investigation on horizontal fragmentation applied to data warehouses using the recent history of data access to minimize response time in OLAP query systems. In the first stage, the authors use histograms to analyze access to the fragments and apply refragmentation according to the new statistics collected. To evaluate this strategy, the APB-1 benchmark was used, which has a star scheme. It was implemented under Oracle 10G with 9000 tuples and 4-dimensional tables. The conclusions indicated that the results obtained are encouraging since the approach reduces the space occupied in memory and the query response time was shorter.

Herrmann et al. [14] presented Cinderella, an autonomous algorithm to carry out horizontal fragmentation in irregularly structured entities. Cinderella maintains the fragments while entities are added, modified, or deleted to increase local queries and reduce execution costs. The approach has the quality of being incremental and abandons the idea of a single fragmentation. The presented work produces entity-based or workload-based fragmentation schemes. The evaluation was carried out with the TPC-H benchmark in PostgreSQL 9.3 and showed favorable results when achieving the planned objectives.

Fetai et al. [6] presented Cumulus, a workload-based adaptive fragmentation protocol for applications with a high need for data consistency assurance. Cumulus is capable of fragmenting shared-nothing architectures, minimizing distributed queries, or even avoiding them. The work presented is adaptive and dynamic as it performs refragmentation at runtime. The evaluation was carried out with the TPC-C benchmark and a database of more than 10,000 tuples, showing the overall performance gain compared to systems where distributed transactions occur. Cumulus processes the workload and refragments before starting data redistribution, intending to eliminate costly and unnecessary reconfiguration.

Abdel et al. [15] presented a distributed database system on a cloud environment to perform dynamic fragmentation, allocation, and replication decisions based on access patterns to sites and the load involved in allocating or migrating fragments or replicas to these. To carry out the objective of this work, an MCRUD (Modified Create, Read, Update and Delete) matrix is built to determine the operations performed with each predicate on each site and each application. The results evaluate the costs of each attribute and are placed in the ALP (Attribute Locality Precedence) matrix. The predicates of the attributes with a higher value in the ALP matrix define the horizontal fragmentation scheme. The MCRUD matrix determines the sites where the fragments will be assigned and where they will be replicated. The evaluation shows cost reduction through the application of the scheme produced by the approach presented.

Serafini et al. [16] proposed Clay, an elastic algorithm capable of load balancing and distributing transactions through dynamic blocks called clumps. Clumps are created at runtime and based on the monitored workload. The authors compared Clay with graph fragmentation techniques and presented differences and key advantages that showed the proposed approach as a better option. Clay runs in an OLTP database environment and a reconfiguration engine that dynamically changes the data layer. The evaluation was carried out through three workloads: TPC-C benchmark, Product-Parts-Suppliers, and Twitter benchmark.

Lwin and Naing [17] described an algorithm for assigning non-replicated dynamic fragments in a distributed database system. This approach was presented in two parts: the pre-processing phase and the action phase. The pre-processing phase uses the log information table stored by each site and obtains information for the method. The action phase fragments, allocates, and migrates data under a horizontal technique. The evaluation included four sites with fully connected distributed database systems and 10,000 tuples in each. A total of 75 percent of response inquiries and 25 percent of update operations were executed.

Olma et al. [18] presented a logical raw files fragmentation scheme. This scheme allows for lightweight indexing on each fragment, highlighting benefits that are refined on the fly using an online random algorithm. The authors integrated this approach into a prototype in situ query engine called Slalom. The evaluation was carried out using synthetic and real workloads comparing Slalom response times against a traditional DBMS (Database Management System), a state-of-the-art in situ query processing engine, and adaptive indexing.

Castro-Medina et al. [19] introduced FRAGMENT, a web application to carry out static horizontal fragmentation and replication over a cloud environment. FRAGMENT proposes a scheme for assigning fragments to the sites where the current cost of each fragment is reduced the most. The authors validated their work within the Amazon Web Services (AWS) platform with a simulated data set and the TPC-E benchmark.

Rodríguez-Arauz et al. [20] collected historical data from the Orizaba Technological Institute and created a multimedia database with these. They used horizontal fragmentation to optimize data management and measured their performance using a cost model, which proved to be better than when horizontal fragmentation was not used.

Abebe et al. [21] developed MorphoSys, a distributed database system that makes decisions automatically on how to partition data, what to replicate, and where to place these partitioned and replicated data. The authors used two benchmarks to evaluate their approach: YCSB and TPC-C. The results show that MorphoSys improves the performance of the evaluated databases 900 times.

Ge et al. [22] introduced a set-based adaptive distributed differential evolution (S-ADDE) algorithm. S-ADDE adopts an island model to maintain population diversity. With the help of the S-ADDE database fragmentation result, the anonymity degree of the original dataset in each fragment increases. According to the analysis of the experimental results, the proposed S-ADDE algorithm is significantly better than the compared algorithms.

Different works were shown that apply horizontal fragmentation in different ways. Table 1 shows the comparison of the different approaches under five headings. It is analyzed: if the works present completeness, that is, everything necessary to be implemented; if they have characteristics to implement them easily; if they are based on a cost model to carry out fragmentation; if the fragmentation they perform is dynamic; and finally, if they contemplate the use of content-based queries. The works that do not present completeness, do not include diagrams, equations, or algorithms that fully describe the approach they present, thus making it difficult to replicate the research. The approaches that are not easy to implement contain elements, data sets, tools, or preconditions difficult to obtain under the context of development, budget, or delimitation of the goals of this research.

Table 1 shows that we aim to provide a horizontal fragmentation method that fulfills the five criteria.

Table 1. Comparison of related works.

Article	1	2	3	4	5
Pinto and Torres [8]		X	X	X	
Saad et al. [9]	X		X		X
Hauglid et al. [10]	X	X	X	X	
Abdalla and Amer [11]		X	X	X	
Bellatreche et al. [12]			X	X	
Derrar and Ahmed-Nacer [13]	X		X	X	
Herrmann et al. [14]	X	X	X		
Fetai et al. [6]			X	X	
Abdel et al. [15]	X	X	X	X	
Serafini et al. [16]	X	X	X	X	
Lwin and Naing [17]	X		X	X	
Olma et al. [18]	X		X	X	
Castro-Medina [19]	X	X	X		
Rodríguez-Arauz et al. [20]	X	X	X		X
Abebe et al. [21]	X	X	X		
Ge et al. [22]	X	X	X		
This work	X	X	X	X	X

(1) Completeness, (2) ease of implementation, (3) cost model, (4) dynamic, (5) CBIR.

3. Dynamic Horizontal Fragmentation Contemplating CBIR

To introduce the description of the developed method, it will be briefly presented where it was implemented, and the tools involved. The strategy is divided into two sections. The first section involves the static fragmentation which takes place within the XAMANA web application. The second section performs the dynamic fragmentation as part of a library ready to be implemented within the projects related to the table that will be fragmented.

XAMANA is a web application developed under the Java programming language with the JavaServer Faces framework. The objective of XAMANA is to host the three types of fragmentation (horizontal, vertical, and hybrid) and execute them in four database management systems widely used in industry and academia [4]: MySQL, PostgreSQL, Postgres-XL, and MongoDB.

Figure 1 shows the workflow of the static fragmentation method proposed in this work. As can be seen, content-based queries are part of the first steps of the diagram, as a new investigation will be developed when they are not present in the table to be fragmented.

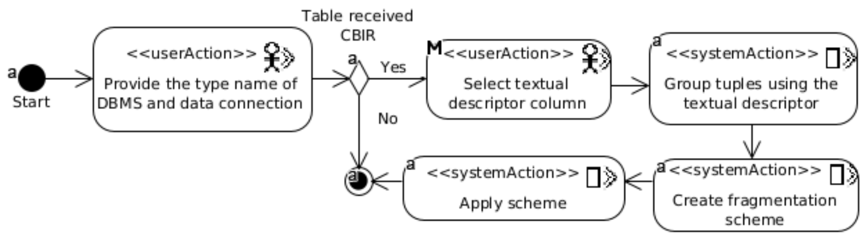

Start
<<userAction>>
Provide the type name of DBMS and data connection
Table received CBIR
Yes
No
<<userAction>>
Select textual descriptor column
<<systemAction>>
Group tuples using the textual descriptor
<<systemAction>>
Create fragmentation scheme
<<systemAction>>
Apply scheme

Figure 1. Static fragmentation method workflow contemplating CBIR.

The first step in the diagram is to obtain the name of the DBMS that the user utilizes (MySQL, PostgreSQL, Postgres-XL, or MongoDB) and the data to establish a remote connection with reading and writing privileges.

After verifying if content-based queries were executed on the table to be fragmented, descriptors hosted in the column that the user chooses are grouped. At this stage, other data are requested, which are detailed below, since they are part of the dynamic horizontal fragmentation workflow. The penultimate stage presents the scheme proposed by this approach. If the user agrees with the scheme shown, it will proceed to the last stage. The application of the fragmentation scheme is carried out under the connection data obtained in the first step of the workflow.

Figure 2 shows the workflow of the dynamic fragmentation method considering CBIR.

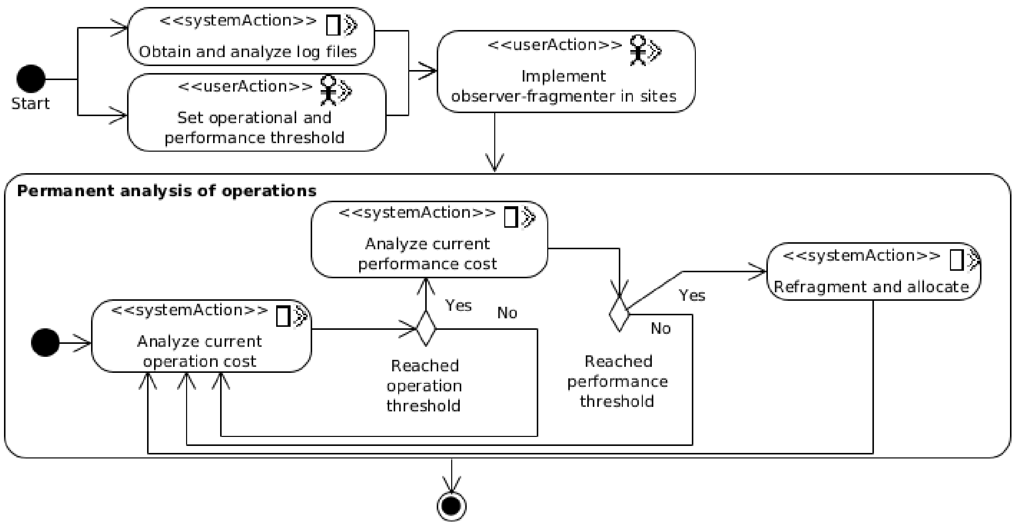

Start
<<systemAction>>
Obtain and analyze log files
<<userAction>>
Set operational and performance threshold
<<userAction>>
Implement observer-fragmenter in sites
Permanent analysis of operations
<<systemAction>>
Analyze current performance cost
<<systemAction>>
Refragment and allocate
<<systemAction>>
Analyze current operation cost
Yes
No
Reached operation threshold
Yes
No
Reached performance threshold

Figure 2. Dynamic fragmentation method workflow contemplating CBIR.

The first three activities of the dynamic fragmentation workflow take place inside XAMANA while the permanent analysis of the operations occurs inside a library named observer-fragmenter. The observer-fragmenter is responsible for dynamically adapting the distributed database schema to new trends in the workload at each site over time. Figure 2 shows the first stage, which consists of obtaining and analyzing the log files. These files contain valuable information about the addresses of the sites and the approximate number of operations carried out in each of them. The analysis of the files consists of obtaining the initial operation and performance thresholds, calculated by the frequency of operations, the type (create, read, update, and delete), and where they come from (local or remote). The user assigns a percentage for the performance and operation threshold, which when reached in each fragment will cause a refragmentation. The frequency of the fragmentation is inversely proportional to these percentages; a low value for these thresholds will trigger a new fragmentation more often, while a high value for these thresholds will cause fewer fragmentations. We decided that the DBA must fix these parameters for two reasons: (1) the DBA knows how often the database access patterns change, and (2) to provide him/her the ability to configure the frequency of the fragmentation. For instance, if 10 operations accessed the data located in fragment *i* in the last fragmentation, an operation threshold of 50% would indicate that six new operations must be executed in fragment *i* to exceed the current operation threshold, while a performance threshold of 100% would imply that the current value must be doubled to trigger a new fragmentation. In the third activity, it is observed that the user must implement the observer-fragmenter in each of his/her applications. The permanent analysis of the operations is carried out after implementing the proposed library in each site where queries are executed on the table of interest. The observer-fragmenter records the operations performed after fragmentation using the primary key and the type of operation and analyzes whether the operation and performance thresholds were reached. The last stage of the diagram shows the result of this approach which are sites with fragments of the original table. The user performs the operations on the fragments and in this way the execution cost and response time of the queries are reduced.

This work contemplates content-based queries under the suggested implementation of SURF (Speeded Up Robust Feature) descriptors under BoofCV [23] and k-NN queries. Under this approach, BoofCV uses all the descriptors to carry out the classification of the multimedia elements and identify the most similar. Under a database approach, all the descriptors of a set of images are retrieved to obtain an answer for each query, which is filtered by the BoofCV libraries. This results in the log files not containing the number of operations used. For this reason, the record of operations within the observer-fragmenter was considered, so that only the real operations are recorded after filtering BoofCV. Figure 3 represents the process of content-based queries under the BoofCV approach. In this research, we validate the proposed method using a multimedia database that contains SURF descriptors of images obtained by BoofCV.

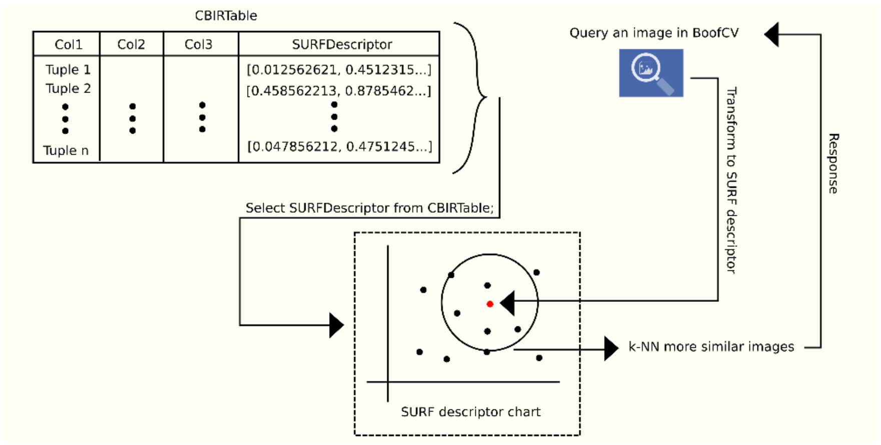


Figure 3. Process of content-based queries under the BoofCV approach.

By reducing the set of descriptors through this fragmentation approach, CBIRs are executed on smaller groups of SURF descriptors, thus improving the performance of these types of queries.

The operation threshold is calculated by adding the number of operations performed since the last fragmentation or refragmentation. Values representing the cost of each type of operation were determined to calculate the performance threshold. Table 2 shows the values assigned to each type of operation.

Table 2. Values for each type of operation.

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

Executing operations from remote sites is more expensive than running operations locally. This work considers in its cost model if the sites where the operations are carried out are local or remote. The performance threshold is obtained by the following equations:

$$PT_i = \sum_{j=1}^{N} (VTO_j \times RV_j \times S_j \times F_j) \tag{1}$$

$$S_j = \begin{cases} 1\,if\; RV_j = 1 \\ Sel_j \;\; otherwise \end{cases} \tag{2}$$

where *PT* is the performance threshold of fragment *i*, *VTO* is the value of the operation type according to Table 2, and *RV* is the remote value. *RV* takes the value of 1 when the operation *j* is local and 2 when it is remote, *S* is the size of the data involved in the queries, *Sel* is the selectivity of the predicates, *F* is the frequency of operations executed from the same site and *N* is the number of operations in the log file considering the fragmentation predicate.

When the operation and performance thresholds are exceeded by the operations performed, a new fragmentation is executed. The dynamic fragmentation that is carried out in the observer-fragmenter uses the graph of the Gaussian curve to normalize the use of the tuples throughout the fragment by the different sites. The diagram is divided into two and the site that represents a greater cost is determined on the tuples of the first and second part of the Gaussian diagram. Finally, the tuples of each half are assigned to their respective fragments and the fragments assigned to their sites.

As part of fragmentation techniques, it must be ensured that the semantics of the database do not undergo changes during fragmentation. Therefore, fragmentation techniques must meet three properties:

- Completeness. If a relation instance R is decomposed into fragments $F_R = \{R_1, R_2, \ldots, R_n\}$, each data item that is in R can also be found in one or more of R_i's;
- Reconstruction. If a relation R is decomposed into fragments $F_R = \{R_1, R_2, \ldots, R_n\}$, it should be possible to define a relational operator ∇ such that:

$$R = \nabla R_i, \ \ \forall R_i \in F_R$$

The reconstructability of the relation from its fragments ensures that constraints defined on the data in the form of dependencies are preserved;

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

The method proposed in this work ensures the three previous properties. Completeness is guaranteed in the dynamic and static part of the approach, creating schemas that contain all the tuples of the original relation, allowing each tuple to be found in the created fragments. This is achieved by adding a fragment to the horizontal fragmentation scheme defined by the negation of the fragmentation predicates. Reconstruction is ensured by creating schemas under predicates that include, together, all the tuples of the original relation, as well as the dependencies of the same defined in the beginning. This method ensures disjointness, as it produces schemes with completely different fragments from each other.

Algorithm 1, Observer, represents the pseudo-code to analyze the changes in each operation performed and trigger a new fragmentation. The Token that Algorithm 1 uses as a parameter is provided by the web application, which is an identifier for obtaining the entire set of fragments, tuples, statistics, sites, and connection data from the XAMANA database. As observed in lines 4 and 8, the operation and performance thresholds are obtained as a percentage and must be interpreted under their respective values to determine when they are reached. Dynamic fragmentation occurs when both thresholds are reached or exceeded, i.e., the frequency of execution of the refragmentations is directly related to the cost model.

Algorithm 1 uses Algorithm 2, Fragmenter, which is responsible for applying fragmentation. As parameters, Algorithm 2 receives the data of the fragment (or the table) to be fragmented and the Token. In both algorithms, different details are omitted to summarize them and make them easier to understand. One of them is the recording of the new statistics in the XAMANA database after the fragmentation has been performed. In Algorithm 2, the implementation of fragmentation in MongoDB is highly related to the language in which the algorithm is implemented; however, this part is described in a general way using instructions from the NoSQL (Not Only SQL) DBMS.

In the application of Algorithm 2 in the observer-fragmenter, implemented in Java, lines 3–11 were carried out as queries to the XAMANA database.

Algorithm 1 Observer

```
1  input: Token
2  fragments[]←Token.get fragments
3  for i from 0 to size of fragments[] do
4      nOperations←(fragments[i].Performance threshold percentage×
5                  fragments[i].Number of initial operations)/100
6      nOperations←nOperations+fragments[i].Number of initial operations
7      if fragments[i].Current operations number >= nOperations then
8          nPerformance← (fragments[i].Performance threshold percentage×
9                        fragments[i].Number of initial performance)/100
10         nPerformance←nPerformance+fragments[i].Number of initial performance
11         if fragments[i].Current performance number >= nPerformance then
12             x←fragmenter(fragments[i], Token)
13             write to activity file x
```

The variable oddEven, on line 13, allows fragmentation for odd tuples and then for even tuples by changing the vector in turn and the IP address to which the tuples will be assigned. The arrangement of the tuples from lowest to highest frequency and the separation of even and odd causes a representation of the graph of a Gaussian curve to normalize the data. Separating even and odd tuples is similar to dividing the Gaussian graph into two.

Line 20 takes into account MongoDB, since being a NoSQL DBMS, the operations are different. Lines 21 and 27 separate the formation of the CREATE statement for MySQL and PostgreSQL DBMSs, since the syntax is similar, but not the same. Another reason why the algorithm separates relational DBMSs is due to data types, as some are not included in both.

The fragmentation process handles four IP addresses. The algorithm in lines 7, 8, and 9 shows the purpose of three of them. The fourth address is the connection to the XAMANA server, which must be considered in the workflow of the fragmentation performed by the web application, since users must allow the remote connection from this IP to their servers.

Lines 38–54 represent obtaining the tuples from the source server to the destination server. Lines 57–62 show a more compact process for MongoDB, since it allows the copying of documents without defining a structure in the destination and source collection, even in new versions of MongoDB, when making a connection to a non-existent collection, the DBMS creates it automatically.

Algorithm 2 Fragmenter

```
1  input: fragment, Token
2  output: status
3  tupleIdFrequency[][]←get tuple identifiers and access frequency from fragment.Identifier
4  oddId[]←sort tupleIdFrequency[][] by frequency from lowest to highest, number them and get
5          odd
6  evenId[]←sort tupleIdFrequency[][] by frequency from lowest to highest, number them and
7          get even
8  connection←get connection data using Token
9  oddIP←get site that occupies the most odd tuples
10 evenIP←get site that occupies the most even tuples
11 inicialIP←get site where the fragment comes from
12 attributes[]←get fragment attributes
13 keyAttribute←get table identifier
14 for oddEven from 0 to 2 do
15     if oddEven is 0 then
16         vectorInTurn[]←oddId[]
17         ipInTurn←oddIP
18     else
```

```
            vectorInTurn[]←evenId[]
            ipInTurn←evenIP
        sentence←""
        if Token.get database type is not "MongoDB" then
            if Token.get database type is "MySQL" then
                foreach r←attribute in atributtes[] do
                    sentence←sentence + r.get name + " " + r.get datatype + "(" + r.get size + "), ";
                sentence←sentence − last two characters
                createSentence←"create table "+fragment.get name+"_"+(oddEven)+ " ("+
                                sentence+", PRIMARY KEY("+keyAttribute.get name+"))"
            if Token.get database type is "PostgreSQL" or "Postgres-XL" then
                foreach r←attribute in attributes[] do
                    sentence←sentence + r.get name + " " + r.get datatype + "(" + r.get size + ")";
                    if r.get name is keyAttribute.get name then
                        sentence←sentence + " PRIMARY KEY, "
                    else
                        sentence←sentence + ", "
                sentence←sentence-last two characters
                createSentence←"create table "+fragment.get name+"_"+(oddEven)+ " ("+
                                sentence+");"
            execute createSentence on ipInTurn using connection
            u←""
            foreach r←attribute in attributes[] do
                u←u + r.get name + ", "
            u←u-last two characters
            foreach y←item in vectorInTurn[] do
                query←"select " + u + " from " + fragment.get name + " where "
                                + keyAttribute.get name + "=" + y + ";"
                t[]←execute query in inicialIP using connection
                values←""
                for w from 0 to size of t[] do
                    values←values + t[w] + ", "
                values←values − last two characters
                query←"INSERT INTO " + fragment.get name + "_" + (oddEven)
                                + " VALUES (" + values + ");"
                deleteQuery←"DELETE FROM " +fragment.get name + "_" + (oddEven) +
                            " WHERE "+ keyAttribute.get name + "=" + y + ";"
                execute query in ipInTurn using connection
                execute deleteQuery in inicialIP using connection
            status←"Fragmentation carried out successfully"
        else
            execute createCollection(fragment.get name + "_" + (oddEven)) in ipInTurn using
            connection
            foreach y←item in vectorInTurn[] do
                where←create a new document with _id:y
                resultSet←find using where in inicialIP using connection
                insertOne with resultSet in new collection called fragment.get name + "_" +
                (oddEven)
                deleteOne using where in inicialIP
            status←"Fragmentation carried out successfully"
    return status
```

4. Results

To obtain the results of this approach we use a multimedia database called HITO (Historia del Instituto Tecnológico de Orizaba, History of the Technological Institute of Mexico) [20], implemented in MongoDB with more than 1.5 GB of storage. To carry out both workflows that perform static and dynamic fragmentation, the operations shown in Table 3 were executed on the collection "*multimedia_records*", shown in Table 4. The attribute *descriptor* contains the SURF descriptor of the images obtained by BoofCV. In this way, a log file is obtained that will be used to obtain statistics of the operations before the first fragmentation using XAMANA.

Table 3. Operations performed on the centralized multimedia database.

1	2	3	4	5
1	db.multimedia_records.find({tipo:"equipment"})	1	2	46.5
2	db.multimedia_records.find({tipo:"event"})	1	1	38
3	db.multimedia_records.find({descripcion:"building"})	1	1	6
4	db.multimedia_records.find({tipo:"personal"})	1	1	21
5	db.multimedia_records.find({tipo:"event"})	2	3	28.3
6	db.multimedia_records.find({tipo:"equipment"})	2	1	8
7	db.multimedia_records.find({tipo:"personal"})	2	1	27
8	db.multimedia_records.find({tipo:" personal"})	3	2	40
9	db.multimedia_records.find({tipo:" building"})	3	2	14.5
10	db.multimedia_records.find({tipo:"event"})	3	1	13

(1) Item, (2) operation, (3) site, (4) frequency, (5) execution time in milliseconds.

Table 4. Attributes of the *multimedia_records* table.

Attribute Name	Datatype
descripcion	String
tipo	String
imagen	String
validado	String
_id	String
descriptor	Array
nombre	String

The evaluation was carried out at three sites. The first site uses Ubuntu 20.04.3 LTS as the operating system, 4 GB RAM, and an AMD A6 processor with a clock frequency of 2.1 GHz. The second site uses Windows 10 as the operating system, 6 GB of RAM, and an Intel processor I7 with a 3 GHz clock frequency. The third site also uses Windows 10 as an operating system, 6 GB of RAM, and an AMD A8 processor with 2 GHz as a clock frequency. The centralized database is on the first site.

The data types shown in Table 4 were interpreted using Java. As seen in Table 3, read operations are performed from three different sites. Site 1 represents IP 192.168.200.5, site 2 represents IP 192.168.200.30, and site 3 represents IP 192.168.200.39. Figure 4 shows the initial XAMANA view requesting the type of database to be fragmented. As a complement to JavaServer Faces in the web application, the PrimeFaces library was used to add a rich open-source user interface.

The connection configuration view is shown in Figure 5. The data of the connection to the database are those mentioned in the preconditions that must be the same for all the sites of the same user (except the IP). The database address must allow remote connection from the XAMANA server IP. The username and password fields are not mandatory in databases where authentication is disabled. When placing the connection data, the list of tables found is automatically displayed.

The status of the connection is shown in the dialog in Figure 6. If the connection data do not allow communication between XAMANA and the user's database, a message informs that the data must be entered properly. If the connection data are correct, the fragmentation must be configured in the following view.

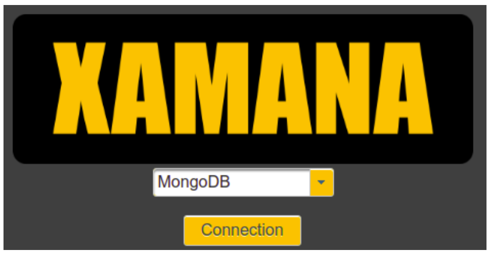
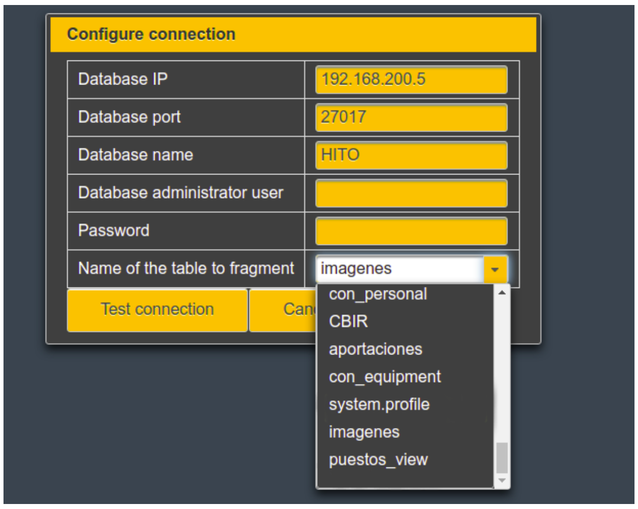
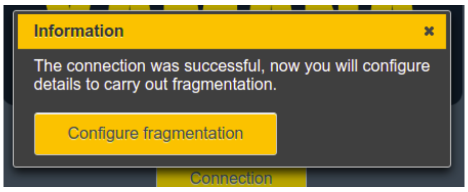


Figure 4. XAMANA web application home page.


Configure connection

Database IP	192.168.200.5
Database port	27017
Database name	HITO
Database administrator user	
Password	
Name of the table to fragment	imagenes

Figure 5. Setting up XAMANA's remote connection to the user database.


Information

The connection was successful, now you will configure details to carry out fragmentation.

Figure 6. Remote connection status after using the data to test communication.

The fragmentation configuration screen is shown in Figure 7. In Figure 7, horizontal fragmentation was selected, being one of the three types of fragmentation that the web application allows. The CBIR option, when selected, displays the dialog shown in Figure 8. The next field requested by the form presents the option to add the log files where the records of the operations previously executed are stored. For this evaluation, the thresholds were set with the percentages shown.

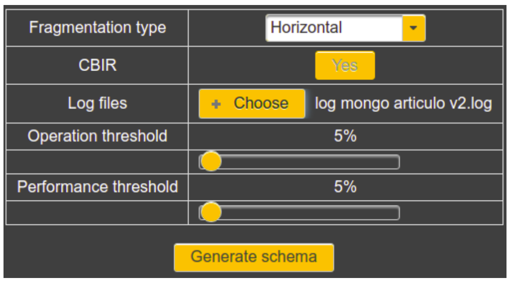
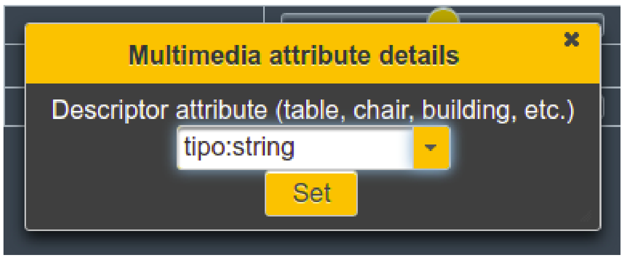

Fragmentation type	Horizontal
CBIR	Yes
Log files	Choose log mongo articulo v2.log
Operation threshold	5%
Performance threshold	5%

Figure 7. Configuration of the fragmentation to be performed.


Multimedia attribute details

Descriptor attribute (table, chair, building, etc.)

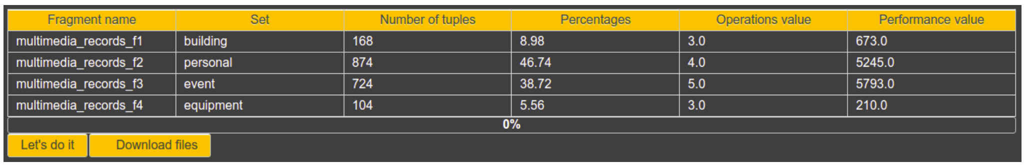
tipo:string

Set

Figure 8. Selection of the attribute that describes the multimedia attribute.

Figure 8 shows the selection of the descriptor attribute, by which the grouping of tuples and production of fragments will be carried out in the first stage of horizontal fragmentation. The descriptor attribute can be of any type of data, considering that it must form groups of tuples that share the same values. It is of interest to mention that the descriptor attribute contemplated in this stage does not share a similarity with the descriptors obtained by SURF. The use and implementation of BoofCV can help to obtain the values of the descriptor attribute, since using the k-NN algorithm with SURF descriptors allows for grouping multimedia content with similar characteristics.

Figure 9 shows the final structure of the fragments. The first column mentions the name of the fragment, the second column contains the set that defines the fragment, the third column represents the number of tuples that each fragment will contain, the fourth column shows the percentage that each fragment represents with respect to the entire table, the fifth column contains the operation value and the sixth column the performance value.



Fragment name	Set	Number of tuples	Percentages	Operations value	Performance value
multimedia_records_f1	building	168	8.98	3.0	673.0
multimedia_records_f2	personal	874	46.74	4.0	5245.0
multimedia_records_f3	event	724	38.72	5.0	5793.0
multimedia_records_f4	equipment	104	5.56	3.0	210.0

0%

Figure 9. Selection of the final structure of the fragments.

The values of the last two columns are obtained from the log file. The calculation of the initial values of operations and performance are obtained by searching the log for all operations that contain the set of each fragment in their predicates. The operations value is the number of operations found from that set. As an example, the *multimedia_records_f3* fragment, defined by the *event* set, had five operations on the log file (one from site 1, three from site 2, and one from site 3 according to Table 3). Therefore, *multimedia_records_f3* obtained a performance value of 5793, which is determined through the performance threshold cost model shown in Equation (1), adding the result of multiplying the value of the operation by the remote value by the variable S by the frequency of each operation for each set. Table 5 describes the performance value for each proposed fragment of Figure 9.

The observer-fragmenter contains a mechanism to save the operations performed and bypass the log file in future fragmentations. The initial values are broadly related to determining the execution of the next fragmentation.

The final fragment structure view provides two final buttons. The button to apply the schema physically on the table and the button to download files to start the dynamic fragmentation. For the PostgreSQL DBMS, in addition to the observer-fragmenter, functions and triggers are offered to maintain a suitable design between the fragments. Functions and triggers analyze changes to the original table structure; that is, adding, removing, and modifying attributes. If any of these operations are executed on the original table, they must also be carried out on the fragments related to this table. In this way, the design of the fragments is synchronized with the original design, properly maintaining the structure of the attributes.

Table 5. Description of the performance value for each proposed fragment of Figure 9.

Set	Operation Types	Remote Value	Operation
	Read	1	
	Read	2	
Event	Read	2	(1 × 1 × 1 × 1) + (1 × 2 × 724 × 3) + (1 × 2 × 724 × 1) = 5793
	Read	2	
	Read	2	
	Read	1	
Equipment	Read	1	(1 × 1 × 1 × 2) + (1 × 2 × 104 × 1) = 210
	Read	2	
	Read	1	
Building	Read	2	(1 × 1 × 1 × 1) + (1 × 2 × 168 × 2) = 673
	Read	2	
	Read	1	
Personal	Read	2	(1 × 1 × 1 × 1) + (1 × 2 × 874 × 2) + (1 × 2 × 874) = 5245
	Read	2	
	Read	2	

After performing the first fragmentation and starting the dynamic fragmentation, new operations are performed from sites 1, 2, and 3 on the observed fragments that trigger a new fragmentation. Table 6 shows the operations performed by the three sites. For example, two operations (creation and read) are performed from site 1 on document IMG1662 located in the fragment *multimedia_records_f1*.

Table 6. Operations performed on the fragmented multimedia database.

Search Item Id	Type of Operations	Site	Fragment Name
IMG1662	CR	1	*multimedia_records_f1*
IMG1875	R	1	*multimedia_records_f1*
IMG1581	RU	1	*multimedia_records_f3*
ObjectId("615e967df0f0e30bc461306e")	U	1	*multimedia_records_f2*
ObjectId("615e94d292bd5f6ac041f431")	C	1	*multimedia_records_f4*
ObjectId("615e94d892bd5f6ac041f432")	D	2	*multimedia_records_f4*
ObjectId("615e9684f0f0e30bc461307c")	RR	2	*multimedia_records_f2*
ObjectId("615e967af0f0e30bc461306b")	RRR	2	*multimedia_records_f2*
IMG1791	UU	2	*multimedia_records_f1*
IMG800	CD	2	*multimedia_records_f3*
IMG788	RU	3	*multimedia_records_f3*
IMG1640	U	3	*multimedia_records_f4*
IMG1790	C	3	*multimedia_records_f1*
ObjectId("615e99d2f0f0e30bc46133cf")	D	3	*multimedia_records_f2*
ObjectId("615e99c3f0f0e30bc46133c7")	RR	3	*multimedia_records_f2*

After performing the 24 operations shown in Table 6, the threshold analysis was carried out. The above operation and performance values shown in Figure 9 represent 100% of the thresholds. However, in this evaluation, the thresholds were set at 5% and this means that both values should be lower than the previous ones. Table 7 shows the thresholds reached by each fragment through the operations performed in Table 6. For example, for *multimedia_records_f3*, the previous operation and performance values according to Figure 9 are 5 and 5793, respectively; therefore, the operation and performance thresholds are 0.25 and 289.65 (5 × 0.05 and 5793 × 0.05), the current operation value is 6 (read and update from site 1, create and delete from site 2, and read and update from site 3 according to Table 6), and current performance value is 20 since the operations executed by site 1 are local and those by site 2 and 3 are remote.

Table 7. Threshold analysis in dynamic fragmentation.

Fragment Name	Previous Operation Value	Current Operation Value	Operation Value at 5%	Previous Performance Value	Current Performance Value	Performance Value at 5%
multimedia_records_f1	3	6	0.15	673	20	33.65
multimedia_records_f2	4	9	0.2	5245	21	262.25
multimedia_records_f3	5	6	0.25	5793	20	289.65
multimedia_records_f4	3	3	0.15	210	12	10.5

Table 7 describes that no fragment exceeds both thresholds of 5% except for the *multimedia_records_f4* fragment. To represent the operation of refragmentation, Figure 10 of the *multimedia_records_f4* fragment is shown.

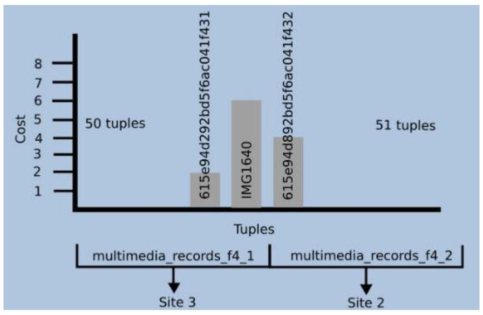


Figure 10. Dynamic horizontal fragmentation of *multimedia_records_f4*.

The tuples are placed on a diagram representing the Gaussian curve to normalize the costs. Figure 10 shows that the Gaussian curve will divide them to generate two new fragments: *multimedia_records_f4_1* and *multimedia_records_f4_2*. The observer-fragmenter looks for the place where the tuples of each fragment represent the highest cost to perform the assignment. Therefore, *multimedia_records_f4_1* is assigned to site 3 and *multimedia_records_f4_2* to site 2. The current values of operations and performance are considered as previous values in the next refragmentation operation. Under the new refragmentation scheme, the same operations presented in Table 3 are executed. Table 8 shows the comparison of the execution times.

Table 8. Comparison of execution times using the items of operations in Table 3.

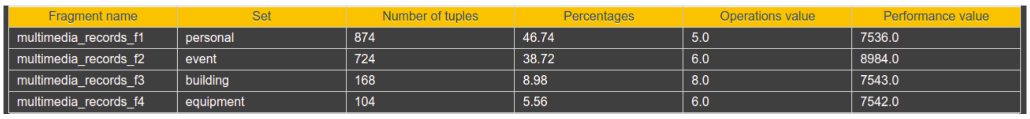
Item	Site	Execution Time without Fragmenting	Execution Time with this Approach
1	1	46.5	2
2	1	38	5
3	1	6	Less than 1
4	1	21	Less than 1
5	2	28.3	7.25
6	2	8	6
7	2	27	Less than 1
8	3	40	6.5
9	3	14.5	2
10	3	13	Less than 1

Table 8 shows a reduction in response times by 84.85% since remote communication was reduced and local communication increased. The initial operations found in the log were reads only and did not include content-based queries. A new experiment is carried out contemplating the retrieval of all descriptors to carry out content-based queries using the same attribute as multimedia description in the previous experiment. New operations are presented in the log simulating a new workload described in Table 9. The initial fragments produced by the web application are the same as in Figure 9, but the initial values of operations and performance are not. Figure 11 shows the new initial values. Table 10 presents the new operations performed in this experiment under the execution of the observer-fragmenter.

Table 9. Operations performed on the centralized multimedia database under a new workload.

1	2	3	4	5
1	db.multimedia_records.find({},{"descriptor:1"})	1	2	493.5
2	db.multimedia_records. update({_id:"IMG1790"},{$set:{"nombre": "cancha01modificado"}})	1	1	58
3	db.multimedia_records.deleteOne({_id:"IMG1790"})	1	1	61
4	db.multimedia_records.insert({_id:"IMG1790",tipo: "building",descripcion:"image of the chemistry group 1999",validado:false,nombre:"img20"});	1	1	41
5	db.multimedia_records.find({},{"descriptor:1"})	2	2	49.5
6	db.multimedia_records.find({validado:{$ne:null}},{imagen:0,descriptor:0})	2	1	13204
7	db.multimedia_records.update({_id: ObjectId("615e94dc92bd5f6ac041f437")},{$set:{"validado":"no"}}	2	1	73
8	db.multimedia_records.find({tipo:"event"})	2	1	13

(1) Item, (2) operation, (3) site, (4) frequency, (5) execution time in milliseconds.



Fragment name	Set	Number of tuples	Percentages	Operations value	Performance value
multimedia_records_f1	personal	874	46.74	5.0	7536.0
multimedia_records_f2	event	724	38.72	6.0	8984.0
multimedia_records_f3	building	168	8.98	8.0	7543.0
multimedia_records_f4	equipment	104	5.56	6.0	7542.0

Figure 11. New values of operations and performance under the operations of Table 9.

Table 10. Operations performed on the fragmented multimedia database in the new evaluation.

Search Item Id	Type of Operations	Site	Fragment Name
IMG103	CRRU	1	*multimedia_records_f4*
IMG110	UUU	1	*multimedia_records_f4*
IMG1663	RRD	1	*multimedia_records_f3*
ObjectId("615e9680f0f0e30bc4613072")	RRRD	1	*multimedia_records_f1*
ObjectId("615e94dc92bd5f6ac041f436")	RDC	1	*multimedia_records_f2*
IMG1570	UDC	2	*multimedia_records_f4*
IMG1561	RR	2	*multimedia_records_f4*
ObjectId("615e950692bd5f6ac041f460")	RU	2	*multimedia_records_f2*
ObjectId("615e96ccf0f0e30bc46130ba")	R	2	*multimedia_records_f1*
IMG1856	DC	2	*multimedia_records_f3*

This evaluation uses 0.4% in both thresholds and is applied under two sites. Table 11 represents the calculation of the thresholds for each fragment.

Table 11. Threshold analysis in dynamic fragmentation under the operations performed in Tables 9 and 10.

Fragment Name	Previous Operation Value	Current Operation Value	Operation Value at 0.4%	Previous Performance Value	Current Performance Value	Performance Value at 0.4%
multimedia_records_f1	5	5	0.02	7536	7	30.144
multimedia_records_f2	6	5	0.024	8984	13	35.936
multimedia_records_f3	8	5	0.032	7543	12	30.172
multimedia_records_f4	6	12	0.024	7542	34	30.168

Table 11 shows that the only fragment to exceed both thresholds was *multimedia_records_f4*. Refragmentation is carried out in the same way as in Figure 9, placing the most used tuples in the center of the diagram and producing two new fragments. The execution times of the operations performed in Table 9 are again compared in Table 12 to observe the different execution times in this new evaluation.

Table 12. Comparison of execution times using the items of operations in Table 9.

Item	Site	Execution Time without Fragmenting	Execution Time with this Approach
1	1	493.5	18.5
2	1	58	23
3	1	61	8
4	1	41	Less than 1
5	2	49.5	21.5
6	2	13,204	84
7	2	73	4
8	2	13	2

Table 12 presents an improvement in response times of 98.84%. Additionally, the work presented in [19] shows an operation simulator for MySQL. The functionalities of the query simulator were enriched to obtain log files simulating workloads and sites on the DBMS of this work. Execution times were obtained using a program written in Java before fragmentation and after fragmentation. A total of 200 queries on each site were simulated on the same MongoDB media collection to gain a more accurate percentage of improvement. Average response times resulted in an 86% reduction.

5. Discussion

The approach presented fulfills the hypothesis of improving performance by reducing the execution time of operations by assigning the fragments to the sites where they are most used. The implementation of CBIR using k-NN and SURF descriptors requires different strategies to be sought so that the task of reading all the descriptors and placing them in a graph to search for similar multimedia elements is a trivial task with low consumption of computational resources. Fragmenting a database is an arduous task, since in large databases the amount of data moving from one place to another is significant. Performing fragmentation without a system in place is a difficult task for database administrators and even more difficult to carry out dynamically. The dynamic fragmentation approach proposed in this work greatly facilitates the task of fragmentation in a precise and adequate way.

6. Conclusions

The results show that the same workload in a fragmented environment under this approach greatly improves response times. Costs are minimized due to reduced data transport and decreased access to irrelevant tuples. When traversing all descriptors to find similar images in content-based queries, access to irrelevant tuples is very large. This work proposes a completely innovative approach, since none of the related works address the different topics included in this research. In addition, this work stands out for contemplating the multimedia content in the CBIR, outperforming other static approaches, such as FRAGMENT [19].

As a future direction of this research, a new method for vertical fragmentation considering content-based queries on XAMANA will be implemented. In addition, an operations translator will be developed for the adaptation of programming code to the new schemes produced by this approach.

Author Contributions: Conceptualization, L.R.-M., F.C.-M. and A.L.-C.; data curation M.L.A.-R., L.R.-M. and F.C.-M.; funding acquisition L.R.-M. and G.A.-H.; software, validation, investigation F.C.-M., L.R.-M., I.M.-C. and G.A.-H.; methodology F.C.-M., L.R.-M. and J.C.; supervision and project administration L.R.-M., G.A.-H. and F.C.-M.; writing—original draft, F.C.-M.; writing—review and editing, L.R.-M., A.L.-C., J.C., G.A.-H., I.M.-C. 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 National Council of Science and Technology (CONACYT) and the Public Secretariat of Education (SEP) through the Sectorial Fund of Research for Education, grant number A1-S-51808.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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 National Council of Science and Technology (CONACYT) and the Public Secretariat of Education (SEP).

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

References

1. Özsu, M.T.; Valduriez, P. Distributed and Parallel Database Design. In *Principles of Distributed Database Systems*, 4th ed.; Springer Nature: Cham, Switzerland, 2020; Volume 1, pp. 33–84.
2. Rodriguez, L.; Li, X.; Cuevas-Rasgado, A.; Garcia-Lamont, F. DYVEP: An active database system with vertical partitioning functionality. In Proceedings of the 10th IEEE International Conference on Networking, Sensing and Control, Evry, France, 12 April 2013.
3. Fasolin, K.; Fileto, R.; Krugery, M.; Kasterz, D.; Ferreirax, M.; Cordeirox, R.; Trainax, A.; Traina, C. Efficient Execution of Conjunctive Complex Queries on Big Multimedia Databases. In Proceedings of the 2013 IEEE International Symposium on Multimedia, Anaheim, CA, USA, 11 December 2013.
4. Castro-Medina, F.; Rodríguez-Mazahua, L.; López-Chau, A.; Cervates, J.; Alor-Hernández, G.; Machorro-Cano, I. Application of Dynamic Fragmentation Methods in Multimedia Databases: A Review. *Entropy* **2020**, *22*, 1352. [CrossRef] [PubMed]
5. 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]
6. Fetai, L.; Murezzan, D.; Schuldt, H. Workload-driven adaptive data partitioning and distribution—The Cumulus approach. In Proceedings of the 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 1 November 2015.
7. Rodriguez-Mazahua, L.; Alor-Hernandez, G.; Cervantes, J.; López-Chau, A.; Sánchez-Cervantes, J. A hybrid partitioning method for multimedia databases. *Dyna* **2016**, *83*, 57–70.
8. Pinto, D.; Torres, G. On Dynamic Fragmentation of Distributed Databases Using Partial Replication. In Proceedings of the WSEAS International Conferences: IMCCAS'02, ISA'02, SOSM'02, MCP'02 & MEM'02, Cancun, Mexico, 12 May 2002; pp. 208–211.
9. Saad, S.; Tekli, J.; Chbeir, R.; Yetongnon, K. Towards Multimedia Fragmentation. *Adv. Databases Inf. Syst.* **2006**, *4152*, 415–429.
10. Hauglid, J.O.; Ryeng, N.H.; Nørvåg, K. DYFRAM: Dynamic fragmentation and replica management in distributed database systems. *Distrib. Parallel Databases* **2010**, *28*, 157–185. [CrossRef]
11. Abdalla, H.I.; Amer, A.A. Dynamic horizontal fragmentation, replication and allocation model in DDBSs. In Proceedings of the 2012 International Conference on Information Technology and e-Services, Sousse, Tunisia, 26 March 2012.
12. Bellatreche, L.; Bouchakri, R.; Cuzzocrea, A.; Maabout, S. Incremental Algorithms for Selecting Horizontal Schemas of Data Warehouses: The Dynamic Case. In *Data Management in Cloud, Grid and P2P Systems*; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8059, pp. 13–25.
13. Derrar, H.; Ahmed-Nacer, M. Exploiting data access for dynamic fragmentation in data warehouse. *Int. J. Intell. Inf. Database Syst.* **2013**, *7*, 34–52. [CrossRef]
14. Herrmann, K.; Voigt, H.; Lehner, W. Cinderella—Adaptive online partitioning of irregularly structured data. In Proceedings of the 2014 IEEE 30th International Conference on Data Engineering Workshops, Chicago, IL, USA, 4 April 2014.
15. Abdel, A.E.; Badr, N.L.; Tolba, M.F. Distributed Database System (DSS) Design over a Cloud Environment. *Multimed. Forensics Secur.* **2016**, *115*, 97–116.
16. Serafini, M.; Taft, R.; Elmore, A.J.; Pavlo, A.; Aboulnaga, A.; Stonebraker, M. Clay: Fine-Grained Adaptive Partitioning for General Database Schemas. *VLDB Endow.* **2016**, *10*, 445–456. [CrossRef]
17. Lwin, N.K.; Naing, T.M. Non-Redundant Dynamic Fragment Allocation with Horizontal Partition in Distributed Database System. In Proceedings of the 2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), Bangkok, Thailand, 24 October 2018; pp. 300–305.
18. Olma, M.; Karpathiotakis, M.; Alagiannis, L.; Athanassoulis, M.; Ailamaki, A. Adaptive partitioning and indexing for in situ query processing. *VLDB J.* **2020**, *29*, 569–591. [CrossRef]
19. Castro-Medina, F.; Rodríguez-Mazahua, L.; López-Chau, A.; Abud-Figueroa, M.; Alor-Hernández, G. FRAGMENT: A Web Application for Database Fragmentation, Allocation and Replication over a Cloud Environment. *IEEE Lat. Am. Trans.* **2020**, *18*, 1126–1134. [CrossRef]
20. Rodríguez-Arauz, M.J.; Rodríguez-Mazahua, L.; Arrioja-Rodríguez, M.L.; Abud-Figueroa, M.A.; Peláez-Camarena, S.G.; 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 Managemen, Lisbon, Portugal, 27 September 2020; pp. 15–21.

21. Abebe, M.; Glasbergen, B.; Daudjee, K. MorphoSys: Automatic Physical Design Metamorphosis for Distributed Database Systems. *VLDB Endow.* **2020**, *13*, 3573–3587. [CrossRef]
22. Ge, Y.; Cao, J.; Wang, H.; Chen, Z.; Zhang, Y. Set-Based Adaptive Distributed Differential Evolution for Anonymity-Driven Database Fragmentation. *Data Sci. Eng.* **2021**, *6*, 380–391. [CrossRef]
23. BOOFCV. Available online: https://boofcv.org/index.php?title=Main_Page (accessed on 17 September 2021).