Comput Syst Sci & Eng (2020) 2: 99–112

International Journal of
Computer Systems
Science & Engineering

Enhanced Schemes for Data Fragmentation, Allocation, and Replication in Distributed Database Systems

Masood Niazi Torshiz[1]*, **Azadeh Salehi Esfaji**[1]† **and Haleh Amintoosi**[2]‡

[1]*Department of Computer Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran*
[2]*Computer Engineering Department, Faculty of Engineering, Ferdowsi University of Mashhad, Iran*

With the growth of information technology and computer networks, there is a vital need for optimal design of distributed databases with the aim of performance improvement in terms of minimizing the round-trip response time and query transmission and processing costs. To address this issue, new fragmentation, data allocation, and replication techniques are required. In this paper, we propose enhanced vertical fragmentation, allocation, and replication schemes to improve the performance of distributed database systems. The proposed fragmentation scheme clusters highly-bonded attributes (i.e., normally accessed together) into a single fragment in order to minimize the query processing cost. The allocation scheme is proposed to find an optimized allocation to minimize the round-trip response time. The replication scheme partially replicates the fragments to increase the local execution of queries in a way that minimizes the cost of transmitting replicas to the sites. Experimental results show that, on average, the proposed schemes reduce the round-trip response time of queries by 23% and query processing cost by 15%, as compared to the related work.

Keywords: Distributed Database, Big Data, Vertical Fragmentation, Allocation, Replication

1. INTRODUCTION

Today, distributed databases have developed in business organizations because of their advantages such as consistency, scalability, availability, and accessibility [1].

With the growth of communication and information technology, DDBMS becomes increasingly essential, leading to the need for fast and efficient access to distributed databases [2]. As these databases are located in different servers and connected with different link speeds, they can profoundly impact the response time and subsequently, the transmission cost and performance of distributed databases. So, there is a need for an effective design of a distributed database in order to achieve the desired reliability and performance.

There exists considerable work such as [4–9] that aims at addressing the fragmentation or allocation/replication problem separately. However, few works such as [10,11,12,14], have addressed all three issues integrally. Moreover, most recent works have concentrated on minimizing the transmission cost, access cost, or the query processing cost and are not concerned about minimizing the delays incurred by transmission and processing times (including queuing delays), which are important for Internet-based systems [16]. Few works such as [14–16] have considered the importance of minimizing response time in distributed database design. Round-trip response time is defined as the time elapsed between the arrival of a query to

*E-mail: niazi@mshdiau.ac.ir
†E-mail: salehiAzadeh@mshdiau.ac.ir
‡Email: amintoosi@um.ac.ir, ORCID: 0000-0002-1447-8086, corresponding author

a network site and the time when the response to the query is received at the query source. In other words, round-trip response time consists of transmitting a query from its source site to a server site, processing the query and generating a response at the server site, and transmitting the response back to the source site [14]. Also, most works have proposed heuristic algorithms for addressing DDB fragmentation and allocation, and only a few have proposed mathematical programming formulations.

In this paper, we address the three above mentioned issues in a distributed database by proposing vertical fragmentation, allocation, and replication schemes to minimize the round-trip response time. The contributions are summarized as follows:

- We propose a vertical fragmentation scheme that partitions the data into fragments such that bonded attributes (i.e., accessed together by the queries) locate in a single fragment, thus reducing the access cost to those attributes by the queries. The proposed scheme utilizes a weighted graph (with attributes as the vertices and the bonds between attributes as edges) and partitions it to subgraphs (i.e., fragments) with maximum connectivity between the relevant vertices (i.e., attributes) of each partition. The graph partitioning is done in a way that prevents the creation of too small or too big partitions. The fragmentation scheme aims to minimize the query processing cost. More details can be found in Section 4.1.

- We propose a static allocation scheme that takes advantage of simulated annealing metaheuristic technique to solve the NP-hard problem of optimized allocation to minimize the round-trip response time. Instead of considering a random allocation sequence for the initial allocation in the simulated annealing process, the allocation scheme creates a targeted initial allocation pattern by considering the fragment access ratios and allocating fragments with higher access ratios to the sites with higher processing speeds. This targeted initialization will decrease the required time for the simulated annealing algorithm to find the optimal allocation pattern. Moreover, in order to avoid the bottleneck problem (allocating high demand fragments to low processing speed sites), the proposed allocation scheme considers the site capacity constraints.

- We propose a replication scheme that performs partial replication of fragments to increase the local execution of queries. The fragments are replicated in a way to minimize the cost of transmitting replicas to the sites. In order to find an optimal replication solution, the proposed scheme utilizes the simulated annealing technique and considers two constraints:

  i) a fragment is replicated to a site only if there is a need for the fragment on that site and

  ii) the fragment is replicated to a site only if the site capacity constraint is preserved.

The remainder of the paper is organized as follows: Section 2 reviews the works concentrating on vertical fragmentation, allocation, and replication in distributed database design. Section 3 expresses the preliminaries and basic concepts referenced throughout the paper. The detailed description of the proposed approach is explained in Section 4. Section 5 presents the simulation results and the validation of the proposed approach as well as a comparative. Finally, Section 6 concludes the paper.

2. RELATED WORK

Fragmentation and allocation have been known as the main procedures for reliable performance and an efficient design for a distributed database and investigated in many articles. The work in [4] developed an integrated methodology for fragmentation and allocation, which incorporated concurrence control and communication network cost in distributed environments. Authors in [11] proposed a clustering-based technique for vertical fragmentation and allocation in distributed database systems. Their proposed scheme created query clusters to form fragments. They assume that each fragment is a set of attributes accessed together by a particular query Similarly, authors in [12] proposed a heuristic approach to reduce transmission costs of distributed queries. They also proposed a site clustering algorithm to ensures the creation of highly-balanced clusters. They also suggested several advanced allocation scenarios with data replication consideration. The work in [17] proposed a new vertical fragmentation algorithm using a graphical technique and an Attribute Usage Matrix (*AUM*), which represents the essential queries whose primary purpose, unlike iterative binary partitioning methods, is to create all fragments by one iteration. The work in [18] proposed an algorithm to measure the similarity between any pair of attributes. This method clusters attributes into sub-relations, which are called fragments. For this purpose, the relations are divided into sub-relations at the design cycle. The works in. [19, 20] presented an objective function to evaluate the "goodness" of fragmentation algorithms. The work in [21] developed a vertical fragmentation approach where an attribute affinity table was used as input to the proposed approach. A dynamic table for fragmentation and allocation was proposed in [22], which monitors the access pattern of network sites to data tables and utilizes it to perform fragmentation, replication, and re-allocation to maximize the number of local accesses. The work in [23] presented a mathematical optimization model called *DFAR* that unifies the fragmentation, allocation and dynamical migration of data in distributed database systems considering the storage capacity of network sites. Their model utilizes the Threshold Accepting algorithm to solve the *DFAR* problem. The works in [14–16] presented a new mathematical model for the fragmentation and the allocation problem, called *VFA-RT*, which aims to minimize the round-trip response time of queries. *VFA-RT* model is made of a non-linear objective function and a group of constraints. In order to solve the model, Threshold Accepting *(TA)* and Tabu Search (*TS*) metaheuristic algorithms were used. The work in [24] proposed the Adaptive Distributed Request Window Algorithm (*ADRW*) to achieve fragmentation and dynamic allocation of data. This approach is compatible with the access patterns changes of requests for attributes and makes decisions on the replications according to their "read/write" requests for data and total servicing part. The purpose of this algorithm is to adjust data allocation patterns to reduce the total servicing cost of the full read/write requests of data. The work in [25] presented a genetic algorithm approach

Table 1 Summary of Related Work.

	Reference	[20]	[19]	[18]	[4]	[23]	[26]	[28]	[14]	[10]	[11]	[12]
Problem addressed	Fragmentation	*	*	*	*			*		*		
	Allocation				*					*		
	Fragmentation + Allocation					*	*		*		*	*
	Re-allocation											
	Replication									*	*	*
Value to minimize	Query Transmission Cost			*		*	*				*	*
	Query Processing Cost	*	*					*				
	Round-trip Response Time								*		*	*

to solve the combined problems of vertical fragmentation and access path selection. Choosing the access path is a kind of mechanism which is capable of conducting an effective search for the physical sites of data. The work in [26] presented a new heuristic approach for fragmentation. This approach reduces the transfer cost of fragments to different sites using a mathematical model. In this approach, fragmentation and allocation are done simultaneously. Authors in [7] propose a linear approach to distributed database optimization that gathers incremental online knowledge about data access patterns and database statistics for online re-allocation of the fragments in order to continually optimize the query response time. In [6], the authors proposed a method based on a particle swarm optimization algorithm to solve the data allocation problem that aims to minimize the query execution time and transaction cost. In [27,28] authors discussed the data allocation issue in the purpose of minimizing data transmission across network sites using an ant colony optimization algorithm. The proposed procedure in [8] was a vertical fragmentation model with the two-phase allocation process. Unlike most earlier studies, the tradeoffs between different allocation scenarios were discussed for finding an optimal way of attribute assignment over sites. However, the model presented in [9] was an extension for [8] and could considerably reduce communication costs and query response time.

The work in [5] considered the data allocation problem in distributed databases where the query execution strategy affects allocation decisions. Authors in [29] propose a vertical partitioning algorithm that uses graphical techniques and starts from the attribute affinity matrix by considering it as a complete graph. Then, forming a linearly connected spanning tree, it generates all meaningful fragments simultaneously by considering a cycle as a fragment.

Table 1 summarizes the most relevant and recent works discussed above. As can be observed, many works have only dealt with the fragmentation problem, and many works have addressed the integration of fragmentation and allocation problems. However, few works have considered addressing the fragmentation, allocation, and replication problems integrally. Replicating fragments has been shown to result in more reliability, accessibility, traffic reduction of network, increase of scalability, and better performance compared to the lack of replications [1, 22, 29]. In this article, we are going to propose solutions for all these three issues.

Moreover, most of the works have concentrated on minimizing the query processing/transmission cost, and only a few (e.g., [14]) have dealt with round trip time minimization. In our proposed allocation scheme, we also consider minimizing this parameter. It is worth mentioning that our work is different from the works in [14–16] since we address the replication problem as well as the fragmentation and allocation problems. Moreover, we have some innovations in the fragmentation and allocation schemes compared to other related works. More detail about the proposed schemes can be found in Section 4.

3. PRELIMINARIES AND ASSUMPTIONS

3.1 Basic Definitions:

Vertical fragmentation divides an original relation (DB table) into some sub-relations (fragments) in a way that the combination of the fragments generates the primary relation [1]. If R denotes a relation with a set of attributes (columns) $A = \{A_1.A_2.\ldots.A_L\}$, vertical fragmentation is partitioning R into some sub relations F_i, such that F_is are derived from Equation 1:

$$F_i = \prod_{PK,A_i} R \quad \forall\, A_i \in A$$
$$R = F_1 \infty F_2 \infty \ldots \infty F_N \tag{1}$$

Where $\prod$ is the projection operator of relational algebra [1], and PK is the primary key that should be replicated in all fragments. Relation R should also be reconstructable by applying the join operator ∞ on the resultant sub-relations (i.e., fragments), as illustrated in the above Equation. So, vertical fragmentation on a relation R is defined as determining sub-relations $F_1, F_2, \ldots, F_N$, such that query execution cost is optimized concerning some criterion (here, minimizing the query processing cost).

Since vertical partitioning puts in one fragment those attributes usually accessed together, there is a need for some measure that would define more precisely the notion of "togetherness" [1]. Query execution frequency (f) and access frequency (the frequency of accessing an attribute by a query) are two crucial factors that define this notion. For each query $Q_i (1 \leq i \leq K)$ and each attribute $A_j (1 \leq j \leq \text{L})$, we associate an *attribute access* value, which equals to 1 if query Q_i references attribute A_j, and zero otherwise. The set of all access values can be

Table 2 Notation Description.

Notation	Description
S	The set of sites (sites), $S = \{S_1, S_2, \ldots S_{SN}\}$
F	The set of fragments, $F = \{F_1, F_2, \ldots F_N\}$
L_{Fi}	Number of attributes in fragment $F_i (1 \leq i \leq N)$
L_{Sj}	Number of fragments in site $S_J (1 \leq j \leq SN)$
Q	The set of important queries, $Q = \{Q_1, Q_2, \ldots Q_k\}$
A	The set of attributes, $A = \{A_1, A_2, \ldots A_L\}$
AAM	The $K \times L$ matrix showing the attribute access values (which are either 1 or 0)
f_{qi}	The execution frequency of query Q_q on site S_i (expressed in queries/sec.)
f_q	The execution frequency of query Q_q
$1/M_q$	Mean length of query Q_q (expressed in bits/query)
$1/M_R$	Mean length of query response (expressed in bits/response)
C_{ij}	Transmission speed between site S_i and site S_j (expressed in bits/sec.)
C_j	The procession capacity of site S_j (expressed in queries/sec.)
Y_{jq}	The existence/non-existence of one or more attributes used by query q in site j (is either 1 or 0)
ABM	The $L \times L$ matrix showing the number of queries that have accessed two attributes together (i.e., bound attributes)
W_{ij}	The number of attributes existing in local fragment F_i and to which the query Q_j accesses.
R_{irj}	Set of relevant attributes not existing in local fragment F_i and must be accessed remotely by query Q_j in fragment F_r
n_{irj}	Total number of attributes that are in fragment F_i accessed remotely with respect to fragment Fr by query Q_j
R_{ij}	The numbebr of relevant attributes not existing in local fragment F_i and must be accessed remotely by query Q_j.
n_i	The number of attributes that exist in fragment F_i.
T_{ij}	Transmission cost between site S_i and site S_j.
MK_q	The number of executions of query Q_q
PK_q	The ratio of execution of query Q_q to the total number of queries' executions
CS_j	The storage capacity of site S_j.
AFM	The $L \times N$ matrix showing whether an attribute belongs to a fragment or not
QFM	The $K \times N$ matrix showing whether a query needs a fragment to be executed.
FSM	The $N \times SN$ matrix showing whether a fragment is allocated to a site
AR_j	The access ratio of all queries to Fragment F_j
AV	A vector of size N showing the access ratios (ARs) of all fragments

represented by a $K \times L$ matrix called AAM[1] as expressed by Equation 2.

$$AAM(Q_i, A_j) = \begin{cases} 1, & \textit{if an attribute } A_j \textit{ is accessed by } Q_i \\ 0, & Otherwise \end{cases} \tag{2}$$

Similarly, we define an *attribute bond* value that measures the strength of an imaginary bond between the two attributes. Attribute bond value represents the number of times two attributes are accessed together by all queries at all sites. The set of all bond values can be represented by a $L \times L$ matrix called ABM[2], as expressed by Equation 3.

$$ABM(A_i, A_j) = \begin{cases} \sum_{q=1}^{K} AAM(Q_q, A_i) * \\ \qquad AAM(Q_q, A_j) & i \neq j \\ 0, & Otherwise \end{cases} \tag{3}$$

Attributes that are accessed by queries are called relevant attributes, and every fragment that contains most of the relevant attributes is defined as the local fragment [25]. W_{ij} shows the number of attributes existing in local fragment F_i and to which the query Q_j accesses. The number of attributes not locating in the local fragment F_i must be accessed remotely by query Q_j in fragment F_r are defined by R_{irj}. Note that in a typical environment, there may be many queries being executed. However, typically, only important queries (for example, 20% of the whole active queries that have made 80% of data accesses) have been taken into consideration [1]. Table 2 presents a detailed description of the notations used in the paper.

3.2 Assumptions:

In this paper, we assume of having a client-server architecture where the server is responsible for performing the proposed fragmentation and allocation schemes, and clients (i.e., sites) store the fragments that are defined and allocated by the server. We also assume a static environment in which, the queries that are to be performed are read-only (i.e., do not modify the database) and are known beforehand (i.e., there exists information about what queries are going to be performed on what sites and what attributes are going to be accessed by these queries). We also assume that fragments are disjoint for all attributes except for the primary key PK, which should be repeated in all fragments of a relation (for reconstruction).

3.3 Cost Model

As mentioned previously, vertical fragmentation and allocation are to be done to minimize the query processing costs. The

[1] Attribute Access Matrix
[2] Attribute Bond Matrix

cost of a distributed query processing can be expressed by two factors: local query processing cost (cost of accessing irrelevant local attributes) and remote query processing cost (the cost of accessing the remote relevant attributes). In this article, we consider the cost model in which, the cost of executing operations such as select, project, and join are not considered. In other words, since CPU time is negligible in comparison with I/O time, we do not consider the processing cost (which includes the time of executing operations such as select, and join) and only consider the cost of accessing the attributes by these operations.

In vertical fragmentation, a query does not usually require retrieving all the attributes of a fragment during query processing. Each attribute that is not required by a query but exists in the local fragment causes irrelevant local attribute access cost. Attributes that are not required to be accessed by a query (but accessed because they reside within the retrieved fragment) are called irrelevant attributes. The existence of the irrelevant attributes in the local fragment may lead to the growth of the number of local access. This, in turn, may result in the rise of the number of disk access, and hence, the local query processing cost increases. Equation 4 expresses the irrelevant local attribute access cost, as described in [17]:

$$Cost_{local} = \sum_{i=1}^{N} \sum_{j=1}^{K} \left[f_j^2 \times |W_{ij}| \times \left(\frac{|W_{ij}|}{n_i} \right) \right] \quad (4)$$

Similarly, there are attributes that are required by the queries but do not exist in the local fragment. These attributes are called relevant remote attributes. A greater number of relevant attributes that are in the remote fragments may also lead to an increase in the remote query processing cost. [17] Equation 5 expresses the relevant remote attribute access cost [19]:

$$Cost_{remote} = \sum_{j=1}^{K} \min_{i=1,N} \sum_{\substack{r=1 \\ r \neq i}}^{N} \left[f_j^2 \times |R_{irj}| \times \left(\frac{|R_{irj}|}{n_{irj}} \right) \right] \quad (5)$$

So, the total query processing cost, denoted by *TCost*, is expressed by Equation 6, as mentioned in [19]. This parameter will be further used in Section 5 in order to evaluate the performance of the proposed fragmentation scheme.

$$TCost = Cost_{local} + Cost_{remote} \quad (6)$$

4. PROPOSED VERTICAL FRAGMENTATION, ALLOCATION AND REPLICATION SCHEMES

In the following, we describe the proposed fragmentation, allocation, and replication schemes. The fragmentation scheme partitions attributes into fragments to minimize the query processing cost. The allocation component then optimally allocates the fragments to the sites to minimize the round-trip response time. Once the allocation is done, data that are commonly accessed by queries are replicated on the query's local site to increase the locality of reference and reduce the communication cost. A detailed description of each component has been presented below.

4.1 Vertical Fragmentation Scheme

The fragmentation scheme is responsible for dividing a database into fragments to minimize the query response cost. As mentioned in Section 3.3, the query processing cost is affected by the cost of accessing irrelevant local attributes and the cost of accessing relevant remote attributes. So, in order to reduce these costs, it is required that relevant attributes which are accessed together by the queries are located in a fragment. The intuition behind this idea is that fragmenting relevant attributes together will decrease the number of irrelevant attributes within that fragment, thus reducing the irrelevant access cost. Besides, locally fragmenting relevant attributes reduces the need for a query to access them remotely, thus reducing the relevant remote attribute cost. The fragmentation process, as proposed by this component, is as follows.

In order to identify the attributes that are to be located within a fragment, we make use of *AAM*. As mentioned previously in Section 3, *AAM* defines whether a query accesses an attribute or not. Next, *ABM* is constructed using Equation 3. Remember that *ABM* defines the number of times two attributes are accessed together by all queries running on a site. A more detailed description of *AAM, ABM* and bond values has been presented previously in Section 3.

In the next step, Graph G is created based on *ABM* in which, vertices resemble attributes and edges connect those two vertices (i.e., attributes) that are bonded together. The weight of an edge between two bond attributes A_i and A_j is obtained from *ABM [i, j]*. Once the graph is created, it is partitioned into subgraphs. As mentioned above, putting highly-bonded attributes in one partition (i.e., fragment) results in the reduction of access cost. So, partitioning is done in a way that each subgraph has the maximum bond values between its vertices. These subgraphs are then considered as fragments. So, in order to do the partitioning, at the first step, we find the edges with the lowest weights and remove them from the graph provided that it does not lead to the graph disconnection. This process can be expressed as finding a maximal spanning tree for the graph, which connects all the graph vertices and includes the edges with higher weights (i.e., bond values).

Once the maximal spanning tree is constructed, we begin partitioning it to subgraphs. A useful parameter in partitioning is the partition size. Partition size is defined as the number of attributes that reside inside a partition. If the partition size is too large, it leads to an increase in the irrelevant local attribute access cost. The same stands for the partition size being too small, which leads to the increase in the relevant remote attribute access cost. In order to create subgraphs, we start removing the edges with the lowest weights. If there are multiple edges with equal weights, we remove the edge that partitions the graph into sub-partitions (i.e., subgraphs) with the least difference in their partition size. This will prevent the creation of too small or too large partitions. The partitioning is done until $N - 1$ edges are removed, resulting in the creation*of N* subgraphs. Each subgraph is considered as a fragment.

The output of the fragmentation component is the AFM[3] (expressed in Equation 7), which is a $L \times N$ matrix that shows whether an attribute belongs to a fragment or not.

[3] Attribute Fragment Matrix

$$AFM(A_i, F_j) = \begin{cases} 1, & \textit{if an attribute } A_j \textit{ belongs to Fragment } F_j \\ 0, & Otherwise \end{cases} \tag{7}$$

4.2 Allocation Scheme

Once fragments are created, the next step is allocating the fragments to the sites. Such allocation is done in a way to minimize the round-trip response time. In other words, we are looking for an optimal fragment allocation that minimizes the round-trip response time.

As described previously, round trip response time is defined as the time elapsed between the arrival of a query to a site and the time the query response is received at the query source. In other words, the average round-trip response time using is described by three terms: average transmission delay of queries incurred by their transmission from query sources to the servers, average processing delay of queries at the servers, and average transmission delay of queries response back to their sources. Specifically, the objective function, as described in [14], is minimizing round trip response time (*RRT)* described as Equation 8.

$$RRT = \frac{1}{\sum_j \sum_q \sum_i f_{qi} y_{jq}} \left[\sum_{ij} \frac{1}{\frac{M_q C_{ij}}{\sum_q f_{qi} y_{jq}} - 1} + \sum_j \frac{1}{\frac{C_j}{\sum_q \sum_i f_{qi} y_{jq}} - 1} + \sum_{ij} \frac{1}{\frac{M_R C_{ij}}{\sum_q f_{qi} y_{jq}} - 1} \right] \tag{8}$$

As mentioned before, the general problem of minimizing round trip response time is NP-hard. Therefore the proposed solutions are based on heuristics. In this paper, we have utilized the simulated annealing (*SA*) metaheuristic approach to find an optimal value for RRT. The *SA* algorithm starts from an initial solution and calculation of the objective function. It then improves the value of the objective function in order to search for an optimal solution.

In our proposed allocation component, instead of considering a random fragment allocation pattern as the initial solution, we provide an efficient initial allocation pattern that considers fragment priorities in the initial allocation. More details are described in Section 4.2.2.

4.2.1 Bottleneck Problem

In real situations, server sites may have different processing speeds. Considering the minimization of round-trip response time as an objective inherently leads to the selection of sites with higher processing speeds to decrease the processing delay. This may result in the heavily-loaded sites with a massive amount of traffic, which leads to the increase in the transmission delay of queries from query sources to the server sites. This is an example of a bottleneck problem.

In order to avoid the bottleneck, we assume that sites have storage capacity constraints, and this constraint should be considered in fragment allocation. In other words, the sum of the sizes of all fragments assigned to site S_j must not exceed the storage capacity of site $S_j (CS_j)$.

Let us assume that a_{tj} denotes whether attribute A_t is allocated on site S_j or not (if yes, equals to 1; if not, equals to zero), l_t is the length of attribute A_t in bytes, and CA is the cardinality of the relation R, then the mean size of all fragments on site S_j (in bytes), denoted as μ_{Sj} is calculated by Equation 9.

$$\mu_{Sj} = CA \sum_{t=1}^{L} l_t \cdot a_{tj} \tag{9}$$

The above-mentioned capacity constraint is then defined by Equation 10.

$$\mu_{Sj} \leq CS_j, \quad \forall j; \quad 1 \leq j \leq SN \tag{10}$$

This constraint should be considered both in the formation of the initial allocation pattern (as the initial solution) and during the execution of the *SA* algorithm to find the optimal allocation pattern.

4.2.2 Fragment Prioritization

Consider a situation in which, there is a fragment that is widely accessed by a large number of queries. If such a fragment is allocated to a site with low processing capability, it results in an increase in processing delay, which contradicts the allocation objective (i.e., minimizing the round-trip response time). So, there is a need to compute the access ratio of each fragment (by all queries) and give allocation priority to those with higher access ratios. In order to do so, the following steps should be done:

1. At first, we calculate the number of executions of query Q_q, denoted by MK_q, which is the sum of the execution frequency of query Q_q per all sites, as expressed by Equation 11.

$$MK_q = \sum_{i=1}^{SN} f_{qi} \tag{11}$$

2. We then calculate the ratio of the execution of query Q_q to the execution of all queries, denoted by PK_q, as expressed by Equation 12.

$$PK_q = \frac{MK_q}{\sum_{q=1}^{K} MK_q} \tag{12}$$

3. Next, we need to define whether the query needs a fragment or not. This is done by calculating a matrix called *QFM*[4] as expressed by Equation 13, in which, operator $\odot$ denotes the Boolean product.

$$QFM = AAM \odot AFM$$
$$QFM(Q_i, F_j) = \begin{cases} 1, & \textit{if query } Q_i \textit{ needs Fragment } F_j \\ 0, & Otherwise \end{cases} \tag{13}$$

4. In the next step, we compute the access ratio of each fragment F_i, denoted by AR_i, as shown in Equation 14.

$$AR_i = \sum_{q=1}^{K} QFM[q][i] \times PK_q \tag{14}$$

[4]Query Fragment Matrix

5. Based on the access ratios obtained from the previous step, we are now able to create an access ratio vector of size$N$,denoted by *AV*, in which, each element *AV[i]* is initialized by the access ratio of Fragment F_i, i.e., $AVi = AR_i$. The access ratios can be regarded as the priority of fragments in the allocation process.

6. Finally, in order to create an initial allocation pattern, we first create an ordered list of sites based on their processing speed and then allocate the fragments to the sites based on their priorities: the fragment with the highest priority (e.g., highest access ratio) will be allocated to the site with the highest processing speed. This step is done until all fragments are allocated. Note that in order to prevent the bottleneck problem discussed above, the capacity constraints of sites, as described in Section 4.2.1, should be considered.

   The result of the above steps is an initial allocation pattern of fragments to the sites, described as FSM[5], as expressed by Equation 15.

$$FSM(F_j, S_j) = \begin{cases} 1, & \textit{if fragment } F_i \textit{ is allocated to node } S_j \\ 0, & Otherwise \end{cases} \tag{15}$$

This initial allocation is then fed to the *SA* algorithm to find an optimal fragment allocation.

4.3 The Replication Scheme

As mentioned previously, fragment replication allows the retrieval queries to be processed locally and quickly, which results in the reduction of transmission time, and subsequently, the round-trip response time of query executions.

In our proposed replication method, we replicate fragments to the sites to minimize the total transmission cost of replicas. The total transmission cost ($Cost_{tr}$) is expressed by Equation 16.

$$Cost_{tr} = \sum_j \sum_k \sum_i T_{ij}\ Size_k FSM_{ki} X_{kj}, \quad \forall i \neq j \tag{16}$$

Where T_{ij} is the cost of transmitting a byte from site S_j to site S_j, $Size_k$ is the size of fragment F_k in bytes,*FSM* is the Fragment Site Matrix which is initial allocation pattern of fragments to the sites, and X_{kj} is a decision making variable, which is 1 or 0, indicating whether fragment F_k should be replicated to site S_j or not.

In the replication method mentioned earlier, there are a set of constraints that should be considered. The first constraint, as expressed by Equation 17, denotes that a fragment is replicated on a site if there is a need for the fragment on that site. In order to determine whether a fragment is needed on a site, we consider two parameters. The first parameter is $\varnothing_{qj}$, which equals 1 if the execution frequency of a query Q_q on a site S_j (i.e., f_{qj}) is greater than zero, and 0 otherwise. The second parameter is QFM_{qk}, which, as described by Equation 13, specifies whether a query Q_q needs a fragment F_k or not. The multiplication of these two parameters denotes whether a fragment F_k is needed by query Q_q on site S_j. So, if a fragment is not needed by any of the queries running on a site, X_{kj} (which indicates whether fragment F_k should be replicated to site Sj) should be zero. The second constraint is similar to the one expressed by Equation 10. Replicas are stored in a site as long as the storage capacity constraint of the site is not violated. In other words, the fragment will be replicated on a site if at least one single access to the fragment has been done AND there exists enough storage on that site to store the replicated fragment.

In order to find an optimal solution for the replication of fragments, the Simulating Annealing Algorithm (*SA*) has been used. The algorithm begins with an initial answer of X_{kj} for the replication of fragments. At every iteration, the cost of the obtained replication solution is calculated (considering the constraints mentioned above) and compared with the previous one. The process is continued until an optimal solution that minimizes $Cost_{tr}$ is obtained.

[5]Fragment Site Matrix

5. NUMERICAL EXPERIMENTS

In this section, we evaluate the performance of our proposed schemes. First, we explain the experimentation setup, the scenarios we consider for performance evaluation, and the datasets we used in experiments in Section 5.1. We then make an initial analysis of our proposed fragmentation scheme in Section 5.2, based on the cost model mentioned in Section 3.3. Then, we compare our proposed schemes with other methods in Section 5.3.

$$X_{kj} \leq \sum_q QFM_{qk} \times \varnothing_{qj} \qquad \forall k, j \tag{17}$$

5.1 Experiment Setup

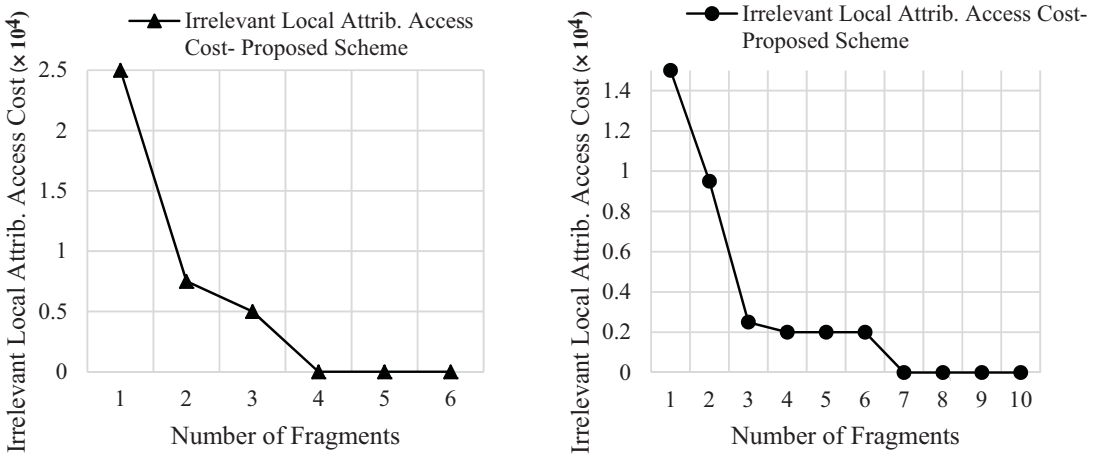
We implemented the proposed schemes in Matlab 8.1 and conducted a series of experiments to evaluate their performance. For these experiments to be done, we randomly created 100 instances and grouped them into five different scenarios S1 to S5 (each having 20 instances) such that the instances in each scenario are similar to each other. We aimed to create scenarios with different loads (i.e., number of queries) and capacities (i.e., number of sites), from fewest (S_1) to the highest (S_5). Table 3 shows the data used to generate the instances which are variable coefficients for expressions in Section 4. It is worth mentioning that the data values presented in this table are typical values that can be found in real cases.

5.2 Cost Analysis

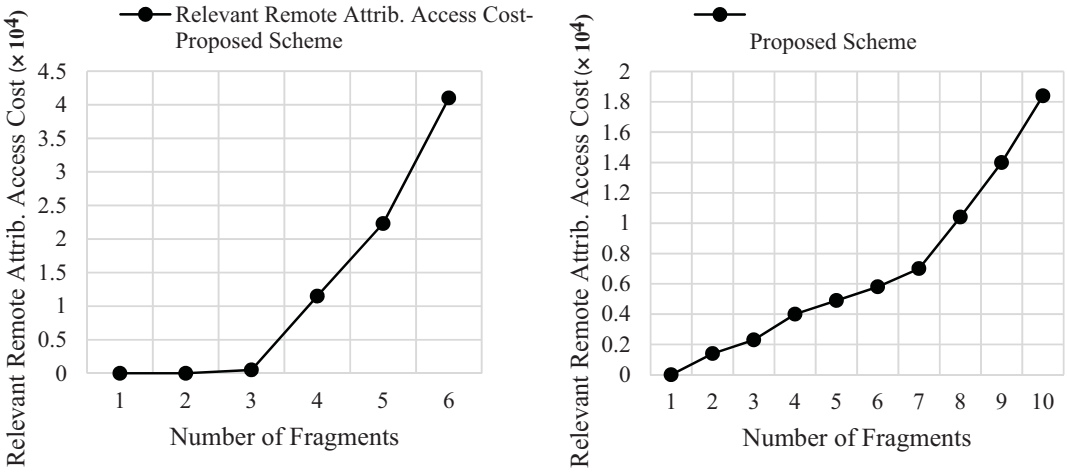
Figure 2 demonstrates the behavior of the two components of the query processing cost, as mentioned in Equations 4 and 5 (i.e., local irrelevant attribute access cost and remote relevant attribute access cost) as a function of the number of fragments for two scenarios S1 and S2. As demonstrated in figure 2.a, the increase in the number of fragments results in the reduction of irrelevant local attribute access cost. This is because when the fragments are few, they each contain a higher number of local

Table 3 The data values for generating the instances.

Scenario	No of Attributes L	No. of Queries K	No. of sites SN	Execution Frequency f_{qi}	AAM	Processing Capacity C_j	Storage Capacity CS_j	Transmission Speed C_{ij}	Transmission Cost T_{ij}	Length of Attribute l_t	Mean Query Length $1/Mq$	Mean Response Length $1/MR$
S1	6	4	3	2–60	0–1	50–500	100- 200	100000–400000	0–25	4–12	1000	5500
S2	5	5	3	5–50	0–1	50–500	110–245	100000–400000	0–25	4–12	1000	5500
S3	5	7	3	5–150	0–1	50–500	100–312	100000–400000	0–25	4–12	1000	5500
S4	10	8	4	0–19	0–1	50–500	90–230	100000–400000	0–25	4–12	1000	5500
S5	20	15	6	5–150	0–1	50–500	122–400	100000–400000	0–25	4–12	1000	5500



a) Evolution of irrelevant local attribute access cost in S1 and S2 scenarios



b) Evolution of relevant remote attribute access cost in S1 and S2 scenarios

Figure 2 The impact of the number of fragments on attributes access costs.

attributes and so, the local attribute access cost will be high. On the other hand, when the number of fragments increases, we have a fewer number of attributes in each fragment. So, when a query gets access to a fragment, it will encounter a fewer number of irrelevant attributes. As the number of fragments increases, the reduction of the number of irrelevant attributes continues until it reaches zero, as shown in figures 2.a. In contrast, an increase in the number of fragments leads to the increase in the number of relevant remote attributes, thus increasing the irrelevant remote attribute access cost, as illustrated in Figure 2.b.

5.3 Evaluation Result

In the following, we compare the performance of our proposed model with other related models for the different S1 to S5 scenarios.

5.3.1 S1 Experiment Results

To evaluate the proposed fragmentation and allocation schemes, we consider the cost model described in Section 3.3. First, in order to obtain the optimal number of fragments, we evaluate the irrelevant local attribute access cost and relevant remote attribute access cost for different number of fragments, as illustrated in Figure 3.a. The optimal number of fragments is acquired when remote and local attribute access cost curves meet, which is 2 for this scenario. Figure 3.b confirms that the least amount of query processing cost is acquired when the number of fragments is 2.

Next, we compare the performance of the proposed model with the *VFA-RT* model [14–16] based on the round-trip response time, expressed in Equation 8. Remember from Section 4.2 that the aim of the allocation process is allocating fragments into sites in a way that minimizes the round-trip response time of queries. Figure 3.c illustrates the round-trip response time obtained from running the experiment on 20 instances of the S1 scenario. As this figure shows, our proposed model has shown better performance in regards to obtaining less amount of round-trip response time. On average, the proposed approach has resulted in the 26% reduction of round-trip response time, as compared to the *VFA-RT* method, for S1 scenario.

5.3.2 S2 Experiment Results

Figure 4 illustrates the query processing cost and the round-trip response time for the second scenario S2, respectively. As figure 4.a shows, the minimum amount of query processing

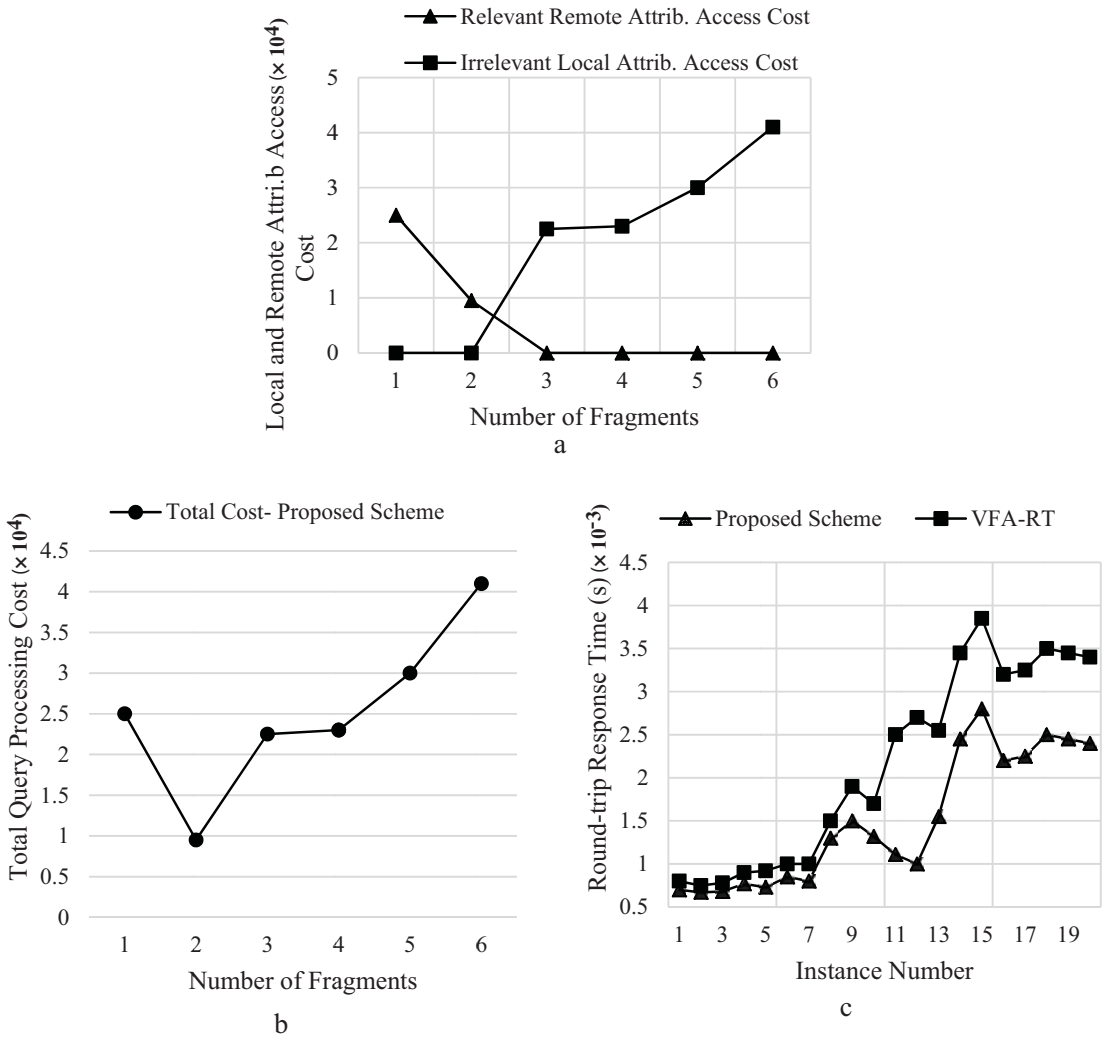


Figure 3 S1 Experimental Results.

cost is obtained when the fragment number equals 2. Figure 4.b compares the performance of the proposed model and the *VFA-RT* model based on the round-trip response time for different instance numbers in S2. Again, the proposed model has achieved better outcomes resulting in a 30% reduction of the average round-trip response time.

5.3.3 S3 Experiment Results

Figure 5.a illustrates the query processing cost and the number of optimal fragments for scenario S3, which is 2. Figure 5.b compares the performance of the proposed schemes with *VFA-RT* based on the round-trip response time of the queries for 20 instances of S3. As has been shown in this figure, applying the proposed fragmentation and allocation scheme has resulted in a 15% reduction of round-trip response time.

5.3.4 S4 Experiment Results

Figure 6.a illustrates the query processing cost obtained with different fragment numbers. As obtained from this figure, the optimal number of fragments which leads to the minimum cost equals to 3. Figure 6.b demonstrates the comparison results between the proposed allocation model and the *VFA-RT* model based on the round-trip response time for different instance numbers. As shown in this figure, the proposed allocation scheme method outperforms the *VFA-RT* model in regards to less amount of round-trip response time. On average, our proposed model has acquired a 27% reduction in round trip response time, compared to the *VFA-RT* model.

5.3.5 S5 Experiment Results

Figure 7.a shows the processing cost of queries in regards to the different numbers of fragments for scenario S5. According to Table 3, the number of queries and attributes in S5 are the highest numbers among all. The optimal number of fragments is 3. Figure 7.b demonstrates the evolution of round-trip response time for 20 instances in S5. A comparison between the results of the proposed method and *VFA-RT* indicates that the proposed approach has caused a 21% reduction in the average round-trip response time.

5.3.6 Query Processing Time Evaluation

As mentioned previously, fragmentation aims to partition attributes into fragments in a way to minimize the processing cost of the queries needing those attributes. In Table 4, we compare the query processing cost of the proposed fragmentation scheme

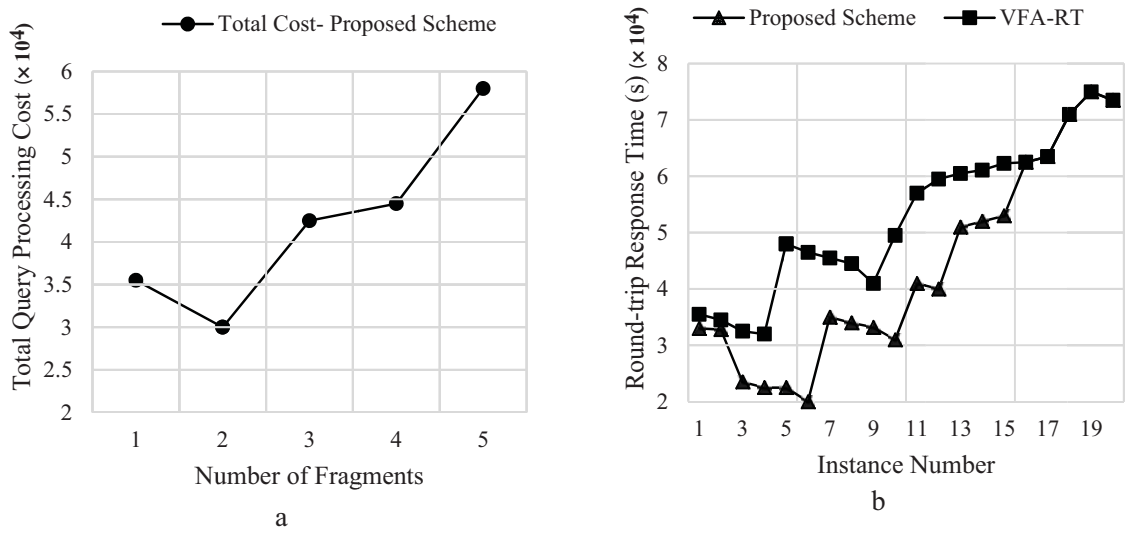


Figure 4 S2 Experimental Results.

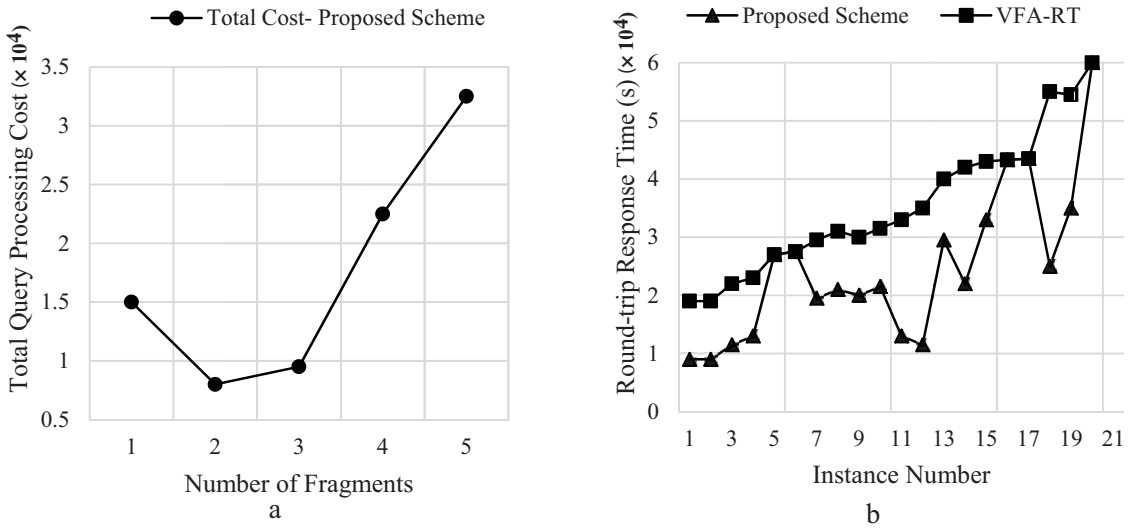


Figure 5 S3 Experimental Results.

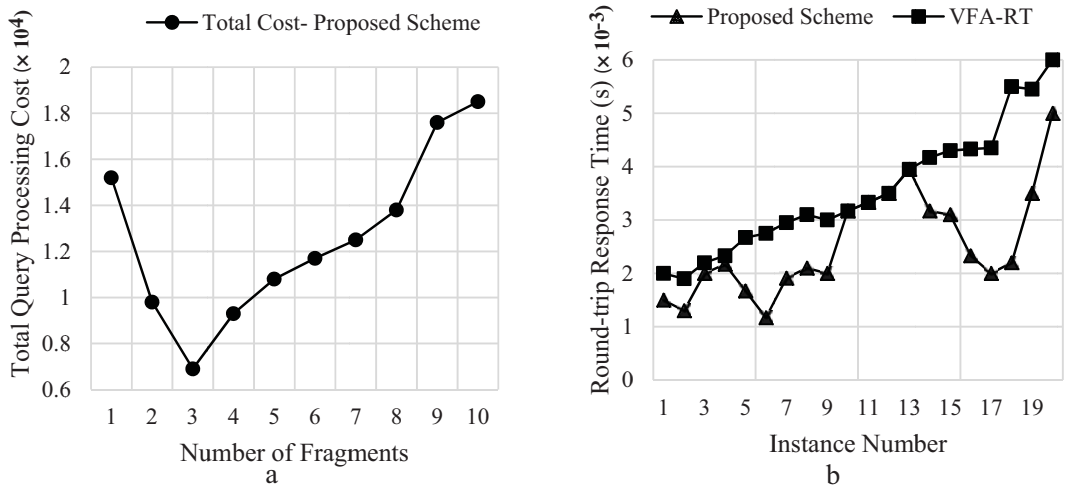


Figure 6 S4 Experimental Results.

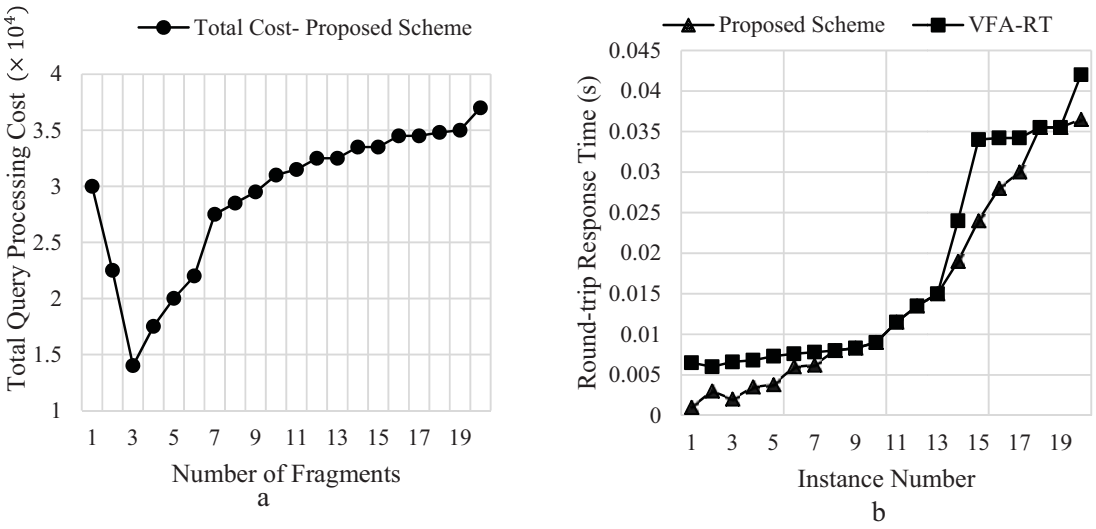


Figure 7 S5 Experimental Results.

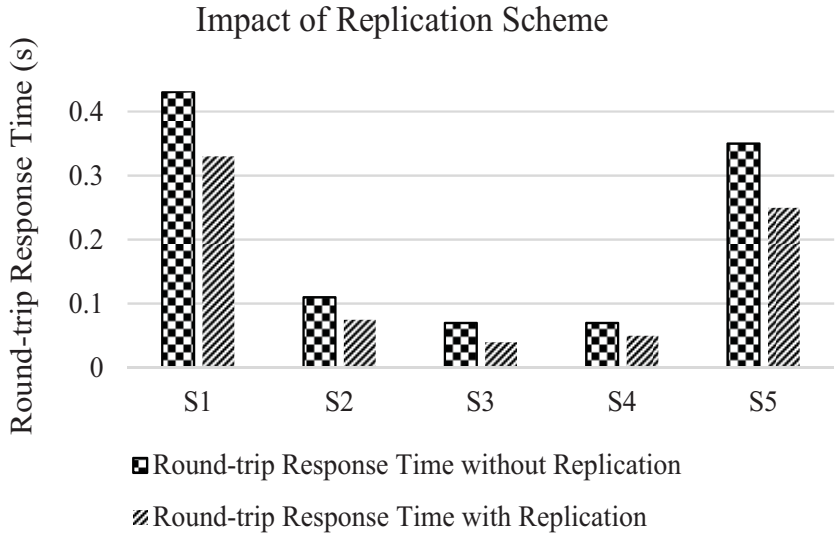


Figure 8 Impact of the Proposed Replication Scheme.

Table 4 Query Processing Cost for the Proposed Fragmentation Scheme and *VFA-RT*.

Scenario	Query Processing Cost ($\times 10^4$)	
	VFA-RT	Proposed Fragmentation Scheme
S1	1.2144	0.9564
S2	0.3145	0.2993
S3	1.3297	0.8348
S4	0.9564	0.6846
S5	1.9821	1.4901

and the *VFA-RT* model for all S1 to S5 scenarios. As has been shown in this table, the proposed fragmentation scheme has shown better performance in comparison with *VFA-RT* model and acquired less query processing cost.

5.4 Impact of the Proposed Replication Scheme

As mentioned previously in Section 4.3, once fragments are allocated to the sites, the replication scheme replicates some fragments, based on some criteria, with the aim of local execution of queries and reducing both the communication cost between sites and the queries' execution time. In order to observe the impact of such replication, we evaluate the round-trip response time for different S1 to S5 scenarios in two different situations: the situation where the replication scheme is applied and the one without replication. The results that have been shown in Figure 8 demonstrate that replicating the fragments will cause a substantial reduction in round trip response time for all scenarios.

6. CONCLUSION

With the growth of information technology and computer networks, there is a vital need for optimal design of distributed databases. Three main challenges in the design of a distributed database are fragmentation, data allocation, and replication. In this article, we present new approaches for vertical fragmentation, allocation and replication for distributed databases. The proposed vertical fragmentation scheme partitions the data into fragments such that those bonded attributes (i.e., accessed together by the queries) are located in a single fragment, thus reducing the access cost to those attributes and minimizing the query processing time. The allocation scheme utilizes the benefits of the simulated to minimize the round-trip response time of queries running on the sites. We also propose a replication that aims to perform partial replication of fragments to increase the local execution of queries. We compare the performance of our proposed schemes with other related work, considering different scenarios. Results show that the proposed fragmentation scheme can reduce the query processing cost by 15% as compared to the *VFA-RT* model. Moreover, the allocation scheme achieves a 23% reduction (on average) in the round-trip response time of queries. Considering the replication scheme also results in a 10% reduction in the round-trip response time, as compared to the situation where replication has not been considered.

REFERENCES

1. Özsu, M.T. and P. Valduriez, Principles of distributed database systems. 2011: Springer Science & Business Media.
2. Nashat D, Amer A.A. A Comprehensive Taxonomy of Fragmentation and Allocation Techniques in Distributed Database Design. ACM Computing Surveys. 2018. 51(1).
3. Iacob, N., Data replication in distributed environments. Annals-Economy Series, 2010. 4: p. 193–202.
4. Tamhankar, A.M. and S. Ram, Database fragmentation and allocation: an integrated methodology and case study. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 1998. 28(3): p. 288–305.
5. Apers, P.M., Data allocation in distributed database systems. ACM Transactions on Database Systems (TODS), 1988. 13(3): p. 263–304.
6. Mostafa Mahi, Omer Kaan Baykan, Halife Kodaz, A new approach based on particle swarm optimization algorithm for solving data allocation problem, Applied Soft Computing, Volume 62, 2018, Pages 571–578.
7. S. Darabant et. al, A linear approach to distributed database optimization using data reallocation T2 - 2017 25th International Conference on Software, Telecommunications and Computer.
8. Amer, A.A. ; Abdalla, H.I.,An integrated design scheme for performance optimization in distributed environments, International Conference on Education and e-Learning Innovations (ICEELI), 2012. p: 1–8.
9. Hassan I. Abdalla, Ali A. Amer, and Hassan Mathkour. A novel vertical fragmentation, replication and allocation model in DDBSs. J. Universal Computer Science. 2014. 20(10). p. 1469–1487.
10. Raouf, A.E.A., N.L. Badr, and M. Tolba. An optimized scheme for vertical fragmentation, allocation and replication of a distributed database. in 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS). 2015. IEEE.
11. Sewisy, A., Amer, A. and Abdalla, H. (2017). A Novel Query-Driven Clustering-Based Technique for Vertical Fragmentation and Allocation in Distributed Database Systems. International Journal on Semantic Web and Information Systems, 13(2), pp.27–54.
12. Abdalla, H.; Artoli, A.M. (2019). Towards an Efficient Data Fragmentation, Allocation, and Clustering Approach in a Distributed Environment. Information, V10, 112.
13. Amer A A, Mohamed M H, Al Asri K A. (2019). ASGOP: An Aggregated Similarity-Based Greedy-Oriented Approach for Relational DDBSs Design. Heliyon, (In press).
14. Pazos, R.A., et al., Minimizing roundtrip response time in distributed databases with vertical fragmentation. Journal of Computational and Applied Mathematics, 2014. 259: p. 905–913.
15. Vázquez, G. and J. Pérez. Modeling the nonlinear nature of response time in the vertical fragmentation design of distributed databases. in International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008). 2009. p. 605–612. Springer.
16. Vázquez, G. and J. Pérez. Vertical fragmentation design of distributed databases considering the nonlinear nature of roundtrip response time. in International Conference on Knowledge-Based and Intelligent Information and Engineering Systems. 2010. Springer.
17. Navathe, S., et al., Vertical partitioning algorithms for database design. ACM Transactions on Database Systems (TODS), 1984. 9(4): p. 680–710.
18. Abuelyaman, E.S., An optimized scheme for vertical partitioning of a distributed database, International Journal of Computer Science and Network Security (IJCSNS), 2008. 8(1): pp. 310–316.
19. Chakravarthy, S., et al., An objective function for vertically partitioning relations in distributed databases and its analysis, Distributed and parallel databases, 1994. 2(2): pp. 183–207.
20. Muthuraj, J., et al. A formal approach to the vertical partitioning problem in distributed database design, in Proceedings of the second international conference on Parallel and distributed information systems, IEEE Computer Society Press, 1993. pp. 28–34.
21. Navathe, S., K. Karlapalem, and M. Ra, A mixed fragmentation methodology for initial distributed database design, Journal of Computer and Software Engineering, 1995. 3(4): pp. 395–426.
22. Hauglid, J.O., et al, DYFRAM: dynamic fragmentation and replica management in distributed database systems. Distributed and Parallel Databases, 2010. 28(2–3): p. 157–185.
23. Pérez, J., et al. Vertical fragmentation and allocation in distributed databases with site capacity restrictions using the threshold accepting algorithm. in Mexican International Conference on Artificial Intelligence. 2000. p. 75–81. Springer.
24. Gu, X., W. Lin, and B. Veeravalli, Practically realizable efficient data allocation and replication strategies for distributed databases with buffer constraints. IEEE Transactions on Parallel and Distributed Systems, 2006. 17(9): p. 1001–1013.
25. Song, S.-K. and N. Gorla, A genetic algorithm for vertical fragmentation and access path selection. The Computer Journal, 2000. 43(1): p. 81–93.
26. Ma, H., K.-D. Schewe, and M. Kirchberg. A heuristic approach to vertical fragmentation incorporating query information. in Proceedings of the 17th Australasian Database Conference, 2006. (49): p. 183–192. IEEE.
27. Rosa Karimi Adl, Seyed Mohammad Taghi Rouhani Rankoohi, A new ant colony optimization based algorithm for dataallocation problem in distributed databases, Knowl Inf Syst (2009) 20: 349–373, DOI 10.1007/s10115–008-0182-y.

28. Goli, M. and S.M.T.R. Rankoohi, A new vertical fragmentation algorithm based on ant collective behavior in distributed database systems. Knowledge and Information Systems, 2012. 30(2): p. 435–455.
29. Shamkant B. Navathe and Mingyoung Ra., *Vertical partitioning for database design: A graphical algorithm*, Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 440–450, 1989.
30. Khan, S.U. and I. Ahmad, *Replicating data objects in large distributed database systems: an axiomatic game theoretic mechanism design approach*. Distributed and Parallel Databases, 2010. 28(2–3): p. 187–218.