Current Journal of Applied Science and Technology

27(2): 1-12, 2018; Article no.CJAST.37079
ISSN: 2457-1024
(Past name: British Journal of Applied Science & Technology, Past ISSN: 2231-0843, NLM ID: 101664541)

A Survey on Distributed Databases Fragmentation, Allocation and Replication Algorithms

H. A. Fuaad[1*], A. A. Ibrahim[1], A. Majed[1] and A. Asem[1]

[1]Thamar University, Yemen.

Authors' contributions

This work was carried out in collaboration between all authors. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/CJAST/2018/37079
Editor(s):
(1) Robert Nowak, Professor, Head of Artificial Intelligence Division, Institute of Computer Science, Warsaw University of Technology, Poland.
Reviewers:
(1) Iroju Olaronke, Adeyemi College of Education, Nigeria.
(2) K. Jayakumar, Information Technology, VPMM Engineering College for women, India.
(3) Manish Mahajan, CGC College of Engineering, India.
Complete Peer review History: http://www.sciencedomain.org/review-history/24551

Review Article

Received 29th September 2017
Accepted 22nd March 2018
Published 9th May 2018

ABSTRACT

Due to the huge amount of computer data stored in databases, one centralized database cannot support and provide good performance and availability when contains huge data which used by large number of users. Thus, the distributed database is a good technique to overcome this problem by fragmenting the database and allocating the right database fragmentation in the right site. Many researches present static optimized algorithms of distributed database fragmentation, allocation and replication (Horizontal/ Vertical) at the initial stage of the distributed database design using different or similar techniques, which affect the performance of database system. Therefore, this study aims at reviewing and comparing the best-presented algorithms from the design perspective, with the aim of identifying the strength and weakness points of each algorithm. Furthermore, this study could be considered as the first study that attempts to identify the most critical criteria that were used for comparing the optimized algorithms that have been proposed and used in distributed database fragmentation and allocation.

Keywords: Distributed database; vertical partitioning; horizontal fragmentation; attribute usage matrix; frequency matrix; attribute affinity matrix; Crud matrix.

**Corresponding author: E-mail: fuaad.abdulrazzak@gmail.com;*

1. INTRODUCTION

Regarding the big growth of databases that cause tables to contain the very big size of data, distributed databases promise to solve the most problem related to centralized databases. For instance, centralized database suffers from adequate reliability, autonomy, less accessibility and limited scalability support. These issues motivate the argent emergence of the distributed database as a response to the rapid growth of users' needs.

Distributed databases depend on splitting the relations of the logical schema to smaller fragments among different sites. There are three types of relation fragmentation including horizontal, vertical and hybrid (mixed) fragmentation. In horizontal fragmentation, the relation fragmented based on tuples, this means each fragment holds all attributes of the relation but an instance of tuples. The main intuition behind horizontal fragmentation (HF) is that every site should hold all data or information that used by queries belong to the applications used in that site to enable them to run faster [1]. HF is defined by using the selection operation σ(R) of a relational algebra.

On the other hand, vertical fragmentation (VF) is used to split the relation vertically to two or more fragments this means the attributes are divided between the fragments, each attribute shall belong to one fragment. The main reason for using (VF) is that, distributing the application among different sites where each site is responsible for processing different functions [1]. Consequently, performing VF is considered to be more complex than horizontal fragmentation. In VF, the primary key is duplicated in each fragment to identify the complete record (tuple) in all fragments, but for good performance, we can use a record identifier in case of using many attributes as a primary key. Projection operation (π) is used to represent the vertical fragmentation, for example, the relation employ (EMP) with A1, A2, A3, A4 is divided into two fragments, fragment EMP1 includes A1 and A2, whereas fragment EMP2 includes A3 and A4.

$$\begin{aligned} EMP &= \pi\ A1, A2, A3, A4(R) \\ EMP1 &= \pi\ A1, A2(EMP) \\ EMP2 &= \pi\ A3, A4(EMP) \end{aligned}$$

The last type of fragmentation is called hybrid or mixed fragmentation, which fragments relations horizontally and vertically at the same time, yet it is considered to be the most complex approach among others.

The largest problem during designing the distributed database is the fragmentation and allocation of relations, which in fact be one of the main challenges facing the design of distributed database. Thus, the main goal of distributed database design is to limit queries to access data relevant to their respective transaction [2], meanwhile, satisfying good locality and reducing remote access to other sites during query processing. To fragment the relation horizontally, min-term (predicate) is used to fragment the data with up to 2^n choices are available. Whereas in VF, fragmentation depends on grouping the attributes to different fragments and a lot of choices can be generated, which is used to obtain an approximation result B(m)=m^m, for example, if m=15 we have B(m)=〚10〛^9 choices [2]. It is worthy to point out that, there is a lack of a clear and an optimal solution that can be used for fragmentation [3]; consequently a heuristic approach is used instead. The main issues of distributed database design are how to fragment relation and where to allocate or duplicate these fragments [4]. Many algorithms are designed for fragmenting the database; most of these algorithms depend on analyzing the queries that used within the applications running in all sites. In HF, queries min-term are used to obtain the nature of data used by different sites, while in VF, the queries are used to obtain the affinity between the attributes. Many algorithms collect the information about query executions empirically from the database statistics after implementing the system and this gives good result, whereas other algorithms consider waiting until system's implementation is not a good idea, instead, this information can be provided by the designers at the initial stage of system design during requirements analysis phase. In both scenarios, most of these algorithms use common matrices in their modules like: attribute usage matrix (AUM), frequency matrix (FM), attribute affinity matrix (AAM), and CRUD (Create, Read, Update, Delete) matrix. AUM is used to reflect the usage of attributes within queries. FM is used to obtain frequencies of queries. AAM is used reflect the affinity between the attributes used in the same query. From a different perspective, CRUD is useful to determine different SQL operations used in each site and various weights can be given to calculate the cost of each operation.

It is obvious that fragmentation algorithms could be divided into static and dynamic. The static algorithms fragment the relations manually and can be done once during the design phase. While, on the other hand, dynamic algorithms consider the changes in access pattern of queries and re-fragment the relations according to the new situations.

However, this study aims at comparing between many static algorithms designed originally for (Horizontal/Vertical) fragmentation, allocation and replication of distributed database (DDB). Furthermore, this study collects different criteria, which are used later in this study as a base for comparing the most known and used algorithms. Therefore, these collected criteria would be used by other researchers to compare many different algorithms in the field of DDB.

The reset of this paper is organized as follow: section 2 describes the different algorithms used in fragmenting, allocating and replicating fragments in DDB, section 3 contains the comparison results between these algorithms, and finally, the conclusion will be in section 4.

2. RELATED WORK

A survey study was conducted by [5], which covers the concepts of distributed systems and different related techniques. Meanwhile, the concepts of fragmentation, allocation, and replication were neglected. A systematic survey study conducted by [6], which focused on the dynamic data allocation algorithms.

Fragmenting distributed database based on attributes and queries predicates have been studied earlier by many researchers. The HF partitions the database by using min-term which is proposed by [7]. HF is a technique that allows the database to be partitioned into instances or tupules [8]. VF algorithm is proposed by [9] using AUM and bond energy algorithm (BEA). Many algorithms have been proposed later with aim of optimizing the algorithms, for example, [10] improved the previous work on VF by proposing an algorithm using a graphical technique. From another point of view, [11] used AAM to generate groups based on affinity values [12]. Marwa et al. in their work reported in [13] used the instance request matrix to horizontally fragment object-oriented database. A static VF algorithm that works at the initial stage of database design based on using the number of occurrences of an attribute in a set of queries rather than the frequent of query (FOQ) accessing these attributes [14]. Most researchers like [15, Abdel Raouf et al. [16] and [2] after that adapt the idea proposed in [14] which fragments the relations at the initial stage of database design. The reported work in [17] developed an HF algorithm using allocation locality precedence (ALP) to determine the most important attribute that will be used to fragment the relation. Another algorithm called "An Apriori-Based Vertical Fragmentation was proposed in [15]. The aim of this algorithm is to fragment relations vertically by clustering the more mutually coupled attributes together based on AUM [15]. In contrast, [12] developed a technique for horizontal fragmentation based on CRUD matrix and AUM in addition to communication and space constraints. Another algorithm called valley based vertical partitioning algorithm (VBVPA) which is proposed in [16]. This algorithm adapted clustered affinity matrix (CAM) based on AUM and AAM [16].

Abdel Raouf et al. [16] proposed an algorithm that vertically fragments, allocates and replicates scheme of a distributed database called (VFAR), it allocates and replicates the fragments to the site using manipulation and reading operations [18]. It can be seen from the literature that, many algorithms have been proposed to optimize the process of DDB fragmentation, allocation and replication such as [17]. Therefore, this study selected the most known and used algorithms including algorithms proposed in [10,14,17, 15,12,16,18].

3. THE SURVEYED ALGORITHMS

3.1 Static Vertical Partitioning of a Distributed Database

Abuelyaman [14] proposed a vertical partitioning algorithm for improving the performance of DBS. The algorithm uses the number of occurrences of an attribute in a set of queries rather than the FOQ accessing these attributes. This makes the fragmentation of a DB schema even before its tables are populated. Thus the DB designer will be able to perform partitioning and consequent distribution of fragments before the database enters operation. A simulator for the algorithm has been developed which will be explained bellow [14]. Results of simulations were consistent with those obtained using frequency based partitioning algorithms. The significant advantage of the suggested algorithm is that the database designer doesn't have to wait for empirical data on query frequencies before partitioning a database. The work [14] provided a solution called StatPart for initial fragmentation of relations, fragmentation can be decided even before database tables are populated. Abuelyaman classified his algorithm as a static partitioning because it does not depend on (FOQ) and he used a randomly generated reflexivity matrix, a symmetry matrix and a transitivity module to produce vertical fragments of the relations.

According to Abuelyaman [14], the only way for a partition to be independent of FOQ is when it is based on a database schema. The DB Designer must gain sufficient knowledge on the business requirements, and gets sufficient information about the intended usage of the database to determine the sets which called the Set of Kickoff Queries (SKQ) and the Set of Future Queries (SFQ). According to [14], the proposed simulator has three modules which are reflexivity, symmetry and transitivity.

3.1.1 Reflexivity module

Reflexivity of an attribute X represents the number of queries that reference X. In Reflexivity Matrix (RM), the number of 1's on the column of attribute X represents the degree of reflexivity of X.

As an input, the module prompts a user to enter values for each of the first three parameters (Na, Nk, Nf) which each stands for

Na	: the total number of attributes.
Nk	: the number of queries in the set SKQ.
Nf	: the number of queries in the set SFQ.

The output is the general matrix that relates attributes to queries that will be called the Reflexivity Matrix (RM). In an RM matrix, the total number of 1's on a column gives the degree of reflexivity of the column header's attribute. Abuelyaman assumed that each attribute is included in at least one query. Consequently, each must have a reflexivity degree of at least one.

3.1.2 Symmetry module

Two attributes can be called symmetric when there is at least one query that includes both attributes in SM graph symmetry between any two vertices U and W is represented by an edge connecting U to W. The Degree of Symmetry (DS) between U and W is represented by the label on their edge, and corresponds to the number of queries that include both. Abuelyaman used two equations to compute the Symmetry Matrix (SM) that defines the desired relationships between attributes. The first equation adds up column entries for each attribute j in RM matrix to determine its reflexivity, the diagonal entries on an SM matrix give the reflexivity degrees of attributes. The second Equation finds the intersection between each pair of attributes i and j.

3.1.3 Transitivity module

The transitivity module receives the SM matrix as input and produces the required partition as output. The presented tactic of the algorithm is to look for the most loosely coupled attribute in the partition and move it to a different subset. The new hit ratio is then computed and checked against the threshold. The process is continued till it achieved. The acceptable hit ratio according to the researcher, the hit ratio t occurs when all attribute is in a single set. The only time this is true is when the schema cannot be partitioned. The DB designer is responsible for setting up a partition's hit ratio threshold.

3.2 A New Technique for Database Fragmentation in Distributed Systems

Khan and Hoque (2010) proposed a technique that depends on the use of Attribute Locality Precedence (ALP) which means fragmenting a relation horizontally based on the locality of precedence of its attributes. ALP represents the value of importance of an attribute with respect to sites of distributed database. Database designers are responsible for building an ALP table for each relation of a DDBMS during database design stage. CRUD (Create, Read, Update, and Delete) matrix and cost functions are used in combination with the ALP table. Results showed that for relational databases in distributed systems, this proposed technique can solve initial fragmentation problems properly.

3.2.1 CRUD matrix (data-to-location)

It is used by the Database analysts and designers in the requirement analysis phase of system development. A CRUD matrix is a table (e.g. Table 1) of which rows show the attributes of the entities of a relation and columns show the different locations of the applications calculating precedence.

3.2.2 Attribute locality precedence (ALP)

ALP used to fragment a relation horizontally according to the locality of precedence of its attributes. It can be defined as the value of importance of an attribute with respect to sites of distributed database of an attribute of a relationship we take the MCRUD matrix of the relation. Khan and Hoque proposed a model as depicted in Fig. 1, this model does some processes. First, Construction of MCRUD Matrix, calculation of ALP, construction ALP table, generation of predicate set, finally, fragmentation of the relation.

3.2.3 Constricting of MCRUD matrix

Modified CRUD (MCRUD) is used by Khan and Hoque as shown in Table (1). It is a table constructed by placing predicates of attributes of a relation as the rows and applications of the sites of a DDBMS as the columns. It different on the normal CRUD because it gives a different weight for each operation (Create, Read, Update, Delete). For example, Khan and Hoque Calculate the cost for each predicate as the sum of all applications" cost of predicate j of attribute i at site k. Then find out at which site cost of predicate j is the maximum. To calculate ALP, researchers assumed that frequency of fC, fR, fU and fD=1 because the designer will not know the actual frequencies, also they assumed the weight of operations of as C=2, R=1, U=3 and D=2.

3.2.4 Construction of ALP table

ALP table contains the values of the entire attribute, the attribute that have the highest precedence value which will be treated as the most important attribute for fragmentation. According to the researcher's table, the attribute "Branch" had the highest precedence value, so predicate set was generated for Branch, the attribute with highest locality precedence of the relation. Finally, According to the predicate set P, the relation was fragmented and allocated to the sites.

3.2.5 Calculating of predicate set

After constructing the ALP table the predicate set is chosen based on the predicate which has the maximum cost, so the fragmentation process will depend on that predicate to fragment the relation.

3.2.6 Fragmentation of relation

After determining the predicate which the relationship will be fragmented based on it, fragments will be allocated to the sites based on queries used by each site for that predicate.

3.3 An apriori-based Vertical Fragmentation Technique for Heterogeneous Distributed Database Transactions

Dharavath et al. [15] proposed an algorithm that is capable of taking proper fragmentation decision from the empirical data (i.e., queries) available at the initial stage. It fragments a relation vertically according to the generation of frequent itemsets of its attributes. Frequent item means the attributes or items, which are being called together "very often" as an item sets. Item sets can contain two or more items (attributes) in each set; this depends on the required ratio of jointness that provided to the algorithm.

The main idea of this algorithm is obtaining these attributes that are used together very often. Initially, it uses the AUM from the empirical data at the requirement analysis phase, in addition to the frequency of each query in each site. After that it composes a file called "trasfile.txt" contains the provided queries, but each query will be repeated in the file by the number of its frequencies. This file will be the input of the apriori algorithm which will generate the frequent item sets for a prescribed support.

Table 1. Example of MCRUD matrix (obtained from [17])

		Site & Application			
		Site 1		Site 2	
		APP1	APP2	APP1	APP2
Entity, Attribute. Predicate	Proj Budget<10000 $	CRUD	RU	RU	R
	Proj Budget=10000 $	CRUD	RU	R	R
	Proj Budget>10000 $	CRUD	R		
	Branch Name=A	CRUD	RU	RU	R
	Branch Name=B	CRU	RU	CRUD	RU
	Branch Name=C	CRUD	RU	R	R

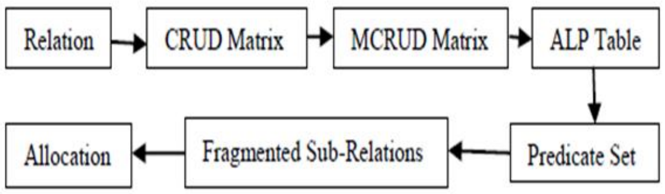

Fig. 1. Block diagram of the system (obtained from [9])

The apriori algorithm will take its inputs from transfile.txt that composed from AUM, and it has minimum support ratio to determine the required satisfaction ratio for the affinity between the attributes, larger min support value results less mutually coupled attribute and vice versa. This algorithm produces item sets each one contains the same number of items (attributes) like 2, 3 or four attributes, the attributes within item sets are jointness, which means same attribute can belong to different item set. The authors tested their algorithm on a relation that has 20 attributes [A0 – A19] and 16 transactions on three sites. The result gave big ratio reached to 60% when the adjacent attributes were two only, but this ratio decreased when more adjacent attributes were required like 3 in that case the ratio decreased to 30%, also when 4 adjacent attributes were required the ration decreased less than 30%.

3.4 A Synchronized Design Technique for Efficient Data Distribution

Hassan I. Abdalla [12] proposed a new optimal fragmentation, allocation, and replication algorithm to perform horizontal fragmentation (HF) in a synchronized fashion using a cost model. This algorithm takes into consideration the communication and storage cost in addition to the other criteria like AUM and CRUD matrices. The intended relation is divided horizontally according to the principle of CRUD matrix for queries used in each site, but the author here divided the operations into two sets one for the read operation and another one for the reset operations (Create, update, delete) in addition to using a frequency of queries. The proposed algorithm in [12] called attribute retrieval and update frequency (ARUM) matrix. From ARUM matrix and by using the provided distance cost matrix between the different sites a new matrix can be obtained which represents the total cost for each attributes in different sites, the matrix called attribute read update frequency (ARUF). Then the relation is divided based on the attribute having the largest cost as we will see in the next section.

3.4.1 The heuristic approach

As it has been mentioned above, this algorithm will fragment, allocate and replicate relations to different sites. First, it fragments the relation based on candidate attribute after calculating the ARUF matrix form ARUM and distance matrix, hence, the fragments will be produced based on the predicates of the candidate attribute. After that this algorithm will allocate the fragments to the sites based on a heuristic model, this model allocates the fragment to the site that performs the highest query cost for the fragment and it has two phases for allocations the fragments, first one it allocates the fragments based on the update operations which means fragments will be allocated to sites sustains the largest cost of update operation, in the second phase if any site constraints can prevent this fragment to be allocated, then it shall be replicated to all other sites required it.

3.4.2 The proposed fragmentation model

Fig. 2 shows the proposed model for Hassan I. Abdalla's technique, which takes the relation as an input and by using the ARUM matrix that provided by the designers at the initial phase of database design it will provide the ARUF matrix, then from this matrix CA (candidate attributes) will be produced.

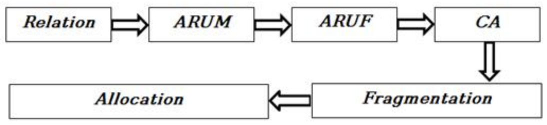

Fig. 2. The proposed system phases (obtained from [12])

Candidate attributes mean the attributes having the highest cost and it will be used to fragment the relation. Fragmentation and allocation depend on CA matrix. The fragmentation model in this algorithm is based on a set of simple predicates Pr [P1, . . . ,Pn] which have been assigned to the relation attributes A[A1, . . . ,An]. Attributes with a numeric value will comprise a predicates Pr that have one of three states: (Pri> V), (Pri< V) or (Pri = V). However, attributes with alphabetical value will comprise a predicates Pr that have only one state: (Pri= V). In addition, each site has a capacity C (C1, C2, . . . ,Cm) and fragment limit FL (FL1, FL2, . . . ,FLm) which indicates how many fragments the site can handle. Also this model uses the communication cost matrix table that represents communication between network sites and used to obtain the distance cost matrix which in its turn is used to calculate the average retrieval cost and the average update cost for candidate attribute predicates individually.

Now to determine the CA we have for each predicate there is a predicate attribute read cost and a predicate attribute update cost, and each query predicate has a frequency value represents the uses of that query in different site. So to compute the cost of using each attribute in the site uses the predicate of that attribute we shall multiply the cost of reading and update by the frequency of each query and a new matrix called Sum of attributes Frequency Table (SAFT) will be produced.

Finally, a distance matrix (DM) which is obtained by minimizing communication cost matrix (Whitten et al., 2004) will be multiplied by SAFT matrix to compose ARUF matrix. Thus, the candidate attribute (CA) can be obtained as the attribute with maximum cost value in this example the salary attribute is considered as a key factor for fragmentation and allocation process. After that three fragments will be generated according to the provided predicates by the designer (>, <, =).

After determining the CA a heuristic model will be used to allocate the fragments to sites, the model will perform the following: First, calculate the Average Retrieval Cost (ARC) and the Average Update Cost (AUC) for candidate attribute predicates individually. Second, for each attribute predicate, it checks if ARC > AUC. If this is the case then replication of the concerned fragments will be performed over the sites that require retrieval, update or both. However, if ARC < AUC, then the concerned fragments will be allocated to the site having the maximum value for update cost. This is the first phase of the algorithm where site constraints are relaxed. In either case, if site constraints are forced and this called second phase, find site with the next maximum update cost value in ARUM table to be the targeted site for the fragment allocation and replication.

3.5 An Optimized Scheme for Vertical Fragmentation, Allocation and Replication of a Distributed Database

Raouf et al. [16] proposed a vertical fragmentation, allocation and replication scheme of a distributed database called (VFAR) that used at initial stage of database design during the requirements analysis phase, this approach uses CRUD matrix and partitions the distribute database relations using the enhanced minimum spanning tree (MST) Prim's algorithm, and it allocates and replicates the fragments to the site that performs high data manipulation for that fragments than other sites to enhance system performance, increase availability and minimize the communication cost. In addition to it also replicates the fragment to the site that performs more read operations for that fragment than other sites. VFAR scheme uses three modules as shown in Fig. 3. These modules are: attribute site usage and similarity matrix, vertical fragmentation and fragment allocation and replication.

3.5.1 Attribute site usage and similarity matrix module

First module uses an enhanced CRUD matrix places the attributes of a relation as the row and application sites as the columns. The output of this matrix is the attribute site usage matrix (ASUM) which reflected the usage of the attributes according to the sites, in this matrix the occurrence of an attribute is represented by one or zero if else. The output of this matrix is the attribute similarity matrix (ASM), It is used to represent the relationships among attributes sites and gives the number of queries referencing both attributes based on ECRUD matrix. After constructing ASM matrix it is transformed into a graph and a vertex with the name of the attribute presents each attribute and the edge represents the similarity between two attributes.

3.5.2 Vertical fragmentation module

This module is used to vertically partition the relations using the enhanced minimum spanning tree Prim's algorithm which is special case of the generic minimum spanning tree algorithm and used by the authors to find the shortest paths in a graph, it starts from an arbitrary root vertex and grows until the tree spans all the vertices in the tree, each vertex should be connected with two branches that have the largest similarity values. If the required fragments are two, then Prim's algorithm will divide the graph to two sub-graphs at the edges with lowest weight in case of two partitions required. In case of three fragments are required the graph should be split at the edge with the minimum value to a new branch, and so on for each new fragment required the graph should be split at the minimum edge value to a new branch.

3.5.3 Fragment allocation and replication

To allocate the fragments to the sites VFAR scheme generates two matrixes from the ECRUD matrix, first one is the attribute manipulate matrix (AMM) which represents the number of manipulation operations (Create, Update, Delete) that access the attribute according to each site, here frequency of queries does not use, just the occurrence of the attribute is represented by one, also each operation given the same weight (one) by the authors. Another matrix is an attribute read matrix (ARM) which uses the read operation only and has the same logic of the (AMM). The authors suggested that allocation of fragments to the sites depends preliminary on the number of manipulation operations then on the number of reading operations, as an example on Table 2 fragment 1 will be replicated to site 5 and fragment 2 will be replicated to site 3.

3.6 Efficient Fragmentation and Allocation in Distributed Databases

M Sumti and K Akhilesh (2015) proposed a new algorithm for Vertical Partitioning in Distributed Database. The proposed algorithm is named as Valley Based Vertical Partitioning Algorithm (VBVPA). The objective of this algorithm is to search the set of frequently accessed attributes by a distinct set of queries. Using the Valley Based Vertical Partitioning Algorithm, user fragments a relation based on Clustered Affinity Matrix (CAM), calculated from Attribute Usage Matrix (AUM) and Frequency Matrix (FM). In this algorithm, the first row of the Clustered Affinity Matrix (CAM) is taken as input to find the clusters of attributes in a relation. Further, we calculate the difference between neighboring attribute values of the first row of Clustered Affinity Matrix (CAM) and the point at which the current differentiated value is less than the previous and the next differentiated value is considered as split point.

3.6.1 Clustered Affinity Matrix (CAM)

Bond Energy Algorithm (BEA) will be used for clustering the attribute which has high Attribute Affinity value, BEA is used to get the position of attributes in CAM. The attribute is placed to a position where the contribution of attribute placement is highest. BEA has been implemented in three steps which are initialization, iteration and row order.

3.6.2 Placement of attributes in CAM

In the initialization step first and second columns of Attribute Affinity Matrix (AAM) are placed in the first and second columns of Clustered Affinity Matrix (CAM) respectively. VBVPA algorithm mainly involves around three steps. The first step is initialization, in this step the variables and arrays are initialized as required by algorithm.

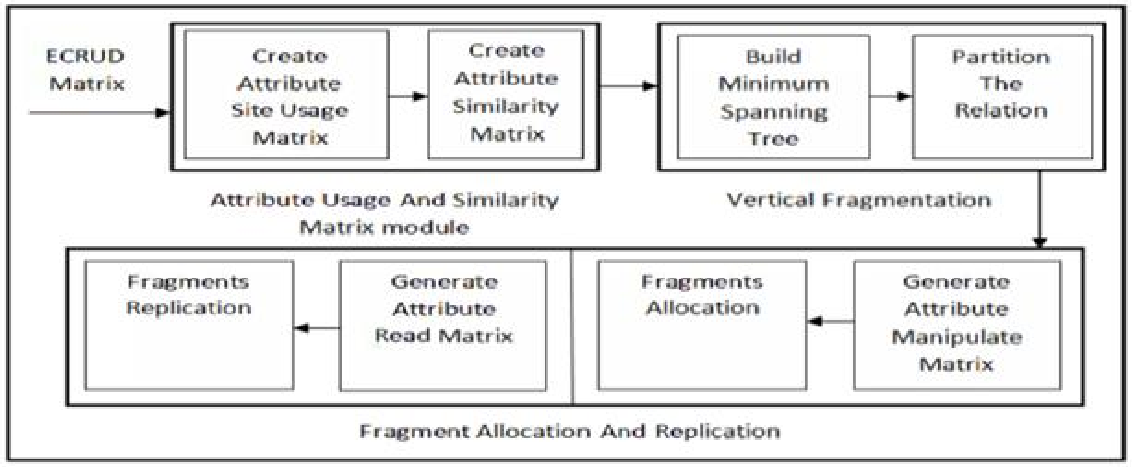
ECRUD Matrix
Create Attribute Site Usage Matrix
Create Attribute Similarity Matrix
Build Minimum Spanning Tree
Partition The Relation
Attribute Usage And Similarity Matrix module
Vertical Fragmentation
Fragments Replication
Generate Attribute Read Matrix
Fragments Allocation
Generate Attribute Manipulate Matrix
Fragment Allocation And Replication

Fig. 3. VFAR module (obtained from [16])

The second step is processing, in this step first row of Clustered Affinity Matrix (CAM) are taken as input for finding the clusters of attributes in a relation. The user keeps the first value of row as it is in Y[i] and then finds the difference of remaining CAM(1,i) values and stores it in Y[i]. The third step is comparison, in this step the user compares each value of array Y[i] with the immediate previous and next values of Y[i], wherever the current value is found less than previous and next values that point is considered as split-point. The following calculation is performed with referenced to CAM.

Fig. 4 shows the valley formed between the values X [2] =2 and X [4] =4 i.e. at point X [3] =3.So the split point is recorded between second and third attribute of CAM. Hence the Clustered Affinity Matrix is divided into two fragments. The first fragment contains the attributes {A1, A3} while the second fragment contains the attributes {A4, A2}.

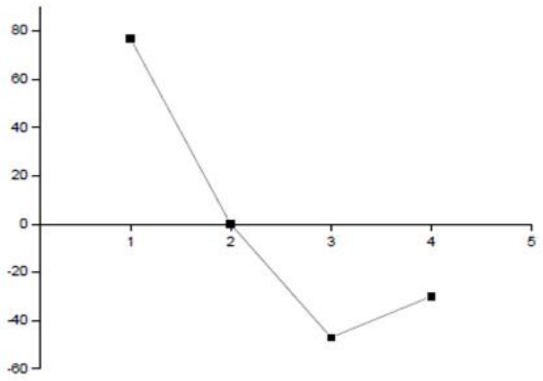

Fig. 4. The graph below shows the Valley value Y[i] at point i (obtained from [18])

4. COMPARISON

In this section we will define the criteria that will be used for comparison between the surveyed algorithms. We designed our comparison based on the criteria that are widely used in the distributed database design and having the most effect on the behavior, performance and results of the used algorithm. We divided these criteria into two different groups. First one (shown in Table 3) contains the inputs matrices and constraints that used by the algorithms to start its work in addition to the clustering techniques used by the algorithms for obtaining its results. While the second group (shown in Table 4) containing the outputs of these algorithms besides its complexity and accuracy. We can describe these criteria as the following:

1 **Type**: represents the approach of fragmentation (Horizontal or Vertical).
2 **CRUD (Create, Read, Update, Delete)**: A matrix used to give different weights to query's operations.
3 **AUM (Attribute Usage Matrix)**: A matrix used to represent the usage of specific attribute in the specific query.
4 **AAM (Attribute Affinity Matrix)**: A matrix used to represent usage of attributes in the same query to determine the affinity between two attributes that are mutually coupled.
5 **FOQ (Frequent of query)**: Represent the frequency of each query used in AUM to reflect the weight of this query regarding the other ones.
6 **CAM (Clustering Affinity Matrix)**: A matrix used to cluster the attributes in disjoint group, each group represents one fragment. These criteria represent the main idea that is provided by the surveyed algorithms.
7 **Communication cost**: The cost of accessing a fragment from a site different than its local site.
8 **Site Constraints**: represent any constraints in any site that allow or prevent allocating the fragments to that site like (space capacity).
9 **Number of Fragments**: represents how many fragments can be produced by the algorithm.
10 **Allocation**: Represents how the algorithm allocates the fragments to the target sites.
11 **Replication**: does the algorithm include replicating fragments to different sites in order to reduce the communication cost.
12 **Difficulty**: does the algorithm uses easy procedures and steps to complete its work or it uses more complicated steps and requirements.
13 **Accuracy**: represents the accuracy of the resulted fragments in addition to allocating and replicating these fragments to the required sites with minimum duplicates and lowest communication cost.
14 **Time Complexity**: represents the number of steps used by the algorithm to complete its work.

Table 2. The number of manipulating and read operation of each site to each fragment (obtained from Raouf et al. [16])

Site	Number of manipulate operation		Number of read operation	
	Fragment 1 (A, B, C, E, G, H, I)	**Fragment 2 (D, F, J)**	**Fragment 1 (A, B, C, E, G, H, I)**	**Fragment 2 (D, F, J)**
Site 1	6	0	3	0
Site 2	10	0	5	0
Site 3	0	6	0	3
Site 4	9	0	2	0
Site 5	17	0	9	0
Site 6	3	0	3	0
Site 7	6	0	5	0
Site 8	2	7	4	2

Table 3. Inputs and technique used by the algorithms being compared

Algorithm	Type	CRUD	AUM	AAM	FOQ	CA	Comm. Cost	Site constraints
(Abuelyaman, 2008) [14]	VF	No	Yes	Yes	No	Graph Theory [1]	No	No
(Khan and Hoque, 2010) [17]	HF	MCRUD	No	No	Yes [2]	Yes/ALP [3]	No	No
(Dharavath et al., 2014) [15]	VF	No	Yes	No	Yes	Priori	No	No
(Abdalla, 2014) [12]	HF	MCRUD	No	No	Yes	No	Yes	Yes
(Raouf et al., 2015) [16]	VF	ECRUD	Yes	Yes	Yes	Spanning Tree	No	No
(Medhavi, 2015) [18]	VF	No	Yes	Yes	Yes	VBVPA	No	No

[1] Graph theory is used first, and then hit ratio is used to improve the clustering of attributes (tactic operation)
[2] In the study the authors [17] used FOQ=1, but it can be given any value during the execution of the algorithm.
[3] ALP used to determine the most significant attribute and if more fragments are required the second attribute will be used

Table 4. Results and technique used by the algorithms being compared

Algorithm	No. of Fragments	Allocation	Replication	Difficulty	Accuracy	Time complexity
(Abuelyaman, 2008) [14]	Two Fragments	No	No	High	High	O(n2)
(Khan and Hoque, 2010) [17]	Many Fragments	Yes	No	Low	High	O(i*j*k*r*)[4]
(Dharavath et al., 2014) [15]	Many Fragments[5]	No	No	High	Medium	O(n3)
(Abdalla, 2014) [12]	Three Fragments or more[6]	Yes	Yes	Medium	High	O(j*h*i*p)[7]
(Raouf et al., 2015) [16]	Many Fragments	Yes	Yes	Low	High	O (E log N)[8]
(Medhavi, 2015) [18]	Many Fragments	No	No	Low	Low	O(n)

[4] i=number of attributes, j=number of predicates, k=number of sites, r=number of applications.
[5] Depending on the hit ratio, smaller hit ratio can produce more fragments.
[6] In case if the candidate attribute has an integer then it will produce three fragments (>, <, =).
[7] j=number of sites, h=number of site queries, i=number of attributes, p=number of predicates.
[8] E=number of edges, N=number of vertices

5. DISCUSSION

To distribute data across many sites that are connected via computer networks, an algorithm that involves making a proper fragmentation and placement decisions is needed. Besides, allocation of the generated fragments should be placed using the appropriate placement algorithm when it is necessary. Query Operation Cost is considered as one of the main factors that make an algorithm more efficient by using CURD model, Clustering, Communication cost and FOQ are also main factors. According to the comparison between the reviewed algorithms for database distribution that is shown in Table 3 and 4, the algorithms [12] covered all main factors in distributing data including fragmentation, allocation and replication, it has good accuracy and low complexity however algorithms ([17] and [16] covered the query operation cost and FOQ besides fragmentation ,and allocation but did not include the cost of communication which is a dominant cost. The three algorithms above have low complexity producers. None of query operation cost and communication cost was used in algorithms [14], [15] and [18], besides not presenting any allocation or replication solution, even though an algorithm [14] has good clustering method to create fragments.

6. CONCLUSION

Many factors shall be considered when implementing fragmentation and allocation in the distributed database environment, Therefore, this study surveyed and compared algorithms that introduce different techniques for distributing fragments over the distributed database. This study aimed also to define the best design for the distribution of fragments, many factors shall be taken in consideration before designing the databases CRUD matrix where each SQL operation shall be given different weights, FOQ factor also has an important effect and shall be considered to get good results, the most important factor is the technique used in clustering the attributes on (VF) or fragmenting the data in (HF). Good algorithms shall consider fragmenting, allocating and replicating fragments to different sites, to reduce the communication cost.

Most algorithms use the provided query at the initial stage of database design, but we think it is better to use dynamic algorithms on the empirical data after implementing the database to refragment the relations and getting new fragments related to the new situation.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Suganya A, Kalaiselvi R. Efficient fragmentation and allocation in distributed databases International Journal of Engineering Research and Technology. 2013;2.
2. Özsu MT, Patrick Valduriez. Principles of distributed database systems. 3rd ed.: Springer Science & Business Media; 2011.
3. Huang YF, Chen JH. Fragment allocation in distributed database design. Journal of International Science and Engineering. 2001;17 491-506.
4. Singh A, Kahlon KS, Virk RS. Nonreplicated static data allocation in distributed databases using biogeography-based optimization. Chinese Journal of Engineering, 2014.
5. Joshi H, Bamnote GR. Distributed database: A survey. International Journal of Computer Science and Applications. 2013;6(2):289-292.
6. Kumar R, Gupat N. Dynamic data allocation in distributed database systems: a systematic survey. International Review on Computers and Software. 2013;8(2): 660-667.
7. Ceri S, Negri M, Pelagatti G. Horizontal data partitioning in database design. Proceedings of the 1982 ACM SIGMOD international conference on Management of data. ACM. 1982;128-136.
8. Bhuyar PR, Gawande AD, Deshmukh AB. Horizontal fragmentation technique in distributed database. International Journal of Scientific and Research Publications. 2012;2(5):1-7.
9. Navathe S, Ceri S, Wiederhold G, Dou J. Vertical partitioning algorithms for database design. ACM Transactions on Database Systems (TODS). 1984;9:680-710.
10. Navathe SB, Ra M. Vertical partitioning for database design: A graphical algorithm. ACM Sigmod Record, ACM. 1989;440-450.
11. AlFaress MY, Abdalla HI, Marir F, Najjar Y. Vertical partitioning for database design: A grouping algorithm. SEDE. 2007;218-223.
12. Abdalla HI. A synchronized design technique for efficient data distribution. Computers in Human Behavior. 2014; 30:427-435.
13. Marwa F, Ali I, Hesham A. A heuristic approach for horizontal fragmentation and

allocation in DOODB Proc. INFOS2008. 2008.9-16.
14. Abuelyaman ES. An optimized scheme for vertical partitioning of a distributed database. IJCSNS International Journal of Computer Science and Network Security. 2008;8:310-316.
15. Dharavath R, Kumar V, Kumar C, Kumar A. An apriori-based vertical fragmentation technique for heterogeneous distributed database transactions. Intelligent Computing, Networking, and Informatics, Springer. 2014;687-695.
16. Raouf AEA, Badr NL, Tolba M. An optimized scheme for vertical fragmentation, allocation and replication of a distributed database. Intelligent Computing and Information Systems (ICICIS), 2015 IEEE Seventh International Conference on, IEEE. 2015;506-513.
17. Khan SI, Hoque A. A new technique for database fragmentation in distributed systems. International Journal of Computer Applications. 2010;5:20-24.
18. Medhavi SK, Akhilesh. Valley based vertical fragmentation in distributed database. e-Journal of Science & Technology. 2015;10:31-40.