WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

A ROBUST MODEL FOR DEFRAGMENTING MISMATCH DOCUMENT IN A DISTRIBUTED DATA SYSTEM

EZEUGBOR I. C[1], EJIOFOR V. E[1], UMEH I. I[1], OKOLO C. C[2]

[1] Department of Computer Science, Nnamdi Azikiwe University, Awka
[2] Electronic Development Institute, NASENI, Presidency

ABSTRACT

Distributed data system is frequently used lately by several companies and establishments in recent time for business transactions in different localities. Pertinent issues relating to the performance, communication cost, complexity, and maintenance of data in distributed data repository for query processing, regarding to the demand of end users from different locations have been identified. Data is being allocated as replicated, distributed and fragmented over the Internet or Intranet within and across an organization with the help of distributed database. This paper gears towards, developing a robust model for defragmenting mismatch document in a distributed data system. Algorithm clustering was employed in defragmenting mismatch document to individual sites and collating these data to one central store. Object oriented analysis and design methodology (OOADM) was employed in the systematic study and design of the system. Subsequently, the implementation of the data system was done using MongoDB database engine. The result shows a system with external fragmented database with capability of saving to a central database using clustering technique. This system can be used for every individual, organizations and establishments that has need to defragment mismatch document in a data system for a maximum reduction of process time, for easy access of data and for complexity reduction.

Keywords: *Robust Model, Clustering Technique, Data System, Defragment Mismatch, Distributed Database, Computer Network*

1. INTRODUCTION

Distributed data system is being said to be a collection of sites connected by a communication network whereby each site is a data system of its own right, as such the sites have affirm to work together so that a user's own site can view and handle data anywhere in the network as if the data were all saved to reside at the user's own site. When all storage devices are not attached to a common processor, the database is seen as a distributed one. It may be accommodated in multiple computers placed in the same physical location or may be dispersed over a network of interconnected computers. Collection of multiple, a logically interrelated database spread over a computer network is termed a distributed database. Distributed database is also seen as bringing together of data that logically belong to the same system but is extended over the sites of a computer network. Therefore, a distributed data system can be seen as dispersed systems that communicate to each other which are being connected to a computer network. Distributed database helps to allocate data as fragmented, replicated and distributed over the intranet or internet within organization and across the organization. Distributed databases have been developed to meet the information needs of business organization engaged in distributed operations. Such organizations typically have facilities (sites) that have one or more computer systems (nodes) connected via some communication networks (links).

Distributed database system involves the following: how a global relation should be partitioned, how many fragmented copies should be replicated; how segments should be designated to the sites of the communication network, what the required information for fragmentation and allocation is. These issues pose a complication to distributed database design. It is still an unmanageable problem even if each issue is considered individually. To break down the overall problems, defragmenting mismatch document issue only is addressed, supposing that all global relations have already been fragmented.

It can be observed that most banks have their own database system but still need to be collated and run a fragmented central system for all banks irrespective of banking type or name. A robust model was developed for defragmenting mismatch document in distributed data systems on one database using clustering technique.

2. REVIEW OF RELATED WORKS

Due to the complexity of checking completeness of the set of simple predicates used for horizontal fragmentation, an author adopted an affinity-based vertical fragmentation approach to horizontal fragmentation. This approach takes predicate usage and predicate affinity matrix as input and employs the bond energy algorithm to cluster predicates. However, the fragments in the resulting fragmentation schema may overlap each other and therefore cannot satisfy the correctness criteria of fragmentation. Another author presented a graph-based algorithm for horizontal fragmentation, with which predicates are clustered based on the predicate affinities. To remove overlapping, an adjust function is presented to merge two overlapped fragments if merging can reduce transaction costs using cost functions. However, the cost function does not show how costs are computed. Using predicate matrix as input, a number of authors including Cheng, J.M et al proposed a genetic algorithm-based clustering approach, which treats

horizontal fragmentation as a traveling salesman problem (TSP). Horizontal fragmentation is achieved by performing selection operation using the set of the grouped predicates, which are grouped according to the distances. The distance of each pair of attributes actually measure the access frequencies of transactions that do not access the pair attributes together. Additional analysis is needed to simplify the clusters of predicates. Obviously, none of the affinity-based horizontal fragmentation approaches takes into consideration of data locality while clustering predicates. These among other works were reviewed.

3. METHODOLOGY

The Bootstrap was the tool used for the site design to make it faster and easier. This includes HTML and Cascading Style Sheet (CSS) based design templates for typography, forms, buttons, tables, navigations, modals and also gave support for JavaScript plugins. The CSS also was adopted to a play a role of describing the presentation and design of web pages which include colors, fonts and layouts. It was solely introduced and applied into this work to enable the distinction between presentation and content, as well as adding colors, layouts and fonts. JavaScript is basically used as a client side scripting language, and the reason for using it is because of its dynamic nature as well as adding special effects on pages like rollover, and putting up many types of graphics. Content can be loaded into a document whenever the user requires it without trying to tamper with reloading their entire page. In order to achieve faster and scalable network applications, Nodejs is adopted and built on chrome's JavaScript runtime. Jquery is a light weight, "write less, do more", JavaScript library; and it was equally used to make JavaScript much easier to be used at the sites. MongoDB database engine was used to implement a data store that provides high performance, high availability and automatic scaling. Npm, a node package manager was used to install packages locally into this project, specifically into the Node_modules folder. This technique aim at overcoming the problems associated with previous techniques, such as restrictions on the number of network sites, complexity, inefficient solutions etc. The network sites were clustered (using Algorithm_Clustering**)** that is grouping sites which have comparable and similar functions together. The high speed clustering technique based on the least average communication cost between sites will be introduced. This is used for distributed databases where the communication costs between two sites are equal or very close, and similar computers on the network are used. The Clustering Decision Value (cdv) determines whether or not a site can be grouped in a specific cluster. The Communication Cost Range CCR is used when the site should match to be grouped in a specific cluster. This value is determined by the network administrators and depends on how much time allowed for the sites to transmit or receive data to be considered in the same cluster. The system is out to improve data accessibility, security, speed of operation and data integrity of a distributed database system.

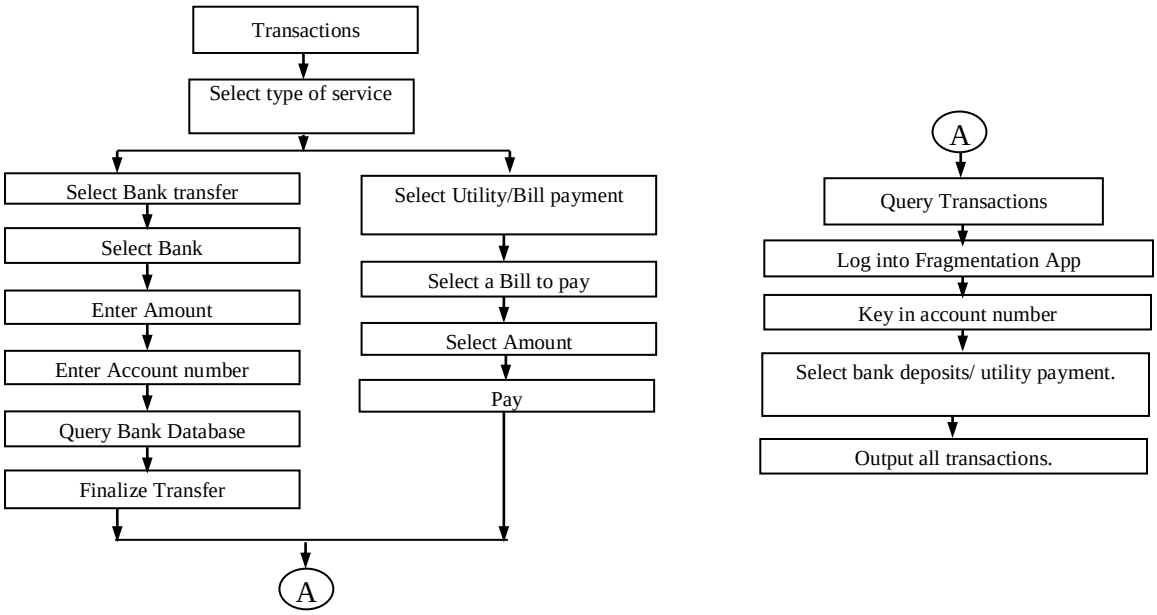

Fig 1: Information Flow Diagram

Design Specification and Algorithms

To solve the problem of taking proper fragmentation decision at the initial stage of a distributed database, clustering algorithm (Algorithm_Clustering technique was adopted as depicted in Figure 3.

Algorithm_Clustering

CR: Clustering Range
NS: Number of sites in the distributed database system network
Output: CSM: Clusters Set Matrix
Step 1: Set 1 to i
Step 2: Do steps (3 - 12) until i > NS
Step 3: Set 1 to j
Set 0 to k: Set 0 to Sum
Set 0 to Average:Set 0 to clusters matrix CM
Step 4: Do steps (5 - 10) until j > NS
Step 5: If $i \neq j$ AND $CC(S_i, S_j) <= CR$,

go to step (6)
Else,
go to step (7)
Step 6: Set 1 to the CM(S_i, S_j) and CM(S_j, S_i) in the clusters matrix
Add CC(S_i, S_j) to sum Add 1 to k
Go to step 8
Step 7: Set 0 to the CM(S_i, S_j) and CM(S_j, S_i) in the clusters matrix
Step 8: End IF
Step 9: Add 1 to j
Step 10: Loop
Step 11: Average = Sum/k Average(i) = Average Add 1 to i
Step12: Loop
Step 13: Set 1 to m
Step 14: Do steps (15 - 36) until m > NS
Step 15: Set 1 to q Set 0 to Minaverage Set 0 to Minrow
Step 16: Do steps (17 - 20) until q > NS or Minaverage >0
Step 17: If Average(q) > 0 Then
Minaverage = Average(q)
Else
Go to Step 18
Step 18: End If
Step 19: Add 1 to q
Step 20: Loop
Step 21: If Minaverage = 0 Then
Set site number to a new cluster
Else
Go to Step 22
Step 22: End If
Step 23: Set 1 to p
Step 24: Do steps (25 - 28) until p > NS
Step 25: If Average(p) > 0 AND Average(p) < Minaverage Then
Minaverage = Average(p)
Minrow = p
Step 26: End IF
Step 27: Add 1 to p
Step 28: Loop
Step 29: Set 1 to a
Step 30: Do steps (31 - 34) until a > NS
Step 31: If CM(S_{minrow}, S_a) = 1 Then
Set 1 to CSM(S_{minrow}, S_a)
CM(S_{minrow}, S_a) = 0 Step 32:
End IF
Step 33: Add 1 to a
Step 34: Loop
Step 35: Add 1 to m
Step 36: Loop
Step37:Stop
Input: Tmax: number of transactions issued in the database
Fmax: number of the disjoint fragments used for allocation
Cmax: number of clusters in the distributed database system

Algorithm_Fragment

Input: K: Number of the last fragment
Rmax: Number of database relations
Nmax: Number of fragments in each relation

Step 1: Set 0 to K
Step 2: Set 1 to R
Step 3: Do steps (4 - 21) until R > Rmax
Step 4: Set 1 to I
Step 5: Do steps (6 - 20) until I > Nmax
Step 6: Set 1 to J
Step 7: Do steps (8-18) until J > Nmax
Step 8:If I ≠ J and ∃ Si ,Sj ∈ SR
goto step (9)

Else Add 1 to J,
go to step (18)
Step 9: If Si ∩ Sj ≠ Ø
do steps (10)-(17)
Else
Add 1 to J and go to step (19)
Step 10: Add 1 to K
Step 11: Create new fragment Fk = Si ∩ Sj and add it to F
Step 12: Create new fragment Fk+1 = Si - Fk and add it to F
Step 13: Create new fragment Fk+2 = Sj - Fk and add it to F
Step 14: Delete Si
Step 15: Delete Sj
Step 16: Set Nmax + 1 to J
Step 17: End IF
Step 18: End IF
Step 19: Loop
Step 20: Add 1 to I Step 21: Loop
Step 22: Set 1 to I
Step 23 Do steps (24 - 35) until I > Nmax
Step 24: Set 1 to J
Step 25: Do steps (26 - 33) until J > Nmax
Step26:If I ≠ J and ∃ Si,Sj Є SR
goto step(27)
Else Add 1 to J,
go to step (33)
Step 27: If Si ∩ Sj = Ø do steps (28)-(33)
Step 28: Add 1 to K
Step 29: Create new fragment Fk = Rj - UF
Step 30: End IF
Step31:If Fk ≠ Ø Add Fk to the set of F
Step 32: End IF
Step 33: Loop
Step 34: Add 1 to I Step 35: Loop
Step 36: Set 1 to I
Step 37: Do steps (38 - 53) until I > F
Step 38: Set 1 to J
Step 39: Do steps (40 - 51) until J > F
Step 40:If I ≠ J and ∃ Fi,Fj Є FR goto step (41) Else, Add 1 to J and go to step (50)
Step 41: If Fi ∩ Fj ≠ Ø do steps (42)-(49) Else, Add 1 to J and go to step (49)
Step 42: Add 1 to K
Step 43: Create new fragment Fk = Fi ∩ Fj and add it to F
Step 44: Create new fragment Fk+1 = Fi - Fk and add it to F
Step 45: Create new fragment Fk+2 = Fj - Fk and add it to F
Step 46: Delete Fi
Step 47: Delete Fj
Step 48: Set F + 1 to J
Step 49: End IF
Step 50: End IF
Step 51: Loop
Step 52: Add 1 to I
Step 53: Loop
Step 54: Add 1 to R
Step 55: Loop

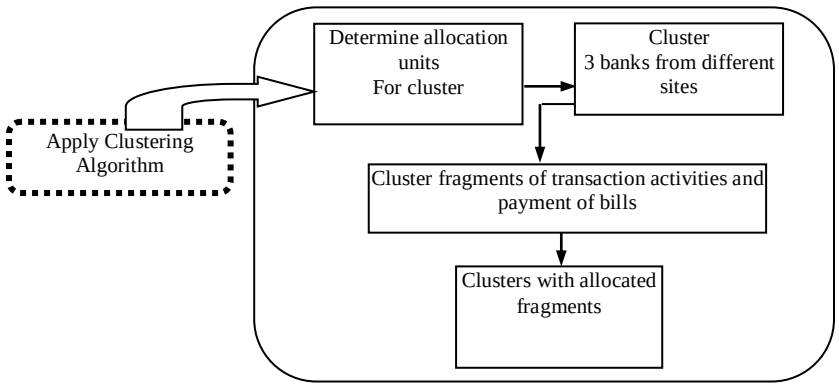

Fig 2: Cluster Fragment Techniques

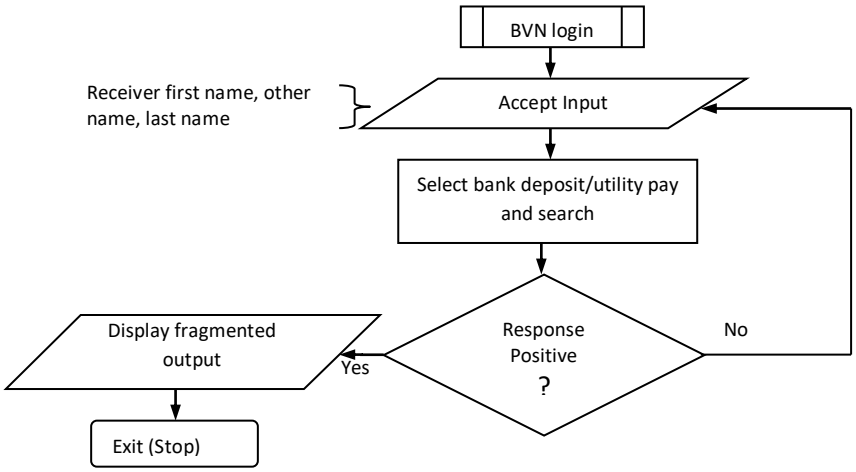

Fig 3: Flowchart on fragmented system

4. RESULTS AND DISCUSSION

The system is designed to display all the transactions made by an individual within a specified period of time by bringing the transactions closer for easy access (Figure 8). It also shows the beneficiary bank, transaction mode, date and time, amount, sender and receiver's account number on the fragmented database. The benefits of this model to the users are reduction in communication cost by reducing the number of servers, easy and fast retrieval of data and closeness of data to the user. This work was achieved by clustering three banks, allotting activities to the banks i.e., transaction of funds and payment of utility bills. The result of this work is as shown in Figure 4 to figure 8

Admin Menu

This consists of Open Account, Search Account and Log Out (Figure 4). Open Account equips the admin with an interface that authorizes the admin open an account for a new customer getting the customers information from the keyboard (Figure 5) and saving to the database for subsequent use, hence a unique account number is generated automatically by the system. Search Account is necessarily needed to navigate from one account to another considering the fact that in an average bank, millions of customers accounts can all be linked to the same database, searching with name or account number allows for ease of transaction after the customers detail and statement is retrieved from the database. There are hosts of functions provided on this level; update of an existing customers account, view and delete accounts in the event "the customer chooses to close their account". Logout, the admin logs out after the admin is through and perhaps satisfied with whatever job done for the day.- Every admin has a designated login ID, with a login ID the system keeps record of user's input and how often a user was active. As such, users will possibly be held countable for any theft or wrong input entered into the system.

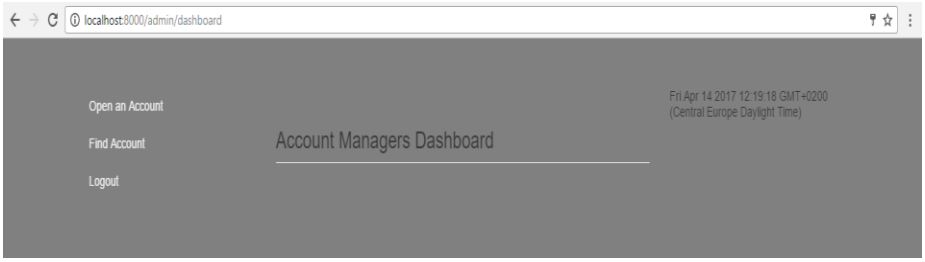

Fig 4: Admin Main Menu

Fig 5: Customer Registration Interface

User Menu

Account Balance, Account Activity, Account Statement, Utility/Bills Payment and Bank Transfer make up the user menu (Figure 6). Account Balance provides a user with the platform that accommodates details of all accounts connected by BVN. The user available balance in all linked accounts will be viewed with no access to printing. Account Activity avails a user see the last five activities or transactions. Account Statement shows all transaction on the account within a stated period of time, the user of the account often decides the range in period. In this menu, the user is availed an option of printing this statement or downloading and saving on their personal device for viewing even without network access at their leisure. Utility/Bills Payment (Figure 7) - A user can equally make other transactions besides viewing statement of account, checking of account balance and so on. This platform gives a user the opportunity to pay for light bill, water bill and data purchase or recharge cards with ease from any of the bank accounts selected and connected with the help of BVN fragmentation. Bank transfer avails a user the chance to transfer funds from any of the user's registered bank accounts to other accounts either to business associates or family.

Fig 6: User Main Menu

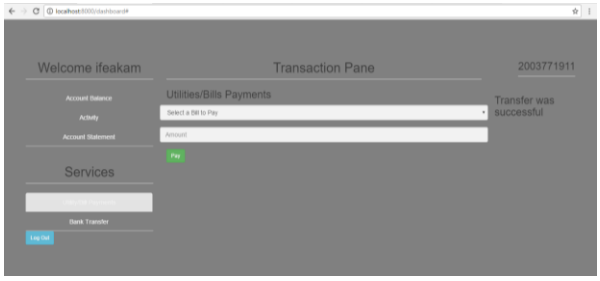

Fig 7: Utility/Bills Payment Interface

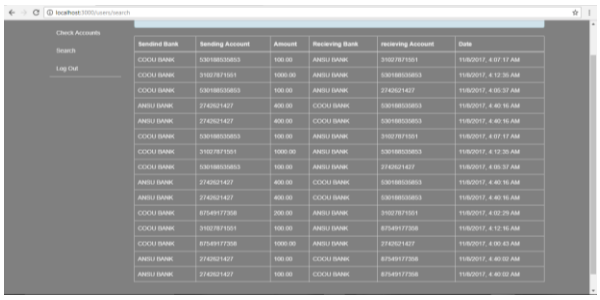

Fig 8: Output Format of the Fragmented System

5. SUMMARY

A robust model for defragmenting mismatch document in a distributed data system was developed and implemented. The new system was developed to rearrange the mismatch document and fragment it in order to create a central financial technology solution that allows users carry out financial transactions from all their bank accounts using database fragmentation to separate the banks and their operations. Problem of delay in processing data at the banks was solved. It equally provided a unified system for all the banks for easy access. The model was created to add fragments to DDS for easy access.

6. CONCLUSION

A modern approach to improve defragment mismatch document in distributed data system was proposed in this work. In addition to our approach is to satisfy a certain level of data availability and consistency. The result gotten in this work indicates that our approach significantly improved and enhanced performance requirement satisfaction in distributed systems. This system has the strength of reducing data transfer between sites, increased security, event log in is also easier as it improves availability, it can control transactions as such lead to safer transactions, efficiency in performing transactions, it is not greatly influenced by errors in assumptions about the distribution of sample errors, data that are not required by local applications are not stored locally and it is easy to implement in any organization. Keeping record of all transactions made from this window for accountability. The system must be informative, robust, responsive, user-friendly and secure. System should be designed to allow possible future expansion.

REFERENCES

[1] **Burrows, (2016)**"The chubby lock service for loosely-coupled distributed systems". http//labs.google.com/papers/chubby.html. retrieved 2010-04-18.

[2] **Chang, S.K. and Liu, A.C.(1982)**"File allocation in a distributed database". Int. J. computInf. Sci, 11(5):325-340. 121,123.

[3] **Cheng, J.M et al. (2005)**"IBM database 2 performance: Design, Implementation and tuning".IBM systems J., 23(2): 189-210.503.

[4] **Chen, J. (2012).** Distributed Database: Fragmentation and Allocation, Journal of Data Miningand Knowledge Discovery ISSN: (3)2229-6662 & ISSN 2229-6670, 2012.

[5] **Ma, H. (2017)** "Prefix-based labeling annotation for efficient XML fragmentation. International Journal of Computer Science & Information Technology (IJCSIT) (7), 2

[6] **Ma, H. (2007)** "An overview of fragmentation design for distributed XML databases". Fifth international conference on computer science and information technology, DOI: 10.5121

[7] **Ozsu, M.T. and Valduriez, P. (1999)**" Principles of distributed database systems": Third edition, DOI 10.1007/978-1-4419-8834-8, c springer science + Buisness media.

[8] **Ra, M. (1993)**"Horizontal partitioning for distributed database design", In advanced in Database research , world scientific publishing, 101-120,1993.

[9] **Zhang, N. (2006)**"Query processing and optimization in native XML databases". PhD thesis,University of waterloo. 719.