e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
Volume:03/Issue:06/June-2021 Impact Factor- 5.354 www.irjmets.com

A FUNCTIONAL MODEL FOR ALLOCATING ALREADY EXISTING DATA IN A DISTRIBUTED DATABASE SYSTEM

Ezeugbor I. C[*1], Anigbogu G. N[*2], Ngene C.C[*3], Okolo C. C[*4]
[*1,2,3]Department of Computer Science, Nnamdi Azikiwe University, Awka.
[*4]Electronic Development Institute, Fed Min of Sci and Tech, Awka Capital Territory.

ABSTRACT

A functional model for allocating already existing data in a distributed database system was developed in this paper. In line with the technological advancement trending today, various establishments are using distributed database system for their constant business transactions in different regions. Some impediments have erupted and observed relating to the analysis, workability, development enhancement, communication cost and complexity of data in distributed data store house for query processing; regarding the demand of end users in different regions. Distributed database helps to keep track of data distribution, fragmentation, replication, as well as having the ability to access remote sites and to transmit queries and data among the various sites in an organization via a communication network. Algorithm_genetic was employed in allocating database fragments to individual sites and collating these distributed databases to one central store. Create, read, update and delete (CRUD) techniques, cascading style sheet (CSS), bootstrap and javascript were employed in the implementation of the system. The methodology used for the design of the system is object oriented analysis and design methodology (OOADM). The implementation of the database of the system was done with mongodb database engine. An external fragmented database system with capability of saving to a central database was achieved. This system aids in maximum reduction of process time, easy access of data and complexity reduction; hence can be used by every individual, organizations and establishments that have possible necessity of fragmenting database in a distributed database system.

Keywords: Functional Model, Fragment, Allocation, Database System, CRUD Techniques.

I. INTRODUCTION

A distributed database 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). To simplify the overall problems, allocation of existing data issue only, is addressed, assuming that all global relations have already been fragmented. The sites of the distributed database can have the same network address and may be in the same room but the communication between them is done over a network instead of shared memory. This implies that there are physical and logical topology considerations in the shared network. The communication network is the only shared resource for distributed database management system. As communication technology, hardware, software and network protocols advance rapidly and prices of network equipment falls every day, developing distributed database systems become more and more feasible. It is well known that every organization has its own designated database but there appears an importance to develop a functional model for allocating already existing data in a distributed database system in order to suit the technological advancement of the globe. In most financial institutions today, a lot of banks have their own database system, still they need to be collated and run a fragmented central system for all banks not minding the banking type or name. In a distributed database, the clumsiness of use of data, the difficulty in accessing and manipulating data, reduction of performance speeds in accessing and the difficulty in data retrieval makes it important to bring in fragmentation. A functional model was developed for allocating already existing data that are in a distributed database systems to one database using two techniques which are and Genetic technique and CRUD technique. Genetic technique comes into play as to identify the activities which are already in existence. In order words, there are so many activities performed by banks in Nigeria but this work is limited to only transactions (credit and debit of account), and also payment of utility bills. CRUD technique comes into play as to determine how long the fragmented data will be displayed before wiping away, which can still be called up when needed, for security purpose. 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, closeness of data to the user and maintaining data integrity.

Review of Related works

Some authors proposed using predicate affinities to perform horizontal fragmentation. However, there is no detailed method on how to perform fragmentation. The resulting set of the clusters of predicates are of the same length in terms of number of simple predicates and are defined on the same set of attributes or methods. An author opined that the effect of horizontal fragmentation should be measured by evaluating the performance of the applications in a distributed database system. Cost-driven algorithm was presented to find a scheme that led to the lowest total query cost based on the cost model. However, in the cost model CPU costs and network communication costs were disregarded because only centralized databases were considered. Therefore, it cannot be applied to distributed databases, where network communication cost predominant total costs. Moreover, complexity of the cost-driven horizontal partitioning algorithm is too high and makes the algorithm computationally intractable. Bellatreche et al. (2007), have studied horizontal class partitioning with input as queries which contain either simple and component predicates, the primary algorithm (PA) is based on a graph theoretic algorithm which clusters a set of predicates into a set of HCFs. The complexity of this algorithm is $O[l * n^2 + \alpha(r + u)]$, where n, l, α, r, u represent the number of predicates, the number of queries, the number of HCFs, the number of attributes used by the queries, and the number of methods used by queries, respectively.

II. METHODOLOGY

The inclusion of HTML and cascading style sheet (CSS) based design templates for typography, forms, buttons, navigations, tables, modals gave rise to a functional model of the system. Bootstrap aided as a structure in designing sites faster and easier; giving support for javascript plugins. The inclusion of CSS to this work is to enable the distinction between presentation and content, that includes colors, layouts and fonts, for the purpose of a better description of the presentation and design of web pages. Javascript, because of its dynamic nature, is commonly used as a client side scripting language; and it adds special effects on pages like rollover and some other types of graphics. It is a standalone language developed in Netscape, as such can load content into a document whenever the user requires it without reloading their entire page. Nodejs is a type of javascript that facilitates the user to run the program in runtime in an environment on external system. Nodejs is also built on chrome's javascript runtime for the purpose of achieving fast and scalable network applications. Jquery is a lightweight, "write less, do more", javascript library; jquery was used in this work for the purpose of making it much easier for javascript to be used at the sites. Mongodb database engine was introduced to this work and was used to implement a data store which provides high performance, high availability and automatic scaling. Mongodb is used to store high volume data and it is also known as a document-oriented NoSQL database. It is used for collections and documents. Npm, is a node package manager used to install packages locally into this project. It was equally used for managing dependencies of various sites. Genetic algorithm (Algorithm_Genetic**)** combined with CRUD technique in allocating fragments to individual sites were employed. Genetic algorithm is considered more attractive when efficiency and solution quality is more important. The aim of combining these techniques is to overcome the problems associated with previous techniques, such as cutting down on the number of network sites, complexity, inefficient solutions etc. The system used one algorithm and one technique to achieve the result in this work. Firstly, fragments are allocated (using Algorithm_Genetic) to the clusters that have transactions using the fragment. Secondly, at each cluster, genetic algorithm was also used. The genetic algorithm searches the solution space for possible allocations, evaluating them using a developed allocation cost function and eventually finds the best allocation in which in most cases is the optimal allocation. The use of genetic technique as a model supports and promotes efficiency of the system in the sense that the data are properly arranged on a row, also displaying the transactions made according in the order of transfer, basically in ascending order. The allocation cost function confirms the status of the allocation for the comparison between the remote access cost of the fragment to the cluster and the cost of allocating the fragment to the cluster. CRUD technique comes into play as to determine how long the fragmented data will be displayed before wiping away, which can still be called up when needed, for security purpose. 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, closeness of data to the user and maintaining data integrity.

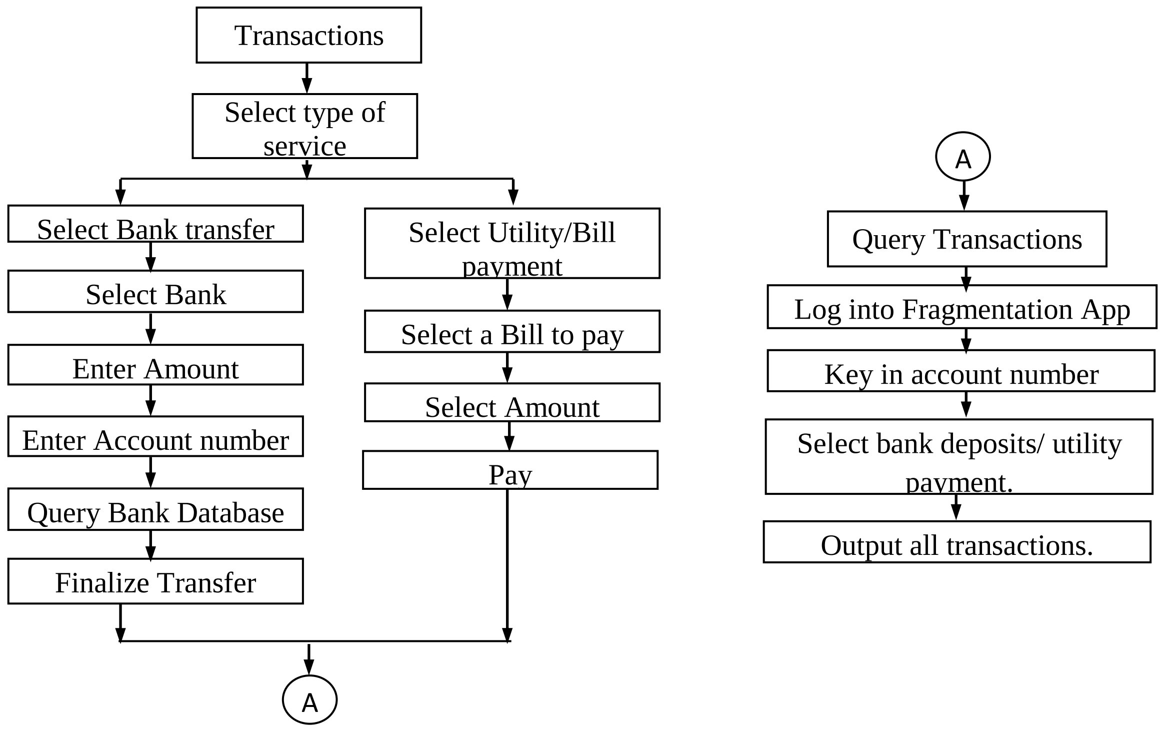


Fig 1: Information Flow Diagram

Design Specification and Algorithms

To solve the problem of taking proper fragmentation decision at the initial stage of a distributed database, genetic algorithm (Algorithm_Genetic and Figure 2) and create, read, update and delete (CRUD) technique were adopted as depicted in Figure 3.

Algorithm_Genetic

```
1. Generate an initial population, repeating random strings of fixed size.
2. Do the selection, reproduction, crossover and mutation operations of the all population.
3. Replace the old population with the new one.
4. Repeat Steps (2, 3) until number of iterations is finished.
5. Display the best answer found (which has the best-fitness).
```

Algorithm_Fragment Allocation

```
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 Є 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
```

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
Volume:03/Issue:06/June-2021 Impact Factor- 5.354 www.irjmets.com

```
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
```

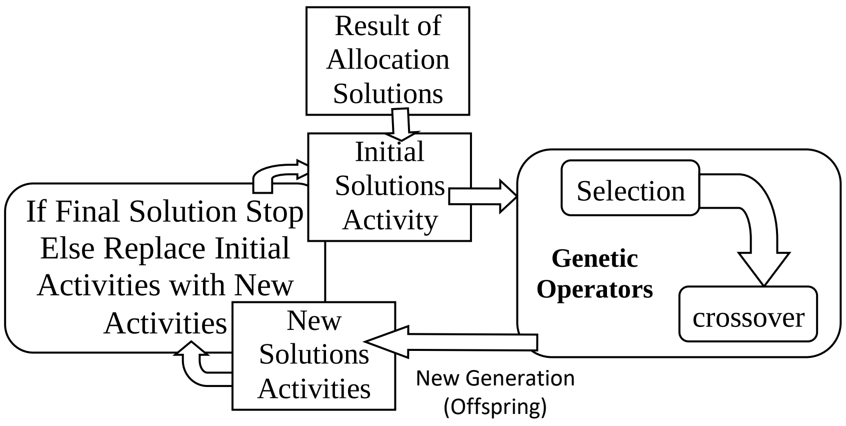

Result of Allocation Solutions
Initial Solutions Activity
Selection
Genetic Operators
crossover
If Final Solution Stop Else Replace Initial Activities with New Activities
New Solutions Activities
New Generation (Offspring)

Fig 2: The Basic Genetic Steps

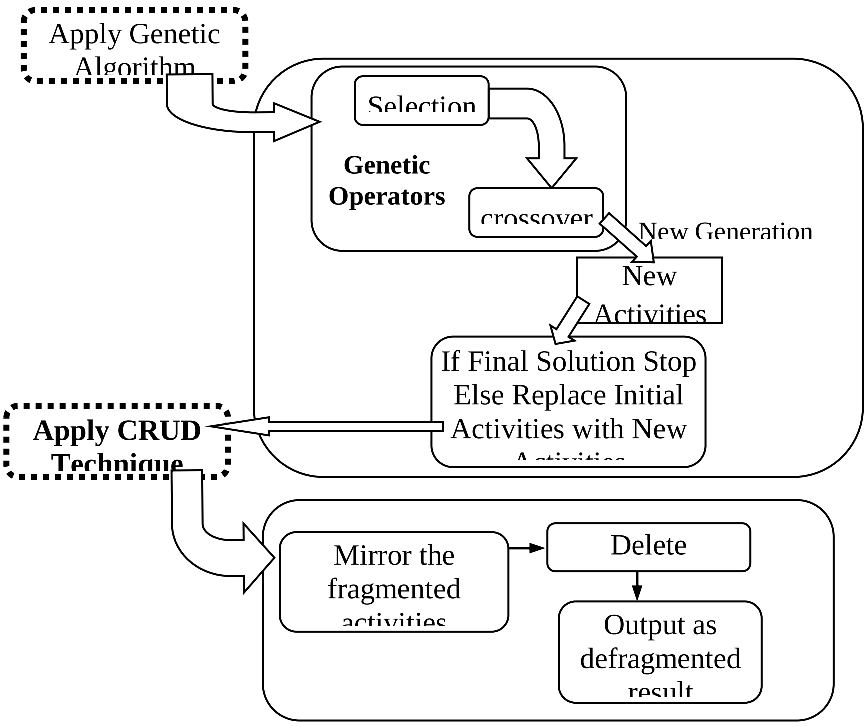

Apply Genetic Algorithm
Selection
Genetic Operators
crossover
New Generation
New Activities
If Final Solution Stop Else Replace Initial Activities with New Activities
Apply CRUD Technique
Mirror the fragmented activities
Delete
Output as defragmented result

Fig 3: Fragment Allocation Techniques

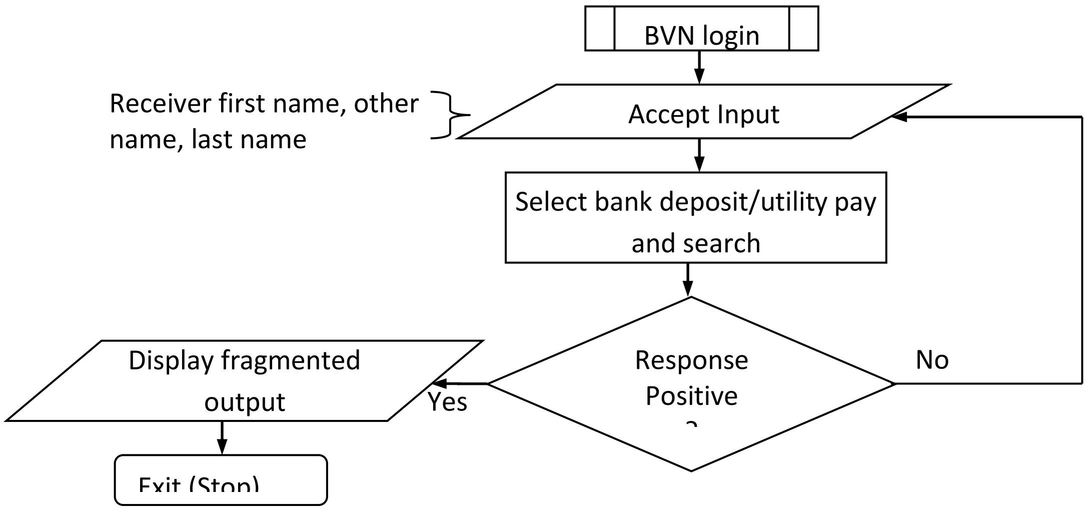

BVN login
Accept Input
Receiver first name, other name, last name
Select bank deposit/utility pay and search
Response Positive ?
No
Yes
Display fragmented output
Exit (Stop)

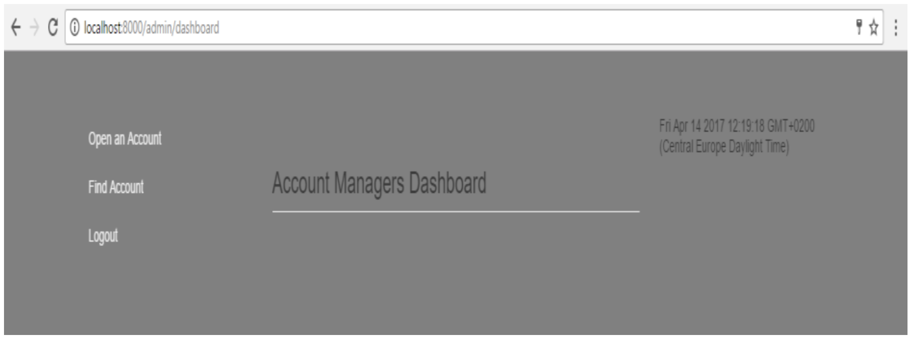
Fig 4: Flowchart on fragmented system

III. RESULTS AND DISCUSSION

Fig 4 to fig 8 shows the results of this work. The system designed displays all the transactions made by an individual within a recorded time by collating all the transactions and bringing them closer for quick access (Figure 4). It displays equally the beneficiary bank, transaction mode, date and time, amount, sender and receiver's account number on the already fragmented database. Genetic technique used as a model supports and promotes efficiency of the system such that the data are rightly arranged on a row, also showing the transactions made in accordance with the order of transfer. Reduction in communication cost by reducing the number of servers, easy and fast retrieval of data and closeness of data to the user are the outlined benefits of this model to the users. This work was realized by allocating already existing activities in the banks i.e., transaction of funds and payment of utility bills.

Admin Menu

This is comprises of Open Account, Search Account and Log Out (Figure 5). Open Account directs the admin to an interface that enables him to open an account for a new customer taking the customers details from the keyboard (Figure 6) and saving to the database for prospective use and after which a unique account number is generated automatically by the system. Search Account is usually important to switch from one account to another taking cognizance of the fact that in an average bank, over a million customer accounts may all be connected to the same database, searching with name or account number permits ease of transaction after the customers detail and statement is recovered from the database. Some other functions provided on this level are; update of an existing customers account, view and delete accounts in case "the customer decides to close their account". Logout, the admin logs out after being through with whatever job for the day.- Each admin has a unique login ID, with a login ID the system keeps track of user's input and how regular a user was active. This equally makes it possible for users to be held accountable of any stealing, mutilation or wrong input entered into the system, although there exists a counter security measure in the developed system which in this case is a session timer which always times out if system is idle for a minute.



Account Managers Dashboard

Fri Apr 14 2017 12:19:18 GMT+0200 (Central Europe Daylight Time)

Fig 5: Admin Main Menu

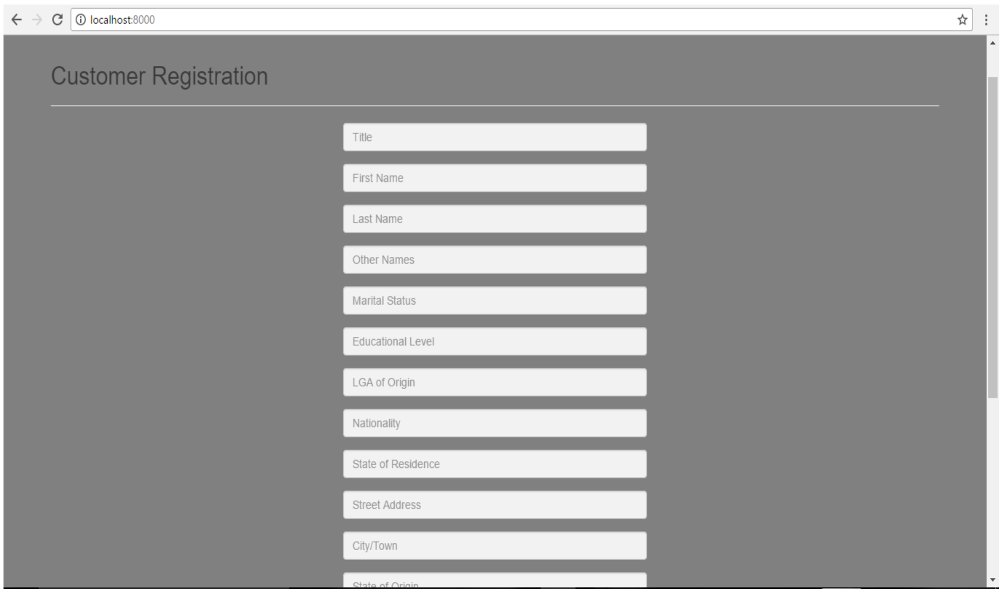


Fig 6: Customer Registration Interface

User Menu

Figure 7, User menu comprises of Account Balance, Account Activity, Account Statement, Utility/Bills Payment and Bank Transfer. Account Balance provides a user with the platform that contains details of all accounts linked by BVN. The user remaining balance in all connected accounts will be viewed with no access to printing. Account Activity gives a user opportunity to see the last five activities or transactions. Account Statement produces all transactions on the account during a stipulated period of time; the interval in period is often decided by the user of the account. The user chooses whether to print this statement or download and saving on their personal device for viewing without access to network at their leisure. In Figure 8, Utility/Bills Payment - A user makes other transactions aside viewing of statement, checking of account balance and so on. This interface avails a user the opportunity to pay for light bill, water bill and buying of data or recharge cards with ease from any of the bank accounts that has been linked with the help of BVN fragmentation. Bank transfer gives a user the opportunity to send money from any of the user's registered bank accounts to other accounts either to business associates or family.

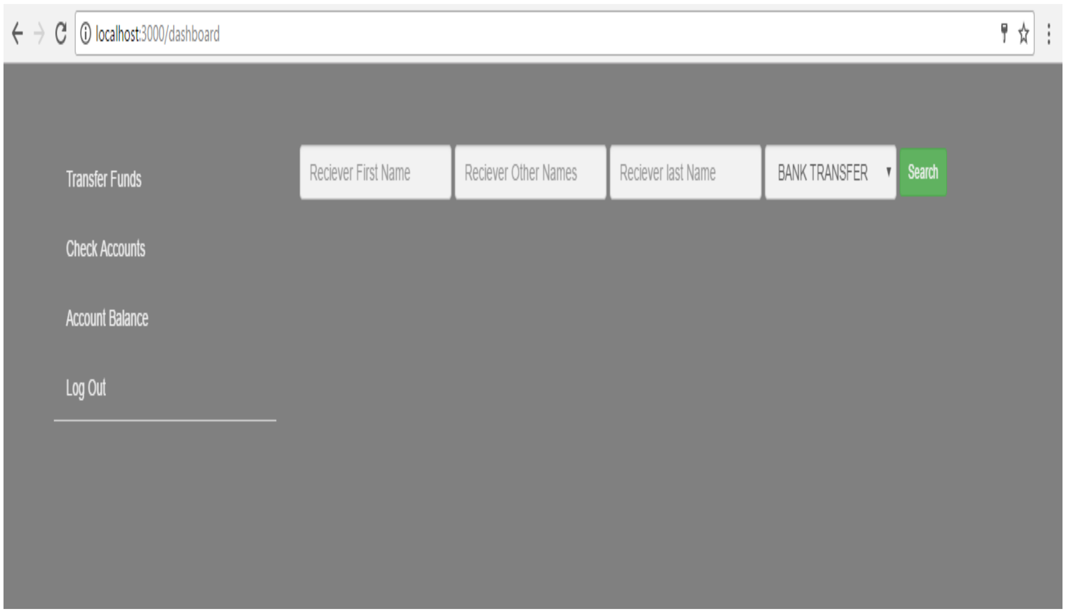


Fig 7: User Main Menu

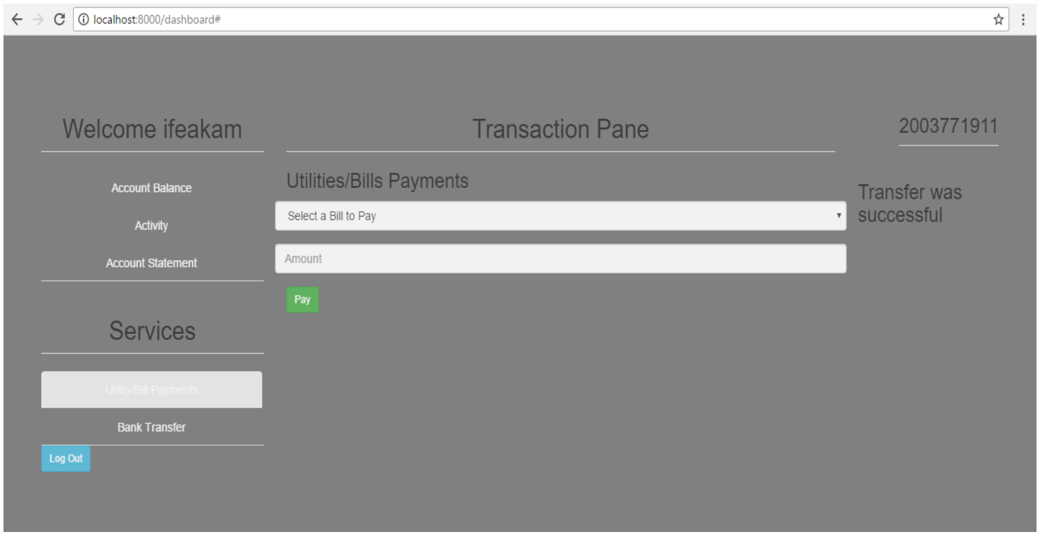
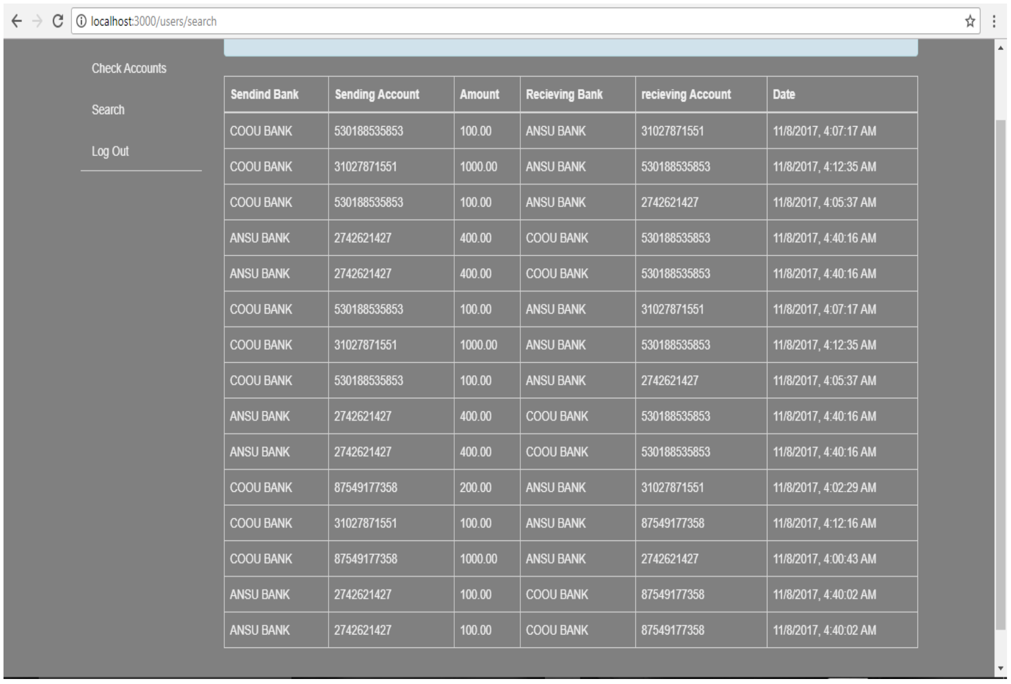


Fig 8: Utility/Bills Payment Interface



Sendind Bank	Sending Account	Amount	Recieving Bank	recieving Account	Date
COOU BANK	530188535853	100.00	ANSU BANK	31027871551	11/8/2017, 4:07:17 AM
COOU BANK	31027871551	1000.00	ANSU BANK	530188535853	11/8/2017, 4:12:35 AM
COOU BANK	530188535853	100.00	ANSU BANK	2742621427	11/8/2017, 4:05:37 AM
ANSU BANK	2742621427	400.00	COOU BANK	530188535853	11/8/2017, 4:40:16 AM
ANSU BANK	2742621427	400.00	COOU BANK	530188535853	11/8/2017, 4:40:16 AM
COOU BANK	530188535853	100.00	ANSU BANK	31027871551	11/8/2017, 4:07:17 AM
COOU BANK	31027871551	1000.00	ANSU BANK	530188535853	11/8/2017, 4:12:35 AM
COOU BANK	530188535853	100.00	ANSU BANK	2742621427	11/8/2017, 4:05:37 AM
ANSU BANK	2742621427	400.00	COOU BANK	530188535853	11/8/2017, 4:40:16 AM
ANSU BANK	2742621427	400.00	COOU BANK	530188535853	11/8/2017, 4:40:16 AM
COOU BANK	87549177358	200.00	ANSU BANK	31027871551	11/8/2017, 4:02:29 AM
COOU BANK	31027871551	100.00	ANSU BANK	87549177358	11/8/2017, 4:12:16 AM
COOU BANK	87549177358	1000.00	ANSU BANK	2742621427	11/8/2017, 4:00:43 AM
ANSU BANK	2742621427	100.00	COOU BANK	87549177358	11/8/2017, 4:40:02 AM
ANSU BANK	2742621427	100.00	COOU BANK	87549177358	11/8/2017, 4:40:02 AM

Fig 9: Output Format of The Fragmented System

IV. SUMMARY

A functional model for allocating already existing data in a distributed database system was developed as the new system. This new system creates a central financial technology solution that permits users to carry out financial transactions from all their bank accounts with the use of database fragmentation to separate the banks and their operations. It solves the problem of delay in processing data at the banks. It provided a unified system for all the banks for easy access. The model was built to add fragments to DDBMS for easy access.

V. CONCLUSION

A new approach to improve allocation of already existing data in distributed database system was proposed in this work. This approach is combination of two techniques namely database fragmentation and fragment allocation. These techniques are developed to avoid drawbacks of database fragmentation and data allocation like data redundancy and complexity of data redistribution problem, as well as to satisfy a meaningful level of data availability and consistency. This work takes us to a result which displayed an approach that significantly improved performance requirement satisfaction in distributed database systems. The system can vehemently boast of reduction of data transfer between sites, increased security, event logging being easier as it improves availability, it can equally control transactions hence lead to safer transactions, efficiency in performing transactions, it is not highly influenced by errors, data that are not required by local applications are not stored locally and it is easy to implement in any organization and industry. Keeping track of all the transactions made from this window for accountability. The system must be informative, robust, responsive, user-friendly and secure. System should be planned and designed to allow possible future expansion. System should equally be descriptive for easy understanding.

VI. REFERENCES

[1] Burrows, (2006)"The chubby lock service for loosely-coupled distributed systems".

[2] http//labs.google.com/papers/chubby.html. retrieved 2010-04-18.

[3] Bellatreche, L., Karlapalem, K. and Li, Q. (2008)"Algorithms and support for horizontal class partitioning in object-oriented database". Distrib. Paral. Databases, 8(2): 155-179.607.

[4] Chang, S.K. and Liu, A.C.(2012)"File allocation in a distributed database". Int. J. computInf. Sci, 11(5):325-340. 121,123.

[5] Ceri, S. and Pelagatti, G. (2016)"Correctness of query execution strategies in distributed databases". ACM Trans. Database syst.,8(4): 577-607. 38, 232, 242, 292.

[6] Karlapalem, K. and Li, Q. (1995)"Partitioning schemes for object-oriented databases", In proc. 5th Int. workshop on research issues on data Eng., 42-49.560.