Digital Innovations & Contemporary Research in SCIENCE, ENGINEERING & TECHNOLOGY
Vol. 9. No. 2, 2021

Article Citation Format
Enyindah, P., Oghenekaro, L.U. & Ojetunmibi, T. (2021): A Comparative Study of a Gitbash-Generative Model and Fuzzy Query Processing In A Distributive Database System. Journal of Digital Innovations & Contemp Res. In Science., Engineering & Technology. Vol. 9, No. 1. Pp 51-64

Article Progress Time Stamps
Article Type: Research Article
Manuscript Received: 14th September, 2021
Review Type: Blind
Final Acceptance: 30th November, 2021

A Comparative Study Of A Gitbash-Generative Model And Fuzzy Query Processing In A Distributive Database System

Enyindah, Promise. Oghenekaro, Linda Uchenna. & Ojetunmibi, Taiye
Department of Computer Science
University of Port Harcourt,
Port Harcourt, Nigeria.
Email: promise.enyindah@uniport.edu.ng; linda.oghenekaro@uniport.edu.ng; taiyeojetunmibi@gmail.com
Phone: +2348036710489

ABSTRACT

This research work proposes an enhanced query processing system to improving the performance of big data. The system employs a GitBash-Generative Model as part of refinements to an existing Fuzzy Query Processing in a Distributed Database System. The availability of intended software that will confront the problem of storage contents and execution plan for query processingis a vital challenge facing big data in a processing system. This in turn results to information loss, memory pressure in a database, reduction in concurrency, missing record of vital document in an enterprise, inflation of the central processing unit (CPU) and threat to the quality of a good query processing in a distributed database environment. In this paper, the proposed system is compared for efficiency with a query processing algorithm known as the Fuzzy Query Processing, which is emerging as an alternative to more conventional technique for query processing in a distributed database system. The system addresses the problems of latency and the inability to transform and store queries. Dynamic simulations were performed using e-library server/data.json dataset to test and evaluate the performance and efficiency of the two systems in query processing operation. The results of simulation study showed impressive result and proves the GitBash-Generative Query System more preferred over the Fuzzy Query Processing System.

Keywords: GitBash-Generative Model, Distributed Database, Fuzzy Query Processing

1. INTRODUCTION

Query processing is an efficient activity that aids Database end-users to retrieve specific information from the database. Due to Poor query execution plan and the absence of an improved aggregate query method which impose threats to qualities of good query processing in distributed databases, causing memory pressure in a database, inflating the central processing unit (CPU) and an overall reduction in concurrency.Since the volume of data use in any organization is progressively increasing in it seize, thereby resulting or demand a large storage location or space system. The availability of intended software that will confront the problem of storage contents and execution plan for query processing is a vital challenge facing big data in a processing system, resulting to information loss and missing record of vital document in enterprises.

To obtain the relevant information is a key part of any modern information management system as time critical industrial systems need to be well informed to minimize losses or cost of operations, improve the working conditions and also create the enterprise. Information can be re-generated as new and passed via industrial control systems over long distances in-order to operate them more efficiently. Information may be used to guide buy and sell market ideas/products, help coordinate traffic and improve health-care decision making schemes, security and emergency units. Because information can be converted into different forms, it is well suited to generative models. Generative models can help build more efficient systems that are robust to making decisions without the usual cost implications in memory or any hand-engineered approach. [1] presented an approach to expand real-time database query solutions further by using dynamic probabilistic models. Whether this approach is sufficient in itself remains to be tried and tested effectively. The ultimate aim of any information system is to obtain relevant messages or codes from noisy or contaminated data distributions.

The origins of information theory could be traced to the works of Shannon [2] and has deep probabilistic roots. "Query" is a unique Database terminology that is used in Database Management Systems (DBMS).This study tends to proffer solution and addresses the problem of latencies, inability to transform and store queries, and poor query processing plans for future improvement of distributed databases, using an unsupervised learning approach such as GitBash-Generative Model. The study compares the proposed Gitbash deep-generative system for efficiency with a query processing algorithm known as the Fuzzy Query Processing, which is emerging as an alternative to more conventional technique for query processing in a distributed database system. The researcher will make use of e-library server/data.json dataset to test and evaluate the performance and efficiency of the two systems in query processing operation. The rest of this paper are arranged in sections as follows: Section 2 discusses related works. In section 3, we present the methodologies. The remaining sections are as follows: 4. Fuzzy Query Processing Technique 5. GitBash-Deep Generative Model 6. Results and Discussion. 7. Comparison of the two techniques 8. Conclusion. 9. Further work

2. RELATED WORKS

The education sector in Nigeria has experienced a huge growth especially in the area of available data. Furthermore, the emergence of social networking has continued to boost the growth of data usage in Tertiary Institutions in Nigeria. However, this trend of growth comes with the problem of big data management. The study also view big data as the joining of data management concepts that Tertiary Institutions to store, organize, manage and manipulate large amount of datasets and still be efficient in speed so as to gain the right insights. Data Mining can also be described as the process of extracting vital information from voluminous data [3,10].

Big data mining is the capability of extracting useful information from these large datasets or streams of data, which was not possible before due to data's volume, variability, and velocity. Big data is a massive volume of both structured and unstructured data that is so large that it is difficult to process using traditional database and software techniques. Big data technologies have great impacts on scientific discoveries and value creation. Structured (numerical) and unstructured (textual) are two main types of data forms in big data [4]**.** A Deep Generative Model is a powerful way of learning any kind of data distribution using unsupervised learning and it has achieved tremendous success in just few years. All types of generative models aim at learning the true data distribution of the training set so as to generate new data points with some variations. But it is not always possible to learn the exact distribution of data either implicitly or explicitly and so there is need to model a distribution which is as similar as possible to the true data distribution. However, neural networks can be used to model a function which can approximate the model distribution to the true distribution. A good information gathering algorithm will maximize computation power and algorithmic accuracy to gather, analyze, link and compare large datasets, to also enable the drawing of large datasets to identify patterns in order to make economic, social, technical and legal claims [5,9].

Big data is a massive volume of both structured and unstructured data that is so large that it is difficult to process using traditional database and software techniques. Big data technologies have great impacts on scientific discoveries and value creation. Structured (numerical) and unstructured (textual) are two main types of data forms in big data. LSTM concepts have been analyzed by numerous researchers. [6] discussed a Long Short-Term Memory Recurrent Neural Network Framework for Network Traffic Matrix Prediction. Most of the decisions that network operators make depend on how the traffic flows in their network. However, although it is very important to accurately estimate traffic parameters, cur-rent routers and network devices do not provide the possibility for real-time monitoring, hence network operators cannot react effectively to the traffic changes.

3. METHODOLOGY

In order to achieve the set goal for this research work, the researchers employs a structural system analysis and design methodology (SSADM). Good methodology allows simplification of modular component have an enhancement criterion for use. This software development methodology uses a recursive procedure and object-oriented development method in manipulation of the entire system by splitting into subsystems and modules. The methodology, software processes are basically the same in use, parts of the process defining to the topmost level structure. The overall software development phases which include the analysis and design will implore occurring data and its performance, which depict the knowledge of fragment allocation in a distributed database System in a system. The SSADM thoroughly analyses the entire modeling process, as well as the flow of data in the system.

It also helps module the project design i.e. splitting the program into segment for easy and efficient execution. This method is efficient as tackling occurring defect on fragment and allocation in a distributed system. The design and development demands repeated investigation to ensure both developers and users has elaborate knowledge of the existing and proposed system. In Analysis system design, investigation to ensure that the developer, user, and customer have a common understanding both of what is needed and what is proposed. However, in this study the prototyping model was used to demonstrate feasibility of data performance on servers. The architectural frame work of the system will be framed on the three arms; Collection, Classification, Analysis.

1. 1.Collection: the user are privileged of rendering feedback on problem associated with the use of the equipment, collection of feedback can come in a form to accommodate several of challenges that may rampage the system. Information could as well be collated from the site (severs) which will yield accurate result to be analyzed and queried
2. Classification: sensitive information retrieved from the event log or from feedback from user will be carefully scrutinized and classified in order to achieve optimum results. Small amount of collated data can be summarized to determine how fragment are analyzed, located and solved
3. Analysis: haven collected and filtered the requisite information, the information should be sized to get the required or target solution

4. FUZZY QUERY PROCESSING IN A DISTRIBUTED DATATBASE

The application ofFuzzy query processing to a distributed database as proposed in [12] Rohan et al 2016, is important especially in terms of user-friendliness and the ability to answer vague human queries. Figure 1 show the architectureand gives details of the system functions. Rohan proposed three-pronged fuzzy logic-based querying datasets in distributed databases which is used for solving ambiguous queries and also incorporating preferences.The system is composed and functions as follows:

- **Crisp Set for Query Input:**The crisp set is the collection of objects which have some properties distinguishing them from other objects which do not possess the properties. Furthermore, in a crisp set, an element is either a member of a set or not. E.g. (apple, bread, banana)
- **Fuzzification:**The fuzzification component uses fuzzy logic to convert the crisp sets to fuzzy values (fuzzification) through the assignment of membership degree. The membership value can range from 0 (not an element) to 1 (a member of the set)
- **Inference System:**The third component on the existing system flow is the Fuzzy Inference System. This component carries out the process of formulating the mapping from a given input to an output using fuzzy logic.
- **De-Fuzzification:**The fourth component of the existing system flow is De-Fuzzification process. This component is the inverse process of fuzzification where the mapping is done to convert fuzzy dataset to crisp datasets.
- **Master Node:**The fifth component of the existing system flow is the Master Node. This is the source point for query results in the network. It consists of data pathways intersected and distributed in the network.
- **Query Output Aggregation:** This component derives group and subgroup query results by analysis of a set of individual data entries. Query Results Output: The seventh component of the existing system flow is the query results output which is displayed to the user. The result is the display of the aggregated query output.

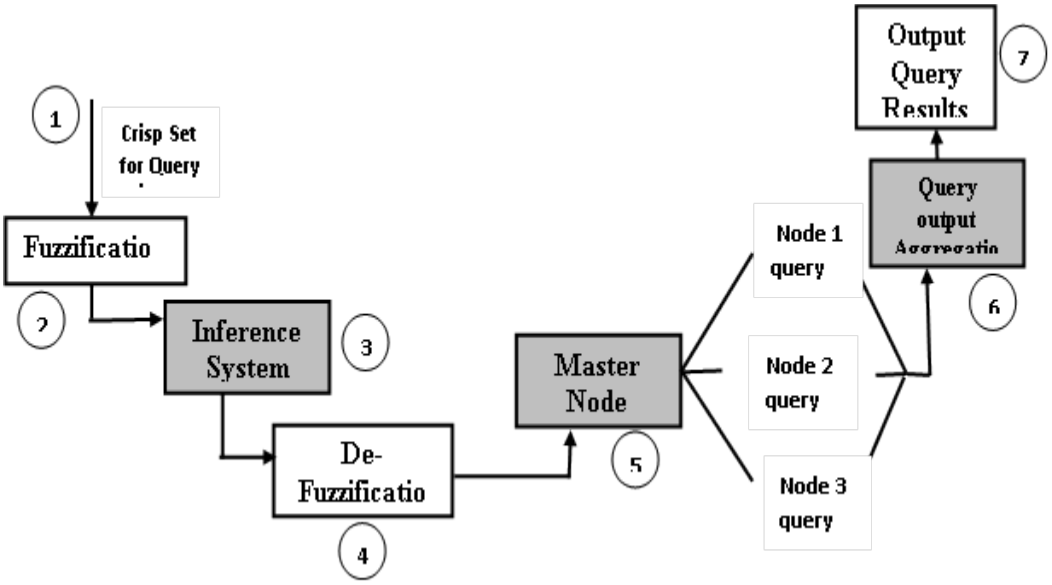

Figure 1: Fuzzy Query Processing system Architecture (Source: Rohan et al, 2016)

5. CONFRONTING THE GITBASH-DEEP GENERATIVE MODEL

The proposed system is an improvement of the fuzzy logic based query processing algorithm that adopts an unsupervised learning approach using a Gitbash-deep generative model. The system employs mongo DB for data storage, which is capable of distributing big data with the help of deep generative algorithm which assist in fast processing and searching processed of an unstructured data.The data to be search are key into the search term. The sorted data will immediately display on the data structure. The system is presented with unlabeled, uncategorized data and the system's algorithms act on the data without prior training. The output is dependent upon the coded algorithms. Figure 2 show the Architecture and details of the systems functionalities. The specification or Requirement for performing simulations on the proposed system is given in Table 1. The requirement specifications typically includes the components necessary for implementing a given software process. It also includes some key information about the type and nature of application domain (such as the use of generative domains e.g. the use of deep generative algorithm domain). It must be emphasized that these requirements may change depending on the Application domain.

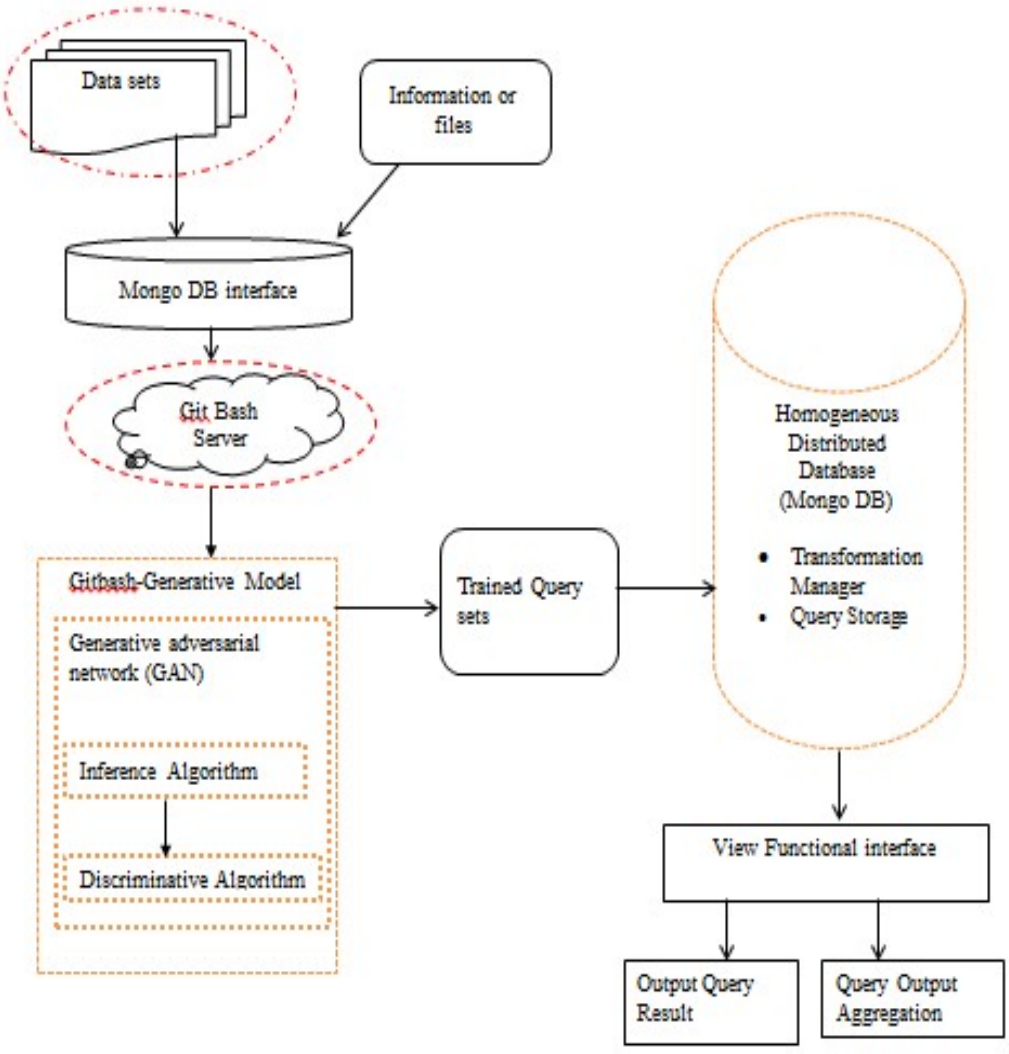

Figure 2: Proposed GitBash-Deep Generative model System Architecture

The proposed system uses Git-Bash Deep generative model for query processing system (artificial system) to evolve a set of system parameters. The proposed Systems components is shown in figure1; these includes:

(i) Dataset: The dataset use is called data.json from the existing system e-library server/data.json (query). It is the crisp set for query input. The dataset contains all the unstructured big data from the cloud e-library server/data.json (query). The dataset was generated from existing system e-library server/data.json (query) and is inserted into the mongodb and send to gitbash server where command is been given and then sent to the gitbash- generative mode for conversion from its original crisp set to query set that is now understand by the user. The proposed model contained one thousand (1000) datasets extracted from more than one million dataset in the existing system e-library server/data.json (query).

(ii) Information or Files: this component holds the unstructured data that are inputted into the mongodb. These data are store in the mongodb interface before sending to the gitbash server. This component is also another interface where information is sending to the model. It holds information or file from different users for conversion from its original state to query set for proper understanding.

(iii) Mongo DB: This component receives all the dataset and the information of the dataset from their respective source and stores them and sends them to the git bash server. The mongodb is a large storage location use to store the information or files and the dataset from the existing system e-library server/data.json. The mongodb encrypts and secure the dataset. The operation perform by the mondodb is done in the MongoDB audit log's which displays the dataset on an html application. There are created tool in the MongoDB audit log's which allows the user to conduct multilevel search queries on the audit log data. The dataset goes into the MongoDB audit log's for auditing. The auditing system writes every audit event on the dataset to an in-memory buffer of audit events. MongoDB writes this buffer to disk periodically. The auditing system writes and display these datasets to the log file in a JSON format. The json format of an audit dataset event is done by typing the attribute types (ex. string/int/timestamp), ("System Event Audit Messages"). This command then sends to the discriminative algorithm to apply rules and parses an array of these objects as its main data source. To execute complex nested queries, the mongodb uses a complex Boolean expression in form of a decision tree. For the nested query operation, the dataset has expressions and groups which denote a level of nesting. The query operation is done in the following format.

```
{
conditions:[A],
groups:[
{
conditions:[B, C],
groupOperator: 'AND',
groups:[{
conditions:[D,E]
groupOperator:OR
}]
}
]
}
```

In the JSON dataset format expressed on a tree as shown in code above, the root of the tree would be the first group operator with the condition A as the left leaf and an 'OR' node on the right for nested group's operator. This node is connected with the conditions inside of its group which is B ∧ C and D ∧ E. Having the query expressed like this makes it easy to walk the tree and convert the expression into Postfix format before processing the query. From the search query in a tree format above, the full query is: A and (B and C or (D and E)). Postfix query is: ABC **AND** DE **ANDORAND**.

(iv) **The Git Bash servers:** The gitbash server component is use to connects the big data from the Mongo DB to the discriminative algorithm. The gitbash server contains the query mechanism that is use to enables the dataset to accommodate the Microsoft window storage location which provide emulation layers. The gitbash server component is use to control and manage the big dataset in the window storage location and allows the users to issues different command and format to the big data. The operation of the git bash server is done by first run or lunch the git bash. The first lunching of the Git Bash, allows the Generative algorithm to have a link on the MongoDB platform, this is done by typing in the code: cd with a space type documents/query, then press enter and type npm with a space type run with a space type serve then press enter. These commands will immediately link the Generative algorithm to have a link on the MongoDB platform, which is the first command execution in the improved query processing. The second operation on the git bash is by lunching it the second time, the second lunching of the git bash server is to connect the datasets in the MongoDB to the discriminative algorithm to display on the design interface. The second operation of the Git Bash server is done by simply right click on its icon and type in the code: cd with a spaces type documents/query/server then press enter and type node with a space type index.js then press enter. This code will immediately connect the algorithm and the MongoDB. When the two operation of the git bash are running simultaneously at the same time, it will display data connection successful. This means that the connect to the program for execution successful.

(v) **Gitbash-Generative Model:** this component receives the data set from the cloud e-library server/data.json and send to the generative adversarial network. The dataset then goes into the inference engine to assign the right rule that will be used to convert the scrip dataset by the algorithm. The gitbash generative model component contains some other interface which are inference engine, discriminative algorithm that perform the conversion of the scrip set to the query set.

(vi) T**he generative adversarial network** is a component in the gitbash-generative model that contains the inference engine and the discriminative algorithm. This component performs the main function of the conversion. It uses the unsupervised learning method on the data set. The rules are store in the inference engine.

(vii) **The inference Engine**. The inference engine component contains several rules use for conversion from the original scrip dataset to the require query set for querying the big data. The inference system retrieved rules from the rule base which then produce the require output query. Each of the rule determined the type of query needed to perform. Once the unstructured database is converted, the corresponding input query sets are passed to the inference engine that process current inputs using the rules retrieved from the rule base, then produces an outputs query set. This component contains rules use to train the algorithm depending on the types of data set. The inference engine was used to build the model. The inference engine specifies the features of inputted datasets based on a given label in the application, which use the output probabilities from the Generative Adversarial Network to make decisions on the most likely variables or parameters that influence the data generating stage. It stores the value in a local variable, and then using that variable in the control condition. local variable was used to hold the length of the logData. The rule use by the inference engine local variable control condition, which is;

```
for (var x=0, arrLength=logData.length; x<arrLength;x++){
//logic
 }
```

(viii) **The discriminative algorithm**: this component contains the algorithm use in the proposed model. When the data set in the inference engine assign the right rule to be used for training the algorithm, the algorithm will then act upon the dataset to produce the desire result. The discriminative algorithm evaluates the dataset by apply step that guide the conversion. When the json format of an audit dataset event, that is attribute types (ex. string/int/timestamp), (“System Event Audit Messages”) or command is sends to the discriminative algorithm, it then applies rules and parses an array of these objects as its main data source. The postfix array is then evaluated in a stack method to filter down the audit log data.

The algorithm looped through the query one time and scanned through the log file for each condition to find matches.

(ix) **Homogeneous Distributed Database:** This component enables the converted query set to match with the corresponding answer of the request (i.e. prestored datasets in the database). Furthermore, the homogeneous distributed database system is a network of two or more databases (with same type of DBMS software) which can be stored on one or more machines on a network (nodes). So, in this system data can be accessed and modified simultaneously on several databases in the network.

(x) **Query Output Aggregation:** This component derives group and subgroup query results by analysis of a set of individual data entries.

(xi) **Query Results Output:** This component displays all the queries in the mongo database storage to the user. It shows all the aggregate components of all the dataset at the same time.

Table 1: Sample Input/output Specifications for the Deep generative algorithm for Query Processing System

Field name	Data Type	Field Size/Width	Decimal	Index
Natrum carbonicum	Character	20	no	id 1
Rheum officinals	Character	15	no	id 2
Benzocaine	Character	20	no	id3
Sodium	Character	15	no	id4
Menthol	Character	15	no	id5
White Alder	Character	20	no	id6
Flouride	Character	20	no	id7
Ethanol	Character	20	no	id8
Pollen	Character	15	no	id9

6. RESULTS AND DISCUSSION

6.1 GitBash-Deep Generative Query Performance Results

Table 2 and 3 show the performance ranking and evaluation of the git-bash generative model result for the proposed system. The variables used during coding are deep n, git-bash, search item, APL, journal, and mongodb. The values in the table were taken during runtime and are measure in second, the highest values recorded in the table 2 is 11. The graph of time against performance is plotted in figure 3 and figure 4 shows Data connection on Git Bash Server to Gitbash Generative Model

The result from the graph shows high increase rate of the variables in the proposed system when compared with the existing system. The parameters use in the graph includes processing speed, scalability, graphical user interface, availability and usability, query storage, and transformation ability. The graph is plotted efficiency against parameters. The highest value of the efficiency rate is 100. On the vertical axis (efficiency rate %) 20 units represent 1cm. the result from the graph shows the increase of each of the parameters in the proposed system which indicate an increase in performance, these shows that the performance ranking of the proposed system is of increase with better performance..

Table 2: Performance Ranking of Query Results for the Proposed GitBash Deep Generative Algorithm

Deep G. Rank	Git Bash Rank	Search Item Rank	API Rank	Search Journal Rank	MongoDB Rank
8	4	2	3	12	9
10	3	3	11	4	5
11	6	9	6	7	6
9	10	4	5	8	3
10	9	10	8	9	10
9	5	7	9	6	8
Second (s) 05.7	7.06	05.09	04.09	03.04	03.04

Table 3: Performance Evaluation of the Proposed GitBash-Deep Generative Model

S/N	PARAMETERS	Efficiency Rate (%)
1	Processing Speed (PS)	98
2	Scalability (S)	92
3	Graphical User Interface (GUI) Quality	95
4	Availability and Usability (AU)	87
5	Query Storage and Transformation Ability	100

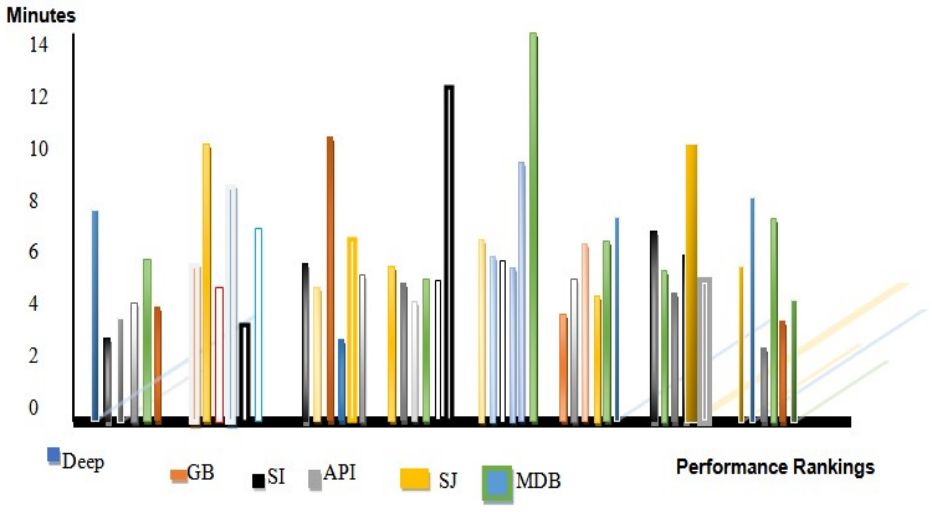

Figure 3: GitBash Deep Generative model Query Results Performance Ranking Chart for the

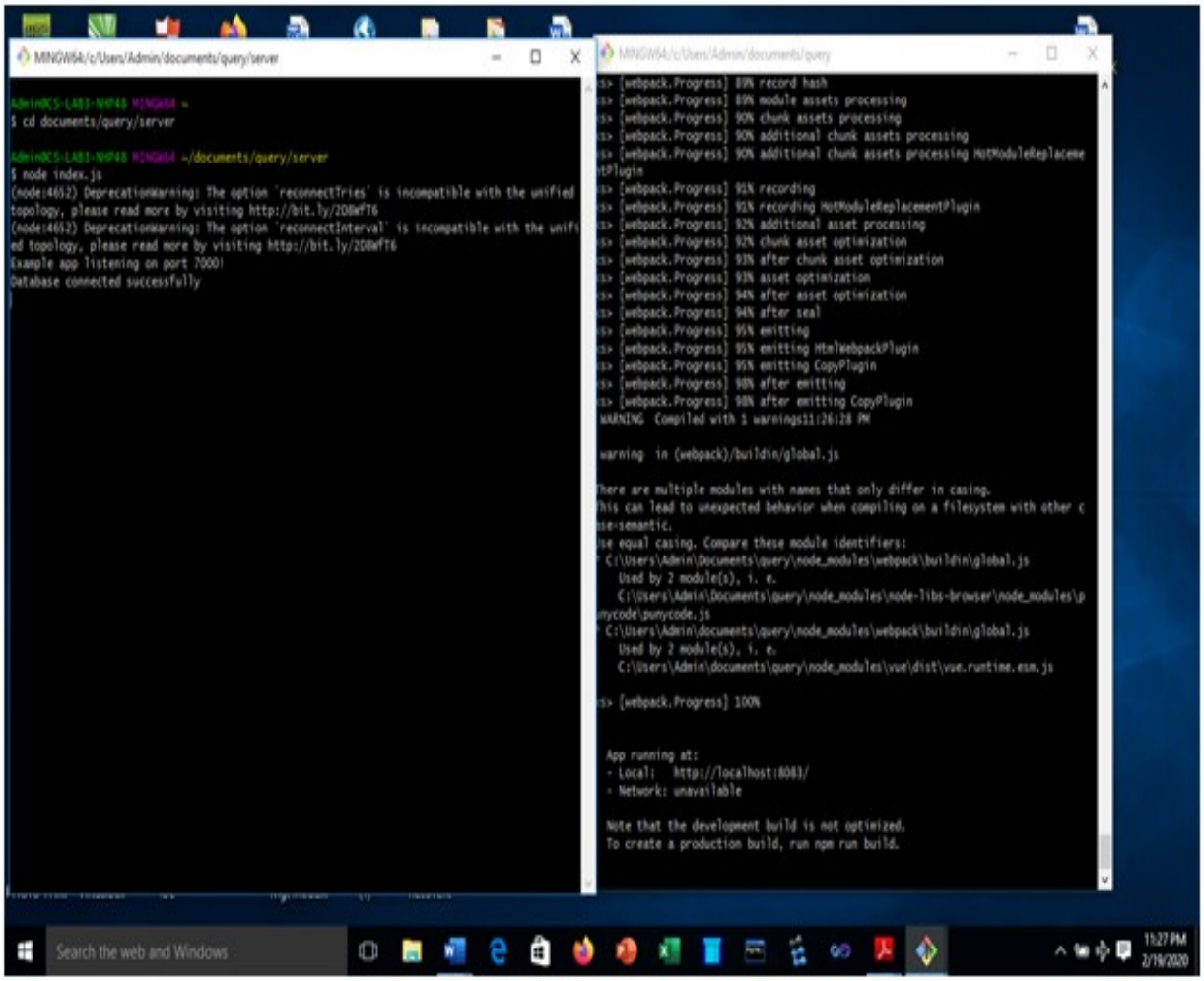

Figure 4: Data connection on Git Bash Server to Gitbash Generative Model

6.2 Fuzzy-Logic Based Query Processing Performance Result

The fuzzy-logic based query processing performance ranking results is tabulated in table 4. The field in the performance of the fuzzy query result include fuzzy rank, aggregate, year, price rank, KM rank and CYL. The figures in the tables were taken at random during runtime with respect to time in minutes. The fields in the column represent the variables use during code implementation, and the graph of time against performance rank is plotted in figure 4.8. The highest values in the performance table of fuzzy query result are 13.

The scale of the graph on the horizontal axis is 1 unit to represent 1cm, while on the vertical (minute) axis 2 units represent 1cm. each of the colour represent each attribute in the performance ranking table, see figure 5 display for the plot of time against performance for the Fuzzy query system

Table 4: Fuzzy Query Performance Ranking Results

REG NO:	FUZZY RANK	AGGREGATE RANK	YEAR RANK	PRICE RANK	KM RANK	CYL RANK
1	1	1	3	9	1	9
4	2	3	4	7	3	10
5	2	4	13	1	5	11
3	3	6	9	8	4	2
12	3	9	11	3	12	6
2	4	9	5	4	2	1
11	4	8	12	6	11	5
6	5	10	1	13	6	12
7	6	2	2	12	7	3
8	7	5	6	11	8	4
13	8	6	10	2	13	7
9	8	8	7	10	9	8
10	9	11	8	5	10	13
		10:03 Minutes	**08:05 Minutes**	**08:03 Minutes**	**11:01 Minutes**	**07:06 Minutes**

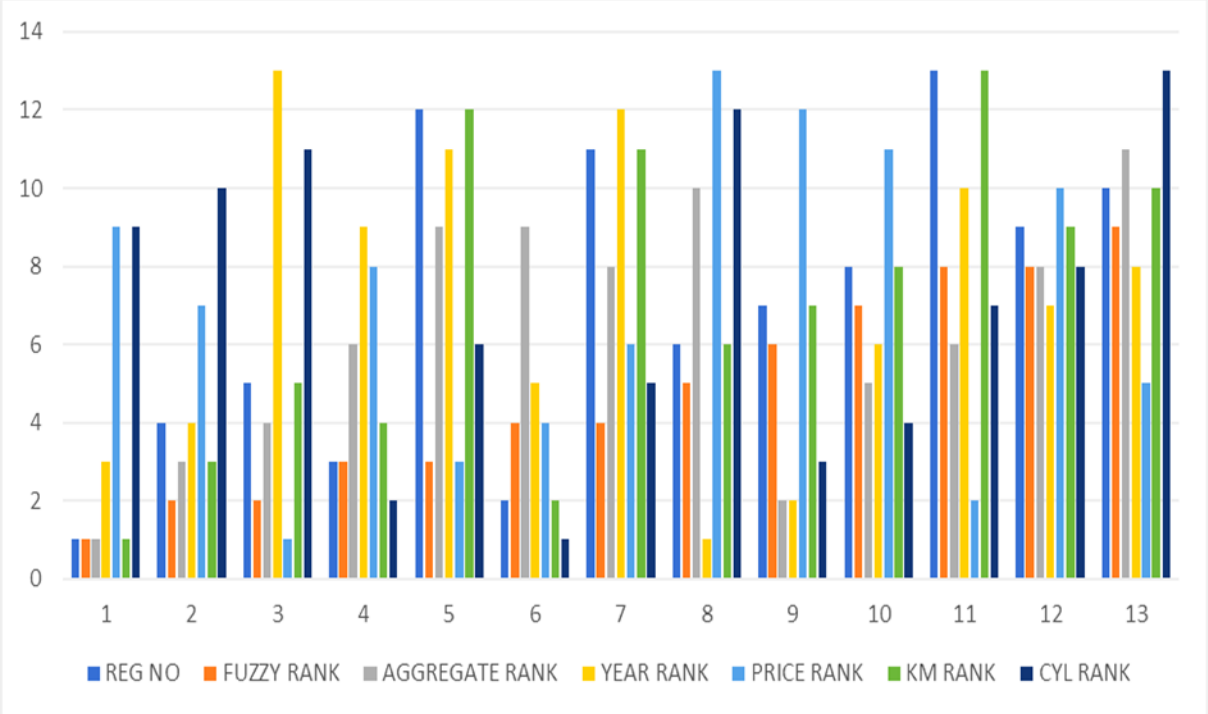

Figure 5: Fuzzy Query Performance Ranking Chart

7. COMPARING SIMULATION PERFORMANCE RESULTS OF THE GITBASH-DEEP GENERATIVE ALGORITHM AND FUZZY-LOGIC BASED QUERY PROCESSING ALGORITHM

Several runs and simulations were performed using a benchmark dataset to evaluate the performance ranking and efficiency of both systems in query processing in a distributed database. The parameters (dataset) used includes; processing speed, scalability, graphical user interface, availability and usability, query storage, and transformation ability. The performance evaluation ranking result of the two systems are tabulated in table 5 and figure 6 show an Efficiency(rate %) against Parameters graph of the both systems. The highest value of the efficiency rate is 100. On the vertical axis (efficiency rate %) 20 units represent 1cm. the result from the graph shows the increase of each of the parameters in the proposed system which indicate the increase in performance while the Fuzzy system parameters decrease to zero (0), these shows that the performance ranking of the proposed system is of increase, hence better than the Fuzzy system.

Table 5: Performance Evaluation of both Proposed and Existing Systems

SN	ROHAN ET AL (2016)	%	%	PROPOSED SYSTEM
1.	Processing Speed (PS)	78	98	Processing Speed (PS)
2.	Scalability (S)	61	92	Scalability (S)
3.	Graphical User Interface (GUI) Quality	47	100	Graphical User Interface (GUI) Quality
4.	Availability and Usability (AU)	72	95	Availability and Usability (AU)
5.	Query Storage and Transformation Ability	0	100	Query Storage and Transformation Ability

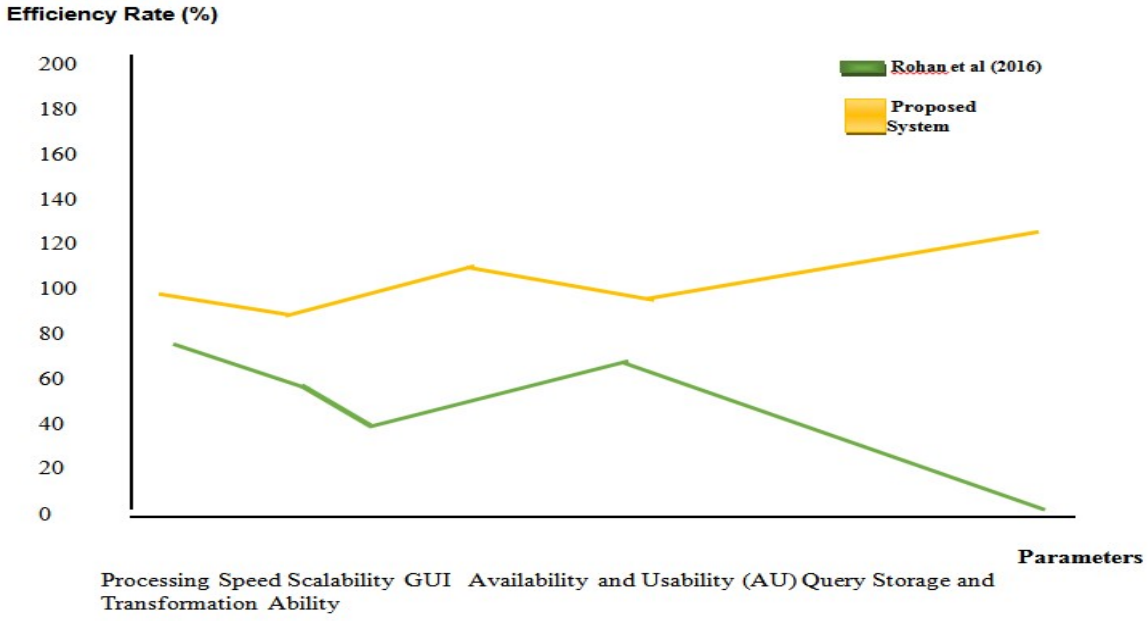

Figure 6: Graph of Efficiency against Parameters

Comparison results show that GitBash-Deep Generative model shows great improvement in performance and efficiency as compared to the Fuzzy Query Processing system in the following aspect;

i. **Latency Reduction:** This is because the proposed system uses a Git-bash deep generative model which consist of a hybridized algorithm (i.e. inference and discriminative) to arrive at a query conclusion that is reached on the basis of evidence and reasoning.
ii. **Best Query Result for Structured and Unstructured Datasets**: Git-Bash Deep Generative Models algorithms can be trained using different data formats, and still derive insights that are relevant to the purpose of its training. For this implies that the proposed Git-Bash deep generative models' algorithm can uncover any existing relations between pictures, social media chatter, industry analysis, weather forecast and more to predict future stock prices of a given company.
iii. **No need for manual labeling of datasets before query processing**:The proposed system supports self-automated query processing which also boycott the need for manual labeling of data. This is because; labeling process is simple but time-consuming. For example, labeling photos "dog" or "muffin" is an easy task, but an algorithm needs thousands of pictures to tell the difference. Other times, data labeling may require the judgments of highly skilled industry experts, and that is why, for some industries, getting high-quality training data can be very expensive.
iv. **An improved Graphical User Interface, Technique for Query storage and transformation for users of Homogeneous Distributed Database:**The proposed system has enabled user-friendliness in the usage of homogeneous distributed database. In addition, users of the proposed system can be able to document, store and transform query sets in the distributed database

8. CONCLUSION

In this study, the researchers have presented an enhanced approach to query processing through the application of a Git-Bash Deep Generative Model and Mongodb. The enhanced approach depicts an unsupervised learning technique for query processing which is also a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data. The findings of this study are recommended to database administrators and analysts in e-library environments, software developers and researchers with keen interest in the study area. This is because data management and request via queries are becoming complex day by day. In other words, the need for an improved query processing using Git-Bash Generative Model is highly indispensable.

9. FURTHER WORK

The limitations of the research should be improved in the study especially in a sophisticated application software device that will recognize real-time unstructured query data for homogeneous distributed databases. In addition, also improvement should integrate on other complex NoSQL databases such as Apache Cassandra, Hadoop and Mapreduce to the proposed system in future.

REFERENCES

[1]. Shammana, J. (2011). A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization, In J. D. Schaffer, (ed.), *Proceedings of the Third International Conferenceon Genetic Algorithms*, 51-60.

[2]. Bengio, B. (2015), Couprie, M. &Valduriez, P. 2015. Overview of Parallel Architectures for Database. *TheComputer Journal*, 36, 734-740.

[3] Francisca O. (2018), Information gathering methods and tools: A Comparative Study, *Research gate Journals, https://www.researchgate.net/publication/326688869, 7(9), 1-6*

[4] Emoghene O. (2018), Information gathering methods and tools: A Comparative Study, *Research gate Journals, https://www.researchgate.net/publication/326688869, 7(9), 1-6*

[5] Kelvin T. (2016), Big Data: Understanding Big Data, Engineering and Applied Science, *Aston University, England, Research Gate Publications, https://www.researchgate.net/publication/291229189, 56 – 61*

[6] Abdelhadi A. (2019), A Long Short-Term Memory Recurrent Neural Network Framework for Network, *International Journal of Computer Applications (IJCA), 7(34), 22-27*

[7]. Komal S. (2016), An Overview of Distributed Database Management System, *International Journal of Computer Science and Information Technology Research (IJCSITR), 4(2), 348-350*

[8] Kossmann, D. (2000). The State of the Art in Distributed Query Processing. *ACM ComputingSurveys.* 32(4), 422-469.

[9] Rajni B,. S.Bhanot, A.Mangla (2016), Fuzzy Query Processing in Distributed Databases, *International Journal of Advanced Computational Engineering and Networking, 4(1), 68 - 73*

[10] Saurabh V. (2015), Cloud Computing using Evolutionary and Swarm-based Algorithms Optimization and schedulin Technique *in Relational Database Management Systems, International Journal of Communication Network Security. 9, 1231-1562*

[11] Tam, K.Y. 1992. Genetic algorithms, function optimization, and facility layout design. *EuropeanJournal of Operations Research*, 63(2), 35-38

[12] Rohan et al (2016) researched on Fuzzy Query Processing in Distributed Databases. pronged fuzzy logic-based technique as a layer of computation. *International Journal of Computer Science and Information Technology Research (IJCSITR), 4(2), 228-360*