ISSN:0975-9646



# Functional Dependency Based Data Distribution by Using Association Patterns

Nisha

*Abstract*— The age of large database is now an big issue. But the traditional data analytics may not be able to handle such large quantities of data. So researchers try to solve to develop a high performance platform to efficiently analyze and maintain the algorithms. Here proposed work has resolve this issue of digital data security by finding the relation between the columns of the dataset which is based on the highly relative association patterns. So columns are distribute on different sites which reduce overall dataset size. So for increasing the security of data on sites Advanced encryption algorithm was used. Experiment is done on real dataset which is named as adult dataset. Comparison is done with existing techniques and results shows that proposed work is better as compare to other previous approaches on the basis of evaluation parameters.

**KEYWORDS**: Distributed Data, Data Mining, Encryption, Effective Pruning, Functional Dependency..

## I. INTRODUCTION

As the cost of maintaining the dataset goes raise day by day so distributed database is come in existence. This is because of high price of the storage disks. But these days cost of storage s=disk is wuit low as compare to few decades before, but at the same time ssecurity of the datset need to be increase from different intruders. As these distributed database is useful for world wide web, social media, military, etc. So defining the distributed database can be done as: A gathering of data from the different data provider is distribute on different datacenters or computers. As various sites has its own automatic processing capability but some time it need to be calculate by various servers parallel then distribution of data is required, this is term as load balancing. Data in a distributed database is divided into fragments. Each fragment is a set of data items that are managed by a computer, or rather a local database management system in the computer. From now on we call the pair computer and its database management system a node. A node in a distributed database may include several fragments, and a fragment may belong to several nodes. Usually it is preferred that a node holds exactly one fragment of data, and a fragment belongs to exactly one node. It simplifies distributed database management and cooperation between database nodes. Since the definition of a distributed database includes a set of local applications, or databases, it is natural to compare distributed and traditional non-distributed databases. It turns out that the distributed database is very close to a traditional database. This implies that traditional database results can be used or modified to suit distribution. The areas to consider are the ACID-properties, data independence, data redundancy, physical implementation, recovery, and concurrency control. ACID-properties.

There are hundreds of works proposed for providing privacy for the data for different requirements. So classification of those methods are done on the basis of following measures: •Data sharing

•Data Perturbation

•Data Mining Algorithm

•Rule suppressing

•Privacy conservation.

The first point discuss the sharing of data. Some of the approach have been proposed and developed for data centralization, while others refer to a data sharing situation. Data Sharing situation can further be divide into horizontal data partition and vertical data partition. Horizontal partition refers to these cases where data is divide row wise on different servers or stations or datacenters. While vertical data partition, refers to the cases where all the data is divide in column wise on different servers or stations or datacenters. The second dimension discusses to the data modification scheme. In general, data modification is used in order to modify the original values of a database that needs to be released to the public and in this way to ensure high privacy protection [6, 8]. It is important that a data modification technique should be in concert with the privacy policy adopted by an organization.

## II. RELATED WORK

In [14] present a hybrid discovery algorithm called HyFD, which combines fast approximation techniques with efficient validation techniques in order to find all minimal functional dependencies in a given dataset. While operating on compact data structures, HyFD not only outperforms all existing approaches, it also scales to much larger datasets.

Li et al (2013), problem of finding the minimal set of constants for conditional functional dependency present in used dataset. Here minimal set of conditional functional dependency is obtained by minimal generator as well as by clousers of those sets. Here proposed work has find the pruning criteria so overall work get reduce and unwanted generator, closures get shorten. So based on the proposed work a dataset modal is generate where each node act as a data row. Pruning of node is depending on two condition first is node have no conditional functional dependency rules. Second is descendent node of the node have no conditional functional dependency rules.

In [15] The discovery of functional dependencies from relations is an important analysis technique. We present TANE, a proficient algorithm for finding functional dependencies from larger databases. TANE is based on partitioning the sets of rows with respect to their attribute values which makes testing the validity of functional dependency fast even for big databases. The results have shown that the algorithm is faster in use. It is observed that for benchmark databases the running times have improved.

In [16] original data is distributed among multiple parties. Here data is horizontally and vertically distribute by utilizing the random tree distribution with homomorpic schema distribution. So all party agree with schema of distributed tree. Here problem of building time is high with increase in number of attributes of the entity. Then data loss is next issue in this paper as schema construction is random so classification accuracy is less.

Yka Huhtala et. al. in [5], has proposed a work that generate conditional functional dependency and approximation rules by utilization of partitions. So by dividing the large dataset in to some partitions generation or searching of functional rules get easy and accurate.
Hong Yao [7] has developed an algorithm named as FDMine (Functional Dependency Mining). Here FD-Mine develops rules by utilizing the functional dependency properties in theory which reduce dataset size for searching as well as filter some of the unwanted or unfruitful rules. It has also proved in the work that pruning of rules not lead to loss of information in the work. Here whole work experiment is done on IS UCI datasets. Here pruning of rules are more as compare to previous works while evaluating results get improved.

## III. PROPOSED WORK

Whole work is a combination of two steps where first include site creation while second include distribution of columns on various sites. While transferring whole row emcryption was performed on the them to save on the sites. Explanation of whole work is shown in fig. 1.

**Pre-Processing**

Pre-Processing: Here dataset is divide into two types of data first is numeric while second is textual. As numeric data is directly distribute on the sites while relation was first developed on the textual data for distribution. Here data need to be read as per the algorithm such as the arrangement of the data in form of matrix is required.

**Association Rules**

| C1 | C2 | C3 | C4 | C5 | C6 |
|---|---|---|---|---|---|
| A1 | B1 | D1 | E1 | F1 | G1 |
| A2 | B2 | D2 | E1 | F2 | G2 |
| A3 | B2 | D3 | E1 | F3 | G3 |

Table 1. Assumed Data for distribution.

In order to hide the information from the dataset one approach is to reduce the support and confidence of the desired item. For finding the item set which is most desired one has to find that the frequent pattern in the dataset. There are many approaches of pattern finding in the dataset which are most frequent one of the most popular is aprior algorithm. In table 1 $C_{1\text{-}n}$ is representing column names.

**Related Column**

Now as per the different frequent rules of the dataset the relation between columns can be evaluate.

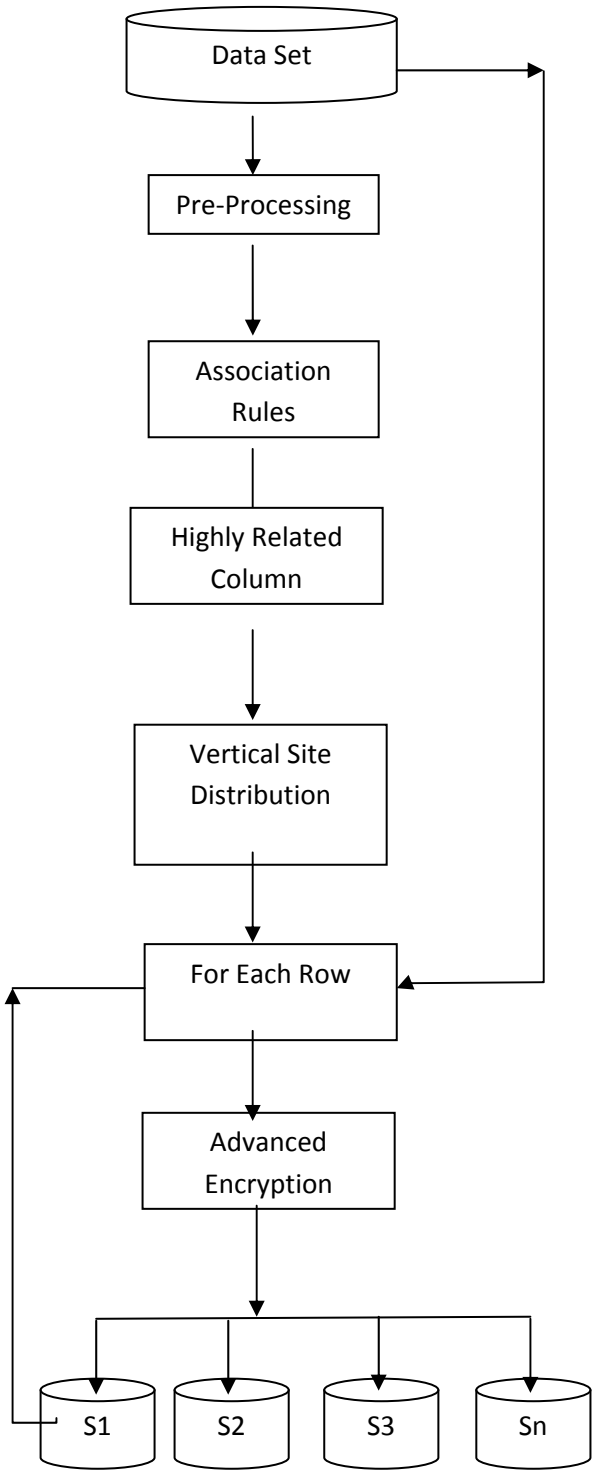


Fig. 1 Proposed Work Block Diagram.

Here all possible pair of columns are prepared then find number of rules between them such as CP={(C1,C2), (C1,C3), (C1,C4), (C1,C5), (C2,C3), (C2,C4)……(C4,C5)}.
This can be understand as column pattern (C1,C2) have $\{I_i \rightarrow I_j\}$ where i=(A1, A2, A3} and j=(B1, B2, B3}. So if total of rules present in the dataset column act as the bond strength between the columns. Sort Highly related Rules in other words pattern having highest number of rules in there group of columns is consider as the strongly related column group.

**Distribute Data on Site**

Here in this step whole columns as per there bonding with other column is distribute on the different site. Here it was try to put strongly column on single site but due to limitations of the site storage, low bonded column is distribute to other site. So depend upon the relation between the columns data partition is done.

| Site | Column | | |
|---|---|---|---|
| **S1** | C3 | C5 | C1 |
| **S2** | C6 | C2 | |
| **S3** | C3 | | |

Table 2. Data Column partition on different sites.

**Insertion of Row**
For each row in the data before transferring data to the selected site it need to be encrypt first. Here Advanced Encryption Algorithm is use.

**AES Encryption**
Now common step for all kind of data is that each data need to be convert into 16 element set of input. Here each input need to be in integer data type. In case of numeric this is ok, but in case of image gray scale will convert pixel values in integer form. While for text unique number is assign for all extracted words.

In this encryption algorithm four stages are perform in each round. While final round consist of three stages only. These steps are common in both encryption as well as decryption algorithm where decryption algorithm is inverse of the encryption one. So round consist of following four stages.

1. Substitute bytes

2. Shift rows

3. Mix Columns

4. Add Round Key

In final round simply all stages remain in same sequence except Mix Columns stage.

**Proposed Algorithm:**

Input: ODS (Original_Dataset)

Output: DDS (Distribute_Dataset)

Step 1. DS ← Pre_Process( ODS )

Step2. FR[n] ← Aprior_Patterns( DS ) // n: number Association patterns. FR: Frequent Rule

Step3 FR[n]←Sort(FR) // Sort as per support value in decreasing order

Step4. Loop 1:n

Step5. Loop 1:n

Step6. If FR[n] ∩ FR[n] // If column match then make in same group

Step7. RR←FR[n] // RR Relative Rule

Step8. Site(s)←RR // s number of sites, site matrix remember the column position to distribute

Step9. Remove Rules having Column in RR from FR list

Step10. Endif

Step11. End Loop

Step12. End Loop

Step13. Loop 1:d // number of rows in the dataset

Step14. Row←AES(ODR[n]) // AES encryption

Step15.DDS←Distribute_on_site(Row, site)

Step16. EndLoop

## IV. EXPERIMENT AND RESULT

**Dataset**
In [9] Sara et. al. has used Adult dataset where it contain different discriminating item set such as country, Gender, Race, 1996. This data set consists of 48,842 records, split into a "train" part with 32,561 records and a "test" part with 16,281 records. The data set has 14 attributes (without class attribute).

**Evaluation Parameters**
**Elapsed Time**
Here total execution time (second) is calculate for the data distribution on different sites. Two type of elapsed time is calculate over here first is Incremental Time and other is Batch time.

Incremental Time: Time required for distribution of single row data on different site is termed as Incremental Time.

Batch Time: Time required for distribution of Batch of data on different site is termed as Batch Time.

**Space Cost**

As data is distributed as per the pattern in the dataset so a perfect pattern have less number of combinations to represent same data. So number of cells required for the storage of data on different sites is termed as Space Cost.

**Results**

Table 3. Comparison of Average Incremental Elapsed time.

| Dataset Size | Proposed Work | Previous Work |
|---|---|---|
| 2000 | 0.004477 | 0.03922 |
| 1500 | 0.004571 | 0.0432801 |
| 500 | 0.004310 | 0.0643501 |

From above table 3 it is obtained that proposed work is better as compare to previous work in [13]. As average incremental elapsed time is less while executing proposed work algorithm. It has seen that by increase in dataset size is remain almost same.

Table 4. Comparison of Average Batch Elapsed time.

| Dataset Size | Proposed Work | Previous Work |
|---|---|---|
| 2000 | 0.012974 | 0.01807 |
| 1500 | 0.031999 | 0.302961 |
| 500 | 0.0301751 | 0.450451 |

From above table 4 it is obtained that proposed work is better as compare to previous work in [13]. As average batch elapsed time is less while executing proposed work algorithm. It has seen that by increase in dataset size also increases. Time requirement is less because of generations of perfect patterns in proposed work as this reduce pattern size.

Table 5 Comparison of Average Space Cost

| Dataset Size | Proposed Work | Previous Work |
|---|---|---|
| 2000 | 7814 | 27459 |
| 1500 | 6947 | 19304 |
| 500 | 6135 | 13165 |

From above table 5 it is obtained that proposed work is better as compare to previous work in [13]. As average space required for proposed work algorithm is comparatively less. Space requirement is less because of generations of perfect patterns in proposed work. It has seen that by increase in dataset size is space cost also increases but not in same ratio as data set increases.

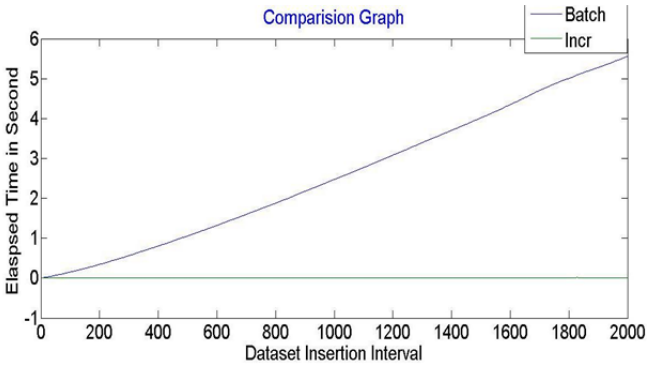


Fig. 2 Proposed work Insertion graph for Batch and Incremental Elapsed Time for input dataset size of 2000.

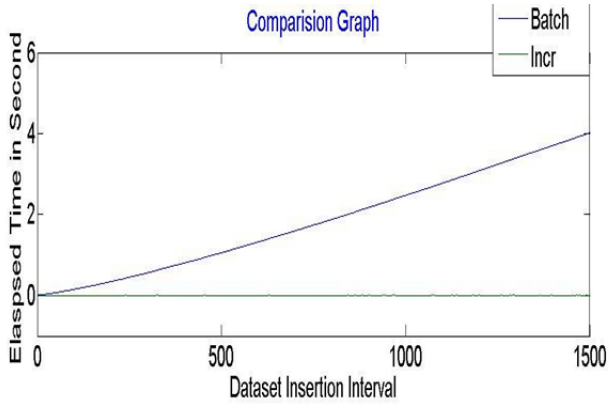


Fig. 3 Proposed work Insertion graph for Batch and Incremental Elapsed Time for input dataset size of 1500.

From above fig. 2 and 3 it is obtained that proposed work incremental time is always constant for all the insertions [13]. In this work batch time increases with insertion of rows.

## V. CONCLUSION

As researchers are working on different field out of which finding an effective vertical patterns is measure issue with this growing digital world. This paper has proposed an data partition algorithm for different sites. Here proper vertical patterns are generate with the help of aprior algorithm. By the use of AES encryption algorithm security of the data at server side get enhance as well. Results shows that proposed work incremental time get reduce by 9%. While batch elapsed time get reduce by 17%. By the use of automatic vertical pattern space cost is also reduce by 5%. As research is never end process so in future one can adopt other pattern generation technique for improving the server performance.

## REFERENCES


[1] Abedjan, Z., Grütze, T., Jentzsch, A., Naumann, F.: Mining and profiling RDF data with ProLOD++. In: Proceedings of the International Conference on Data Engineering (ICDE), pp. 1198–1201(2014).

[2] Rostin, A., Albrecht, O., Bauckmann, J., Naumann, F., Leser,U.: A machine learning approach to foreign key discovery. In: Proceedings of the ACM SIGMOD Workshop on the Web and Databases (WebDB) (2009)

[3] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schonberg, Jakob Zwiener and Felix Naumann, "Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms", Proceedings of VLDB 2015.

[4] Huhtala, Y., Karkkainen, J., Porkka, P., and Toivonen, H., (1999), TANE: An Efficient Algorithm for discovering Functional and Approximate Dependencies, The Computer Journal, V.42, No.20, pp.100-107.

[5] Huhtala, Y., Karkkainen, J., Porkka, P., and Toivonen, Dependencies Using Partitions, IEEE ICDE 1998.

[6] Shyue-liang Wang, Jenn-Shing Tsai and Been-Chian Chien, "Mining Approximate Dependencies Using Partitions on Similarity-relation-based Fuzzy Databases", IEEE International Conference on Systems, Man and Cybernetics(SMC) 1999.

[7] Yao, H., Hamilton, H., and Butz, C., FD_Mine: Discovering Functional dependencies in a Database Using Equivalences, Canada, IEEE ICDM 2002.

[8] Wyss. C., Giannella, C., and Robertson, E. (2001), FastFDs: A Heuristic-Driven, Depth-First Algorithm for Mining Functional Dependencies from Relation Instances, Springer Berlin Heidelberg 2001.

[9] Russell, Stuart J. and Norvig, Peter. Arti cial Intelligence: A ModernApproach. Prentice Hall, 1995.

[10] Mannila, H. (2000), Theoretical Frameworks for Data Mining, ACM SIGKDD Explorations, V.1, No.2, pp.30-32.

[11] Stephane Lopes, Jean-Marc Petit, and Lotfi Lakhal, "Efficient Discovery of Functional Dependencies and Armstrong Relations", Springer 2000.

[12] Heikki Mannila and Kari-Jouko R¨aih¨a. Design by example: An application of Armstrong relations. Journal of Computer and System Sciences, 33(2):126{141, 1986.

[13] Wenfei Fan, Jianzhong Li, Nan Tang, And Wenyuan Y. "Incremental Detection Of Inconsistencies In Distributed Data". Ieee Transactions On Knowledge And Data Engineering, Vol. 26, No. 6, June 2014 1367

14. Thorsten Papenbrock, Felix Naumann ." A Hybrid Approach to Functional Dependency Discovery". SIGMOD'16, June 26-July 01, 2016, San Francisco, CA, USA c 2016 ACM. ISBN 978-1-4503-3531-7/16/06. .

15. Akshay Kulkarni, Sachin Batule, Manoj Kumar Lanke, Adityakumar Gupta. "Functional Dependencies Discovery in RDBMS". International Journal of Advanced Research in Computer Science and Software Engineering Volume 6, Issue 4, April 2016 ISSN: 2277 128X.

[16] Jaideep Vaidya, Senior Member, IEEE, Basit Shafiq, Member, IEEE, Wei Fan, Member, IEEE, Danish Mehmood, And David Lorenzi. "A Random Decision Tree Framework For Privacy-Preserving Data Mining" . IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 11, NO. 5, SEPTEMBER/OCTOBER 2014