International Journal of Advanced Research and Review

www.ijarr.in

AN APPROACH TO INCREASE THE PERFORMANCE OF FUNCTIONAL DEPENDECY FINDING ALGORITHM

W.C.Uduwela[*1,] **P.G.Wijayarathna**[2]

[1]Department of Mathematics and Computer Science, Faculty of Natural Sciences, Open University of Sri Lanka, wasana@ou.ac.lk
[2]Department of Industrial Management, University of Kelaniya, gamaini@kln.ac.lk

ABSTRACT

Relational database model is the most common database model in current information systems to keep the transaction data. The basis of the relational database design process is Functional Dependencies (FDs). Moreover, FDs in the existing database can be used to discover new knowledge and to do data mining; therefore various researches have been carried out to develop algorithms to discover the hidden FDs in the existing data sets. These findings help to database designers too in various ways: to database design verifications, to database management, to reverse engineering, and to query optimization. We could find four popular functional dependencies in the literature. They are TANE, FD_Mine, FastFD, and Dep_Miner. The literature says that the performance of FastFD is better for large number of attributes with lesser number of records, while the performance of the FD_Mine is better for large number of records with lesser number of attributes. We could find an improvement for FD_Mine algorithm, but nothing for FastFD. We suggested an approach to increase the performance of FastFD algorithm using equivalence attributes. The paper concluded that the equivalence sets helps to reduce the time complexity of the FastFD algorithm some extend, by reducing the number of records to be checked.

Keywords - Functional Dependencies, Relational Database, Relational schema, Functional Dependency Algorithms, Equivalence Attributes

INTRODUCTION

Most of the information systems in the commercial applications use the relational database model, which is proposed by Dr Codd (P. S. Dhabe, et al., 2011). Its performance, fully depends on the relational schema (database schema), which is the core of the relational database model. Relational schema describes the categorizations of the data and the relationships among them. The basis of the relational schema is Functional Dependencies (FD). FDs describes the relationships between the attributes of the database relations (tables). It uniquely determines the value of an attribute with values of some other attributes(H. Yk¨a, et al., , 1999). Typically, database designers use the semantic model of the application domain to obtain the functional dependencies (H. Yao, et al , 2002), (H. Yao, H. J. Hamilton, 2008). It can be formally denoted as X → Y in a relational schema R, where X, Y R, is satisfied by r(R), if all pairs of tuples ti, tj r(U), if ti [X] = tj [X] then ti [Y]=tj [Y].

Moreover, Functional Dependencies in the existing database support to discover new knowledge as well as to do data mining (H. Yk¨a, et al., 1999); therefore, researches have been carried out to develop algorithms to discover FDs in the existing data sets in the past few decades. Not only that, but also those findings (algorithms) facilitate to manage databases in various ways: to verify database design (J. Liu, et al., 2012), to database management, to reverse engineering and, to query optimization (H. Yk¨a, et al., 1999). Therefore, database designers, especially non technical people and novel database designers can get the help of these algorithms to make the correction in the existing relational schema as it is difficult to develop the correct relational schema at the beginning.

Among the four popular functional dependencies in the literature (TANE, FD_Mine, FastFD, and Dep_Miner), the performance of FastFD is better for large number of attributes with lesser number of records, while the performance of the FD_Mine is better for large number of records with lesser number of attributes (W.C.Uduwela, P.G.Wijerathna, 2015). In this paper, we suggest an approach to improve the efficiency of the FastFD algorithm using equivalence class.

The paper has organized as follows. Section 2 reviews the existing approaches to discover FD while describing the steps of FastFD algorithm. Section 3 describes the proposed algorithm to improve the FastFD algorithm. Section 4 compares the improved version of FastFD with the old version and concludes the paper.

FASTFD ALGORITHM

From early 1980s, researchers motivate to find efficient solutions to discover functional dependencies automatically from existing datasets (Sood, 2014). These findings can be grouped as either top-down approaches or bottom-up approaches (J. Liu, et al., 2012). Top-down approaches start by generating candidate FDs level-by-level, from short left hand side (lhs) to long lhs. Then it checks the satisfaction of the candidate FDs for satisfaction against the relation or its partitions (J. Liu, et al. 2012). TANE and FD_Mine (J. Liu, et al., 2012), (Sood, 2014) are famous algorithms in this category. On the other hand, the bottom-up approaches, start with comparing tuples to get either agree-sets or difference-sets. (Agree set can be defined for the tuples ti and tj agree on X if ti[X] = tj[X], where ti and tj be tuples and X an attribute set. Difference set can be derived using agree set. Then it generates candidate FDs and check them against the agree-sets or difference-sets. FastFD and Dep_Miner (J. Liu, et al., 2012), (Sood, 2014) are famous algorithms in this category.

The performance of FastFD is better than Dep_Miner, TANE and FD_Mine when the number of attributes getting larger (W.C.Uduwela, P.G.Wijerathna, 2015). Its performance gets poor when the number of records getting larger, but FD_Mines perform better for a large number of records (W.C.Uduwela, P.G.Wijerathna, 2015). Its performance gets poor when the number of attributes getting larger (W.C.Uduwela, P.G.Wijerathna, 2015). A research has been carried out to improve the performance of FD_Mine by minimizing the time complexity and pruning the redundant functional dependencies (H.Nayak, K.Pathak, 2014). We could find any research has not carried out to improve the performance of FastFD algorithm in the literature.

FastFD algorithm mainly contains three stages according to the article (W. Catharine, et al., 2001 published on the algorithm. They are described below.

***Stage1- Finding the difference set*:** Difference set can be found by comparing each tuple with every other tuple in a relation. At the comparison combine the attributes which have the

same value. Then remove the founded set of attributes from the candidate set. Difference set for each attribute can be found by finding the set of attributes which contain that attribute and removing that attribute from the set of attributes. Sometimes, there can be an empty set for the difference set as a result.

Stage2 - *Finding the minimal candidate set*: Minimal set can be found at this stage by removing the superset of the candidate set present in the relevant difference set (out come on the stage 1). A superset is a set that contains all elements of a smaller set. If B is a subset of A, then A is a superset of B.

Stage3 - *Finding the minimum cover of the difference set*. Every different set of the candidate set which does not contain attribute A is a potential candidate for a minimum cover of A (Assume A is an attribute in the candidate set). Consider a search tree representing a simple, "brute-force" method which generates the subsets of candidate set not containing A in a depth-first, left-to-right fashion. The minimal cover of each of the attribute gives us the functional dependencies.

PROPOSED IMPROVEMENT FOR FASTFD ALGORITHM

According the algorithm the time complexity of FastFD (performance) mainly depends on the number of records of the dataset (W. Catharine, et al., 2001). The main objective of the proposed enhancement for FastFD is to find all functional dependencies from the given dataset while reducing the processing time by pruning equivalence attributes, if there is. It reduces the number of records to be checked for FastFD algorithm. The approach modifies FastFD algorithm by introducing new steps to find and remove the equivalence attributes without losing any useful information.

New steps which are introduced for FastFD algorithm to find the equivalence attributes.

- First, read the data set and its attributes (A1,A2….Am).
- Then, compute the partition for each attribute, i.e. partition of attribute A can be denoted as P_A = {{t1, t2, t3, t4, t7}, {t5, t6}}. The values of tuples t1, t2, t3, t4, and t7 on attribute A are all the same, they are assigned to the same group. Likewise, as the values of t5 and t6 are the same, they are assigned into another group. The cardinality of the attribute, which is the number of groups in the partition is two for the attribute A. It can be denoted as $|\pi A| = 2$.
- Then compute the equivalence attribute. Combine the selected attribute with the rest of attributes one by one and compute the partitions. Assume there are two attributes as A and B, and $|\pi A|=|\pi AB|$ as well as $|\pi B|=|\pi AB|$. Then we can conclude that A and B are equivalence attributes.
- Then remove one attribute from the equivalence attributes (either A or B).

Example: Take database with A, B, C and D attributes shown in Table I.i.e. R= (A, B, C, D)

- Partition of attribute A, B, C and D are PA = {(1,2,3,5,7), (4,6)}, PB = {(1,2,3,5,7), (4,6)}, PC = {(1), (2), (3,4), (5), (6,7)} and PD = {(1,7), (2), (3), (4), (5), (6)}
- Compute partition of PAB = {(1,2,3,5,7), (4,6)}, PAC = {(1), (2), (3), (4), (5), (6), (7)} and PAD = {(1,7), (2), (3), (4), (5), (6)} to find the equivalence attributes for attribute A. Since $|\pi A|=|\pi AB|$. Likewise, we can compute $|\pi B| = |\pi AB|$; therefore attribute A is equivalent to attribute B.
- Remove either A or B from the candidate set. If we remove attribute B, then the candidate set is reduced to = (A, C, D)
- Then apply the FastFD algorithm for the reduced data set.

Table-I Sample Data Table

Tuples	A	B	C	D
1	1	7	1	1
2	1	7	2	3
3	1	7	5	4
4	0	4	5	7
5	1	7	7	5
6	0	4	0	2
7	1	7	0	1

RESULT AND CONCLUSION

Both FastFD algorithm and its improved version were implemented using C# for the comparison of their performance. Algorithms were tested on few dataset using a personal computer installed with Windows 7 professional operating system, core i7 processor, and 8GB RAM. Data sets with equivalence attributes, a large number of attributes, and large number of record sets were used for the analysis. Table II illustrates the summary of the result.

Table-2 Comparison Of Fastfd Algorithm With Its Improvements

|r| Number of Raws, |R| Number of Attributes, |F| Number of Functional Dependencies Generated, |e| Number of equivalence set						
|r|	**|R|**	**|F|**	**|e|**	**Time Complexity of FastFD (seconds)**	**Time Complexity of FastFD improved (seconds)**	
					To search FD	*To search Equivalence attributes*
217	05	08	01	35.21	28.19	0.0087
217	15	311	01	366.64	288.45	0.024
50	20	3055	01	228.98	169.61	0.019
50	26	3159	01	537.84	531.73	0.025

The analysis shows that the time complexity of the improved version of FastFD algorithm is better than the time complexity of existing FastFD algorithm, though it takes time to find the equivalence attributes (It takes scanty time to discover equivalence attributes compared to the time took to find the functional dependencies). Better performance can be gained, even for the large number of records, if there are lesser number of attributes with lesser number of functional dependencies. When the number of attributes and the number of functional dependencies getting larger performance of the improved version of FsatFD also getting reduce, even for the lesser number of records. Therefore, we can conclude equivalence sets helps to reduce the time complexity of the FastFD algorithm some extend, by reducing the number of records to be checked.

REFERENCES

H. Yao, H. J Hamilton, and C. J. Butz. . (2002). Fd_Mine: discovering functional dependencies in a database using equivalences. *IEEE International Conference* (pp. 729–732). ICDM2003. Proceedings.

H. Yao, H. J. Hamilton. (2008). Mining functional dependencies from data. *Data Mining and Knowledge Discovery*, 197-219.

H. Yk¨a, J. K¨arkk¨ainen, P. Porkka, and H. Toivonen, . (1999). Tane: An efficient algorithm for discovering functional and approximate dependencies . *The computer journal*, 100-111.

H.Nayak, K.Pathak. (2014). Fast and Efficient Approach for Discovering Functional Dependency from Database. *International Journal of Advanced Research in Computer Science and Software Engineering*, 1099-1105.

J. Liu, J. Li, C. Liu, Y. Chen. (2012). Discover Dependencies from Data—A Review. *Knowledge and Data Engineering, IEEE Transactions*, 251-264.

P. S. Dhabe, S. V. Deshmukh, Y. V. Dongare. (2011). RDBNorma: - A semi-automated tool for relational database schema normalization up to third normal form. *Journal of Database Management Systems (IJDMS)*, 133-154.

Sood, K. (2014). *"Comparison of Functiona Ldependency Extraction Methods and an Application of Depth First Search.* M.Sc. Disseration, Department of Computer and Information Science and the Graduate School, University of Oregon.

W. Catharine, C. Giannella, E. Robertson. (2001). FastFds: A heuristic-driven, depth-first Algorithm for mining functional dependencies from relation instances extended abstract. *In Data Warehousing and Knowledge Discovery*, 101–110.

W.C.Uduwela, P.G.Wijerathna. (2015). Comparative Study of Functional Dependency Generation Algorithms. *International Journal of Advanced Information Science and Technology*, 11-14.