Volume 3, Issue 12, December 2018 ISSN (online): 2456-0006

International Journal of Science Technology Management and Research

Available online at: www.ijstmr.com

Discovering Valuable Relationships in RDBMS Using Complements of Maximal Set

Preeti Katiyar
M Tech (C.S.E.) 4 semesters
Lord Krishna College of Technology
Indore M.P. India
Preetikatiyar5@gmail.com

Vijay Kumar Verma
Asst. Professor C.S.E department
Lord Krishna College of Technology
Indore M.P. India
vijayvermaonline@gmail.com

Abstract: ***Data normalization is a common mechanism employed to support database designers to ensure the correctness of their design. Normalization transforms unstructured relation into separate relations, called normalized database. There are many different levels of normalization depending on the purpose of database designer. Most database applications are designed to be either in the third normal forms in which their dependency relations are sufficient for most organizational requirements. Dependency discovery has attracted a lot of research interests from the communities of database design, machine learning and knowledge discovery since early 1980s. Three typical types of dependencies are often involved in the discovery, functional dependencies (FDs), Inclusion dependencies (INDs) and Conditional Functional Dependency (CFD). FDs represent value consistencies between two sets of attributes while INDs represent value reference relationships between two sets of attributes. In recent years, the discovery of conditional functional dependencies (CFDs) has also seen some work. The aim of dependency discovery is to find important dependencies holding on the data of the database. In this paper we proposed new and efficient approaches which identify meaningful relation between attributes.***

Keywords: Normalization, databases, relations, knowledge, discovery

I. INTRODUCTION

The process of data mining mainly includes association rules, classification and prediction, and clustering. Data dependencies play very important roles in database design, data quality management, and knowledge representation. Functional dependency is a kind of data dependencies. Now-a-days there is a fast growing amount of data that are collected and stored in large databases. As a result, the databases may contain redundant or inconsistent data. Dependencies are very important in the case of database design, data quality management and knowledge representation. Dependencies in the case of database design, data quality management and knowledge representation are extracted from the application requirements and are used in the database normalization and implemented in the designed database to warranty the data quality. In case of knowledge discovery, dependencies are extracted from the existing data of the database. The extraction process is known as Dependency Discovery. The aim of dependency discovery is to find all dependencies which are satisfied by existing data. Conceptual schema and logical designs are two important steps regarding correctness and integrity of the database model. Data normalization is a common mechanism employed to support database designers to ensure the correctness of their design. Normalization transforms unstructured relation into separate relations, called normalized database. The main purpose of this separation is to eliminate redundant data and reduce data anomaly (i.e., data inconsistency as a result of insert, update, and delete operations). There are many different levels of normalization depending on the purpose of database designer. Most database applications are designed to be either in the third normal forms in which their dependency relations are sufficient for most organizational requirements. In recent years, the discovery of conditional functional dependencies (CFDs) has also seen some work. The aim of dependency discovery is to find important dependencies holding on the data of the database. These discovered dependencies represent domain knowledge and can be used to verify database design and assess data quality. Data dependencies play very important roles in database design, data quality management, and knowledge representation. Functional dependency is a kind of data dependencies. Now-a-days there is a fast growing amount of data that are collected and stored in large databases. As a result, the databases may contain redundant or inconsistent data. Dependencies are very important in the case of database design, data quality management and knowledge representation

II. TYPES DATA DEPENDENCY

Dependency discovery has attracted a lot of research interests from the communities of database design; machine learning and knowledge discovery. Three typical types of dependencies are often involved in the discovery, functional dependencies (FDs), conditional functional dependence and inclusion dependencies (INDs). The aim of dependency discovery is to find important dependencies holding on the data of the database. These discovered dependencies represent domain knowledge and can be used to verify database design and assess data quality. In recent years, the demand for improved data quality in databases has been increasing and a lot of research effort in this area has been given to dependency discovery.

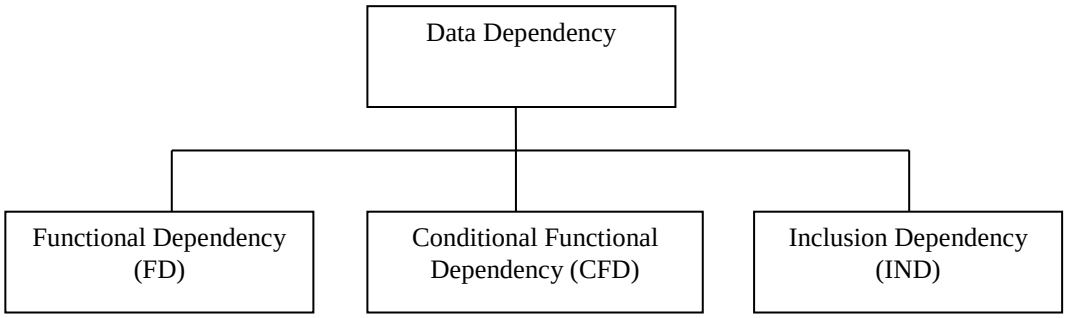


Figure 1- Type of dependencies

Functional dependency

Let R = {A_1,……,A_m} be a database table schema and r be a set of tuples from dom(A_1)×... …×dom (A_m) where dom(A) represents the domain of attribute A. The projection of a tuple t of r to a subset X⊆R is denoted by t[X]. Similarly r[X] represents the projection of r to X. The number of tuples in the projection, called the cardinality, is denoted by |r [X] |. For simplicity, we often omit braces when the context is clear. If X and Y are sets and A is an attribute, XA means X ∪{A} and XY means X ∪ Y.

A functional dependency is a statement X →Y requiring that X functionally determines Y where X; Y⊆ R. The dependency is satisfied by a database instance r if for any two tuples t_1; t_2 ∈ r, if t_1 [X] =t_2 [X] then t_1[Y] = t_2[Y]. X is called the left-hand side (lhs) or the determinant and Y is called the right-hand side (rhs) or the dependent.

Given a relation 'R', attribute 'Y' of 'R' is functional dependant on attribute 'X' of 'R' if- each 'X' value of "r' is associated with precisely one value of 'Y' in 'R' " .

A functional dependency is a statement X →Y requiring that X functionally determines Y. For example city → state i.e. the state value depends on city value

Conditional functional dependencies

Conditional functional dependencies (CFDs) are the extension of functional dependencies (FDs) by supporting patterns of semantically related constants.CFD extends FD by incorporating a pattern tuple of semantically related data values.

For each attribute A in a schema R we denote its associated domain as Dom(A) which is either infinite or finite . A CFD f on R is a pair(R: X→Y, T_P),

Where

- X and Y are sets of attribute in attr (R).
- X→Y is a standard FD referred to as the FD embedded Φ.
- T_p is a tableau with attribute in X and Y referred to as the pattern of Φ where each A in X→Y and each tuple t Tp t(A) is either a constant 'a' in dom(A) or an unmanned variable that draw values from Dom(A)

Inclusion dependency

An inclusion dependency (IND) over a database schema R is a statement of the form R_1[X] ⊆ R_2[Y] where R_1, R_2 ∈ R and X, Y are sequences of attributes such that X ⊆ R_1, Y ⊆R_2 and |X| = |Y| . A unary inclusion dependency (UIND) is an IND such that |X| =|Y| =1. An IND R_1[X] R_2[Y] is of size i if |X| =|Y| =i.

Let d be a database over a database schema R_ where r_1, r_2 ∈ d are relations over relation schemas R_1, R_2 ∈ R. An inclusion dependency R_1[X] ⊆ R_2[Y] is satisfied (or holds) in a database d over R. denoted by d |= R_1[X] ⊆ R_2[Y] if ∀t_1 ∈ r_1, t_2 ∈ r_2 such that t_1[X]= t_2[Y] (Equivalently d |= R_1[X]⊆ R_2[Y] whenever .

III. TYPES OF FD TECHNIQUES

FD discovery methods follow either top-down or bottom up approach. Top down method first generates candidate FDs and form an attribute lattice and the test there satisfaction. Then at lower level the satisfied FDs are used to prune candidate FDs to reduce the search space. Bottom up method compares the tuples of the relation to find agree-sets or different-sets. These sets are then used to derive FDs satisfied by the relation. Various algorithms for Functional dependency developed under each method

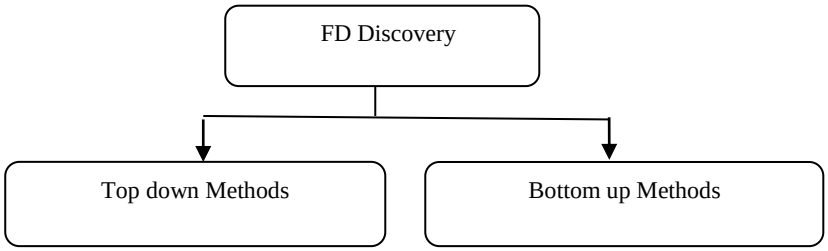


Figure 2 FD discovery techniques

Top-down methods start with candidate FD generation. These methods generate candidate FDs following an attribute lattice, test their satisfaction, and then use the satisfied FDs to prune candidate FDs at lower levels of the lattice to reduce the search space.

1. ***Candidate FDs and pruning*:-**We first present candidate FD generation and pruning. Candidate FDs (canFDs) are FD expressions that are syntactically possible in a relation schema. Their satisfaction against the relation instance has not been tested.Given schema R= {A_1,……..,A_m}, can FDs are calculated using all possible attribute combinations of R as lhs. As we are only interested in minimal FDs with single attribute on the rhs, the number of attributes for the lhs of a canFD contains at most (m-1) attributes. For example, the canFDs with zero attributes in their lhs are $\Phi \rightarrow A_1$…….. $\Phi \rightarrow A_m$. The canFDs with one attribute in their lhs can be $A_1 \rightarrow A_2$, $A_1 \rightarrow A_3$, $A_m \rightarrow A_{m-1}$, etc. The canFDs with two attributes in lhs can be $A_1A_2 \rightarrow A_3$, $A_1A_2 \rightarrow A_4$, $A_{m-1}A_m \rightarrow A_1$, etc. The canFDs with (m - 1) attributes are $A_1 \ldots A_{m-1} \rightarrow A_m, \ldots, A_2 \ldots A_m \rightarrow A_1$. The lhs can be shown graphically in an attribute lattice

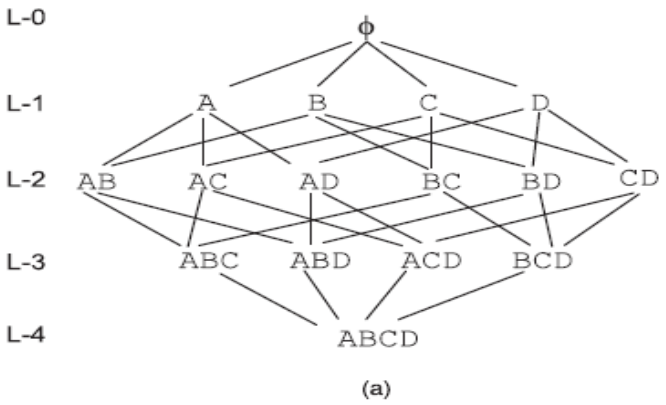


Figure 3 search space for candidate generation

Because the number of canFDs is exponential to the number of attributes, pruning implied FDs from the lattice becomes important to many of the proposed methods. FD pruning is to remove the canFDs (edges) in the lattice implied by the discovered FDs so that we do not check them against r.

2. ***The partition method:-*** The partition semantics of relations is proposed in some research paper . The semantics is used to check the satisfaction of candidate FDs. We call the methods using the partition semantics the partition methods. The algorithms implementing the method include TANE and FD Mine. Both TANE and FD Mine use the essential pruning rules , but FD Mine uses symmetric FDs too. Although we use the term "partition method" to mean a category of algorithms, the concept "partition" is also used in the algorithms of other categories, such as the free-set method and the bottom-up methods, to optimize performance.Given a relation r on R and a set X of attributes in R, the partition of r by X, denoted by PX, is a set of nonempty disjoint subsets and each subset contains the identifiers of all tuples in r having the same X value. Each subset of a partition is called an equivalent class. A stripped partition is a partition where subsets containing only one tuple identifier are removed. To store partitions, the partition methods need to allocate space to each node at two levels: Level (i- 1) and Level (i). Thus, the algorithm has extra space cost in comparison to the nested loop approach.
3. ***Free Set:*** **-** A free set is a minimal set X of attributes in schema R such that for any subset Y of X, $|r[Y]| < |r[X]|$. Thus, every single attribute is a free set because they do not have a subset. If X is a free set, $A \in (R- X)$ and $|X| < |XA|$ and $|A| < |XA|$, then XA is another free set. The lhs of any minimal FD is necessarily a free set. The free set of relation r, denoted by Fr(r), is a set of all free sets on r. Anon-free-set is a set whose cardinality equals to the cardinality of some of its subsets. A superset of a non-free-set is also a non-free-set. To calculate the FDs supported by r, two more concepts are needed: attribute closure X^+ and quasi attribute closure X^o. The closure of set X is calculated using cardinality as $X^+ = X+\{A|A \in (R - X) \wedge |r[X]|= |r[XA]|\}$. That is, X^+ contains attribute A on a node at the next level if $X \rightarrow A$. The quasiclosure of X is $X^o= X+(X- A1)^+ +$……..+ $(X - A_k)^+$ where $X = A_1 \ldots A_k$. In fact X^o contains the attributes on all the parent nodes of X and all the dependent nodes of the parent nodes .

Bottom-Up Methods

Different from the top-down methods above, bottom-up methods compare the tuples of the relation to find agree-sets .or difference-sets. These sets are then used to derive FDs satisfied by the relation. The feature of these methods is that they do not check candidate FDs against the relation for satisfaction, but check candidate FDs against the computed agree-sets or difference-sets.

1. **Negative Cover: -** Negative Cover is a cover of all FDs violated by the relation. Negative cover is calculated using agree-sets of tuples of the relation. The agree-set of two tuples t_1 and t_2, denoted by $ag(t_1; t_2)$ is the maximal set X of attributes such that $t_1[X]$ $t_2[X]$. The set of all agree-sets on relation r is denoted by ag(r). Agree-sets can be calculated from attribute partitions. Given the partition P_A of attribute A and two tuples t_1 and t_2, A is in $ag(t_1, t_2)$ if there exists a subset c in P_A such that t_1 and t_2 are contained in c. For efficiency reasons, stripped partitions are often used in the calculation. The property of agree-sets is that if $ag(t_1; t_2)$= X, then for any $A \in (R - X)$ $(t_1[A] \neq t_2[A])$. In other words, $X \rightarrow A$ is violated by t_1 and t_2. This is the basic principle of the negative cover approach. The complexity of the negative cover approach is exponential to the number of attributes in R as in the worst case. Different variations of the negative cover approach are proposed in various research paper.

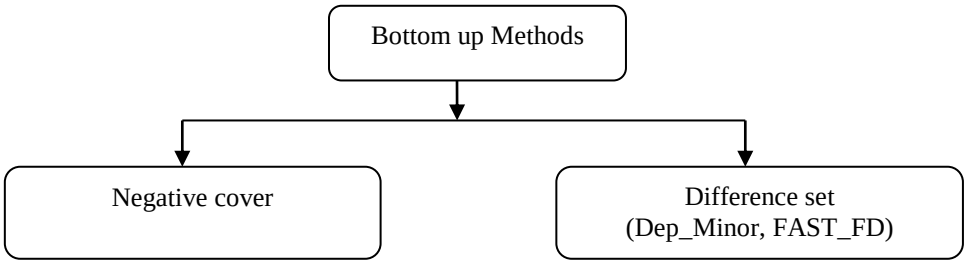


Figure 4 Types of bottom up methods

2. **Difference-sets.** The term difference-set is same as necessary-set and the complement of max-set [22]. The method difference-set employs an opposite thinking from negative cover. The different-set of an attribute A, denoted by dif(A), is a set containing subsets of attributes such that whenever attribute A has different values on two tuples, a subset in dif(A) has different values on the same two tuples too . Once dif(A) is obtained, the lhs of satisfied FDs should contain an attribute from each subset of dif(A). Although the principle of deriving the lhs of satisfied FDs is simple, the search space of satisfied FD calculation is exponential to the number of all attributes in dif(A).

IV. LITERATURE REVIEW

In 2010 Jixue Li u Jiuyong Li "proposed "Discover Dependencies from Data - A Review". They reviews methods for functional dependency, conditional functional dependency, approximate functional dependency and inclusion dependency discovery in relational databases and a method for discovering XML functional dependencies. They also reviewed the methods for discovering FDs, AFDs, CFDs, and INDs in relational databases and XFDs in XML databases. They show that the dependency discovery problem has an exponential search space to the number of attributes involved in the data[1].

In 2011 Wenfei Fan , Floris Geerts , Jianzhong Li , Ming Xiong proposed "Discovering Conditional Functional Dependencies". They investigate the discovery of conditional functional dependencies (CFDs). They show that CFDs are a recent extension of functional dependencies (FDs) by supporting patterns of semantically related constants, and can be used as rules for cleaning relational data. However, finding quality CFDs is an expensive process that involves intensive manual effort. To effectively identify data cleaning rules, we develop techniques for discovering CFDs from relations. Already hard for traditional FDs, the discovery problem is more difficult for CFDs. They provide three methods for CFD discovery. The first, referred to as CFD Miner, is based on techniques for mining closed item sets, and is used to discover constant CFDs, namely, CFDs with constant patterns only. Constant CFDs are particularly important for object identification, which is essential to data cleaning and data integration. The other two algorithms are developed for discovering general CFDs. One algorithm, referred to as CTANE, is a level wise algorithm that extends TANE, a well-known algorithm for mining FDs. The other, referred to as Fast CFD, is based on the depth-first approach used in Fast FD, a method for discovering FDs. It leverages closed-item set mining to reduce the search space. As verified by our experimental study, CFD Miner efficiently discovers constant CFDs. For general CFDs, CTANE works well when a given relation is large, but it does not scale well with the arity of the relation[2].

In 2012 Thierno Diallo JeanMarc Petit proposed "Discovering Editing Rules for Data Cleaning" They proposed a new semantics of ERS taking advantage of both source and master data. The problem turns out to be strongly related to the discovery of both CFD and one-to- one correspondence between sources and target attributes. They proposed efficient techniques to address the discovery problem of ERS and heuristics to clean data. They implemented and evaluated our techniques on real- life databases. They show that the feasibility, the scalability and the robustness of our proposal. They proposed new semantics of Editing Rules in order to be able to infer them from existing source database and a corresponding master database. Based on this new semantics, they proposed a mining process in 3 steps: 1 Eliciting one-to-one correspondences between attributes of a source relation and attributes of the master database. 2 Mining CFDs in the master relations 3 Building Editing Rules[3].

In 2012 Jixue Liu, Jiuyong Li, Chengfei Liu, and Yongfeng Chen proposed "Discover Dependencies from Data—A Review" . They reviewed the methods for functional dependency, conditional functional dependency, approximate functional dependency, and inclusion dependency discovery in relational databases and a method for discovering XML functional dependencies. They also reviewed the methods for discovering FDs, AFDs, CFDs, and INDs in relational databases and XFDs in XML databases. The dependency discovery problem has an exponential search space to the number of attributes involved in the data. Fortunately, most data contain FDs and INDs with single or a few attributes on the lhs[4].

In 2013 Sujoy Dutta & Dr. Laxman Sahoo proposed "Mining Full Functional Dependency to Answer Null Queries and Reduce Imprecise Information Based on Fuzzy Object Oriented Databases". They proposed the concept of fuzzy functional dependency is extended to full functional dependency on similarity based fuzzy object oriented data model. From this functional dependency, we shall be able to reach full functional dependency the major objective of this paper is to reduce imprecise information over databases. Different degrees of similarity to the elements in each domain are introduced and compared with similarity relation for the representation of "fuzziness" in the fuzzy object-oriented data model based on fuzzy similarity database model. An attempt has been made to answer null queries using analogical reasoning in basis of full functional dependency on similarity based fuzzy object-oriented data model. An algorithm to find out full functional dependencies from semantic relations has been provided. The approach is based on considering partitions of the relation and deriving valid dependencies from the partitions. The algorithm searches for dependencies in a level wise

manner. They showed how the search space can be pruned effectively, and how the partitions and dependencies can be computed efficiently. Partition and equivalence classes are also used to find out the full functional dependency easily and efficiently[5].

In 2014 P.Andrew, J. Anish kumar and S.Charany proposed "Investigations on Methods Developed for Effective Discovery of Functional Dependencies" . They give details about various methods to discover functional dependencies from data. Effective pruning for the discovery of conditional functional dependencies is discussed in detail. Di conditional Functional Dependencies and Fast FDs a heuristic-driven, Depth-first algorithm for mining FD from relation instances are elaborated. Privacy preserving publishing micro data with Full Functional Dependencies and Conditional functional dependencies for capturing data inconsistencies are examined. The approximation measures for functional dependencies and the complexity of inferring functional dependencies are also observed. Compression – Based Evaluation of partial determinations is portrayed. This survey would promote a lot of research in the area of mining functional dependencies from data. They also give detailed about various methods to discover functional dependencies from data. Effective pruning for the discovery of conditional functional dependencies is discussed in detail. Di conditional Functional Dependencies and Fast FDs a heuristic-driven, Depth-first algorithm for mining FD from relation instances are elaborated. Privacy preserving publishing micro data with Full Functional Dependencies and Conditional functional dependencies for capturing data inconsistencies are examined[6].

In 2015 R.Santhya1, S. Latha proposed "Further Investigations on Strategies Developed for Efficient Discovery of Matching Dependencies". They give details about various methods prevailing in literature for efficient discovery of matching dependencies. The concept of matching dependencies (MDs) has recently been proposed for specifying matching rules for object identification. Similar to the functional dependencies with conditions, MDs can also be applied to various data quality applications such as detecting the violations of integrity constraints. The problem of discovering similarity constraints for matching dependencies from a given database instance is taken into consideration. This survey would promote a lot of research in the area of information mining. This paper detailed about various methods prevailing in literature for efficient discovery of matching dependencies. The concept of matching dependencies (MDs) has recently been proposed for specifying matching rules for object identification. Similar to the functional dependencies (with conditions), MDs can also be applied to various data quality applications such as detecting the violations of integrity constraints. The problem of discovering similarity constraints for matching dependencies from a given database instance is taken into consideration. This survey would promote a lot of research in the area of information mining [7].

In 2016 Akshay Kulkarni proposed "Functional Dependencies Discovery in RDBMS". They presented 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. Functional dependencies are important metadata which can used to gain knowledge. Discovery of functional dependency can help in removing inconsistent and redundant data we propose a algorithm, TANE, for the discovery of functional and approximate dependencies from relations. The approach is based on deriving dependencies from partitions and searching for it in level-wise manner [8] .

In 2017 A.B Amure et al proposed Functional Dependency: Design and Implementation of a Minimal Cover Algorithm. They focused on solving redundancy problem of Functional Dependencies in a relational schema using the closure of Functional dependency and minimal cover algorithm. An algorithm was designed using the three Armstrong's axioms which are reflexivity, augmentation and transitivity and implemented to generate closure of functional dependencies (FDs) called F+. Other rules used are decomposition, union and pseudo-transitivity, which are products of the three basic axioms. The attribute closure algorithm will also generate the attribute closure X+ under any given set of FDs. Minimal cover (G+) algorithm is then designed and implemented to eliminate redundant FDs. This save storage and optimize the database by eliminating unneeded attributes in FDs[9].

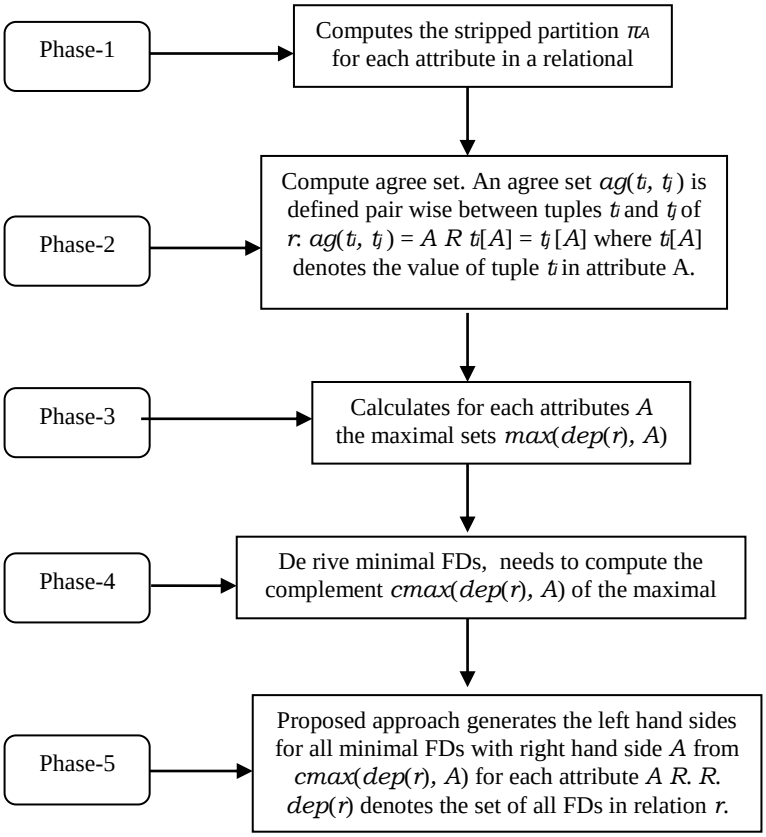
In 2017 Thorsten Papenbrock et al. proposed classify the algorithms into three different categories, explaining their commonalities. They describe all algorithms with their main ideas. The descriptions provide additional details where the original papers were ambiguous or incomplete. Our evaluation of careful re-implementations of all algorithms spans a broad test space including synthetic and real-world data. We show that all functional dependency algorithms optimize for certain data characteristics and provide hints on when to choose which algorithm. In summary, however, all current approaches scale surprisingly poorly, showing potential for future research. Shown that FD discovery is still an open research _eld: None of the state-of-theart algorithms in our experiments scales to datasets with hundreds of columns or millions of rows. Given a dataset with 100 columns and 1 million rows, which is a reasonable size for a table, DFD , Dep-Miner, FastFD will starve in runtime, whereas Tane, Fun, and FD Mine will use up any available memory. This observation indicates potential for future research [10].

V. PROBLEM STATEMENT

The problem addressed in this proposed work is to find all functional dependencies among attributes in a database relation. Specifically, we want to improve on previous proposed methods for this problem. Early methods for discovering of FDs were based on repeatedly sorting and comparing tuples to determine whether or not these tuples meet the FD definition
Our work addresses two main problems.
(1)How we can other information from discovered FDs are used to prune more candidates than previous approaches.
(2) How efficiently this pruning can be done so that the overall efficiency of the algorithm is improved.

VI. PROPOSED APPROACH

The proposed algorithm infers all minimal functional dependencies from sets of attributes that have same values in certain tuples . These sets are called *agree sets* and their inverse *difference sets*. On an abstract level, Dep-Miner can be divided into five phases. In Phase 1, computes the stripped partition π_A for each attribute in a relational instance r. The π_A are then used in Phase 2 to build the agree sets $ag(r)$. Phase 3 transforms the agree sets into maximal sets, attributes that have no superset with same values in two records of r. Phase 4 inverts the agree sets into complement sets. From the complement sets, the algorithm then calculates all minimal FDs in Phase 5. Thereby it uses a level-wise search on top of the complement sets.



VII. RESULT ANALYSIS

We evaluate the performance of our algorithm and compare Dep_Miner (Depth Miner) algorithm. The experiments were performed on i5 processor (2.5GHz Intel Processor with 4M cache memory), 2GB main memory and 400 GB secondary memory , and running on Windows 2008. The algorithms are implemented in using C# Dot Framework Net language version 4.0.1. Synthetic datasets are used to evaluate the performance of the algorithms We have used to dataset first dataset is employ table which contain five attribute Emp_No, Dep_No ,Year, Dep_Name, and Mgr _No and 150 tuples . Second data base is Binary relation which has five attribute A, B, C, D and E. In second dataset each attribute has a binary value mean present(1) and absent second data set has 100 tuples .We use SQL Server R2(2008) to store database. For comparing the performance of the proposed algorithms with Dep_Miner. We have used number of attributes and dependencies at different level.

Table 6 Level and generated dependencies

Level number	Depth Minor	Proposed approach
1	5	5
2	10	6
total	15	11

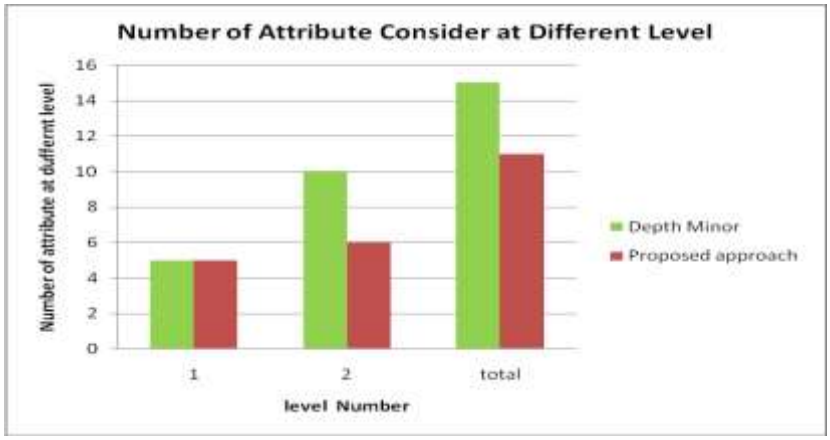


Figure 5- Number of dependencies at different level

CONCLUSION

We have suggested a new approach to reduce extra attribute to find out data dependency form a given relational database algorithm. Discover FDs which utilizes the mathematical prosperities of the database. In the proposed work we us the concepts of concur set, agree set and closure properties. The aim of proposed algorithm is generate meaning full dependencies and optimize the time and memory requirements. We compare proposed approach with Dep_minor algorithm. From the example and implementation it is clear that proposed approach works efficiently as compared to the Dep_minor and has a better performance. The main factor that we consider the proposed works includes, Number of dependencies at different level, Number of attributes process at different level,Execution time need by algorithms.

REFERENCE

1. Jixue Liu, Jiuyong Li " Discover Dependencies from Data - A Review" School of Computer and Info. Sci., University of South Australia {jixue.liu, jiuyong.li}@unisa.edu.au 2 Faculty of ICT, Swinburne University of Technology cliu@swin.edu.au 2010.
2. Wenfei Fan , Floris Geerts , Jianzhong Li & Ming Xiong" Discovering Conditional Functional Dependencies TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL.23, NO. 5, May 2011.
3. Thierno Diallo & JeanMarc Petit "Discovering Editing Rules For Data Cleaning" This article was presented at the 9th International Workshop on Quality in Databases (QDB) 2012.
4. Jixue Liu, Jiuyong Li, Chengfei Liu, and Yongfeng Chen "Discover Dependencies from Data—A Review" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 24, NO. 2, FEBRUARY 2012.
5. Sujoy Dutta "Mining Full Functional Dependency to Answer Null Queries and Reduce Imprecise Information Based on Fuzzy Object Oriented Databases" International Journal of Computer Science & Engineering Technology (IJCSET) ISSN: ISSN: 2229-3345 Vol. 4 No. 03 Mar 2013.
6. P.Andrew, J.Anishkumar & S.Balamurugan "Investigations on Methods Developed for Effective Discovery of Functional Dependencies International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization) Vol. 3, Issue 2, February 2015.
7. Thorsten Papenbrock & Jens Ehrlich "Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms" International Conference on Very Large Data Bases, August 31st September 4th 2015, Kohala Coast, Hawaii. Proceedings of the VLDB Endowment, Vol. 8, No. 10 Copyright 2015 VLDB Endowment 21508097/ 15/06.
8. Akshay Kulkarni and Sachin Batule Functional Dependencies Discovery in RDBMS Volume 6, Issue 4, April 2016 ISSN: 2277 128X International Journal of Advanced Research in Computer Science and Software Engineering Research Paper Available online at: www.ijarcsse.com
9. A.B Amure , M.A. Amosu, H.O.D. Longe " Functional Dependency: Design and Implementation of a Minimal Cover Algorithm" IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 5, Ver. I (Sep.- Oct. 2017), PP 77-81 www.iosrjournals.org.
10. Thorsten Papenbrock and Jens Ehrlich Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms 41st International Conference on Very Large Data Bases, August 31st - September 4th 2017, Kohala Coast, Hawaii. Proceedings of the VLDB Endowment, Vol. 8, No. 10 Copyright 2017 VLDB Endowment 2150-8097/17/06.