Volume 4, Issue 7, July 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Fast and Efficient Approach for Discovering Functional Dependency from Database

Harshila Nayak
Mahakal Institute of Technology,
Ujjain, India

Kshitij Pathak
Mahakal Institute of Technology,
Ujjain, India

Abstract— Dependency Discovery is most famous research topic since last three decades. The aim of dependency discovery is to discover all dependencies from the existing database. Dependencies which are extracted from database are used for normalization as well as there are implemented in designed database to warranty the data quality. Functional dependency is very important and traditional database analysis technique. A functional dependency is defined as a constraint between two sets of attributes in relation from a database. We present novel algorithm i.e. FD Generation for finding minimal FDs from relational database .The main objective is to reduce data and candidate generation. Proposed research work can reduce both the size of the dataset and the number of FDs to be checked by pruning redundant data and skipping the search for FDs that follow logically from the FDs already discovered .At the end we compare our result with FD_mine algorithm.

Keywords— Functional dependency, inclusion dependency, conditional dependency, equivalence class, dependency discovery.

I. INTRODUCTION

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.

A. Data dependency in relational database

Dependency discovery has attracted a lot of research interests from the communities of database design; machine learning and knowledge discovery[14],[18], [15], [17], [16], [19]. Integrity constraints are also one of the main fundamental concepts in defining semantics in relational databases. Integrity constraints are one of the oldest and most important topics in database research, and they find application in a variety of areas such as database design, data translation, query optimization and data storage.

B. Dependency Discovery Techniques

There are basically three typical types of dependencies discovery techniques:

1) *Functional Dependency (FD):* A functional dependency is defined as an association between two sets of attributes in a relation from a database. Let R be a relation having X and Y are set of attributes such that set of attribute X functionally determine another attribute Y, of relation R,which is represented as $X \rightarrow Y$ if and only if each X value is associated with at most one Y value, here X is the determinant set and Y is the dependent attribute. Thus, given a tuple and the attributes values in X can functionally determine the corresponding value of the Y attribute.

2) *Conditional Functional Dependency(CFD):*Given two subsets X and Y of attributes of R,a conditional FD is a statement (X->Y,S) where S is a pattern tableau on XY .A CFD [1],[7],[8],[9]is satisfied by relation r over R iff for any two tuples t1; t2 € r and for each pattern tuple p in S, if t1[X]= t2[X]= p[X] then t1[Y] = t2[Y]= p[Y].

3) *Inclusion Dependency(IND):*An inclusion dependency [10],[11],[12],[13] R.X < S.Y between two sets of attributes —X of relation schema R, and Y of relation schema S—specifies the constraint that, at any specific time when r is a relation state of R and s a relation state of S, we must have

$\pi X(r(R)) \subseteq \pi Y(s(S))$

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.

II. DISCOVERY OF FUNCTIONAL DEPENDENCY

In this section of paper, we are defining concept of functional dependency and techniques used in FD discovery process.

A. Definition

Let R = {$A_1, \ldots, A_m$} 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 U {A} and XY means X U Y.

A functional dependency [1] 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. Research on FDs was an important aspect of relational database design the Armstrong Axioms, minimal cover, closure and an algorithm for decomposing a schema into third normal form while preserving dependency. The Armstrong axioms can be stated as three rules [20]:

1) *Reflexivity rule*: if Y⊆X, then X →Y ;
2) *Augmentation rule*: if X->Y, then XZ->YZ;
3) *Transitivity rule*: if X ->Y and Y->Z, then X->Y.

These rules can be applied repeatedly to infer all FDs implied by a set of FDs.

B. Techniques of Functional Discovery

We are specifying two techniques for discovery of FD:

1) *Top – Down Technique: The* top-down approach start with generating candidate FDs level-by-level, from short lhs to long lhs, and then check the satisfaction of the candidate FDs for satisfaction against the relation or its partitions. Top-down technique employs two methods for FD discovery: *Partition Method* [4][5] defined as Let 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.TANE [5] and FD_Mine [4] algorithm is based on partition method. *Free set Method* [1] defined as 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 € (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.In short free set method which uses the cardinality of projected relations to test satisfaction. FUN [22] algorithm use the concept of free set.

2) *Bottom - Up Technique:* The bottom-up approach, start with comparing tuples to get agree-sets or difference-sets, then generate candidate FDs and check them against the agree-sets or difference-sets for satisfaction. Bottom-up approach defined following two method for FD discovery: *Negative Cover* is a cover of all FDs violated by the relation and it is computed by using agree- set concept of the tuples of relation[20].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).*Difference- Sets* 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 [21].FastFD[21] algorithm employs difference set method.

III. LITERATURE SURVEY

Research on FDs was an important aspect of relational database design [6] in the 1980s, and led to achievements such as the Armstrong Axioms, minimal cover, closure and an algorithm for decomposing a schema into third normal form while preserving dependency.

In [1] Jixue Liu, Jiuyong Li, Chengfei Liu, and Yongfeng Chen reviewed the methods for discovering FDs, AFDs, CFDs, and INDs in relational databases and XFDs in XML databases. These methods are either Top-down or bottom-up. With FD discovery, the direction of computation starts with FDs having fewer attributes in lhs. The discovered FDs are then used to prune other candidate FDs in the attribute lattice so that the search space of the computation is reduced. The most commonly proposed and cited method in the literature is the partition method and the negative cover method. The partition method is also used in XFD discovery. In discovering AFDs, the sampling method is used to find FDs that are approximately satisfied. With regard to the IND discovery, the direction of computation starts with small INDs too. The

invalid INDs discovered are then used to prune candidate INDs to reduce the complexity of computation. In the area of CFD discovery, although some algorithms for FD discovery can be adapted for CFD discovery purpose, the discovery of an optimal tableau is Incomplete and the discovery of a good tableau seems not an easy task. More simple but effective and efficient algorithms are still needed. Research work associated with these review paper is that work in discovering XML functional dependencies has just started. The only work done on XFD discovery converts XML data into relational data and then applies the partition method to the converted relational data. No work on XML IND discovery.

In [2] Nittaya Kerdprasop and Kittisak Kerdprasop proposed a novel technique to discover functional dependencies from the database table. The discovered dependencies help the database designers covering up inefficiencies inherent in their design. Proposed discovery technique is based on the structure analysis of Bayesian network or Bayes net. Most data mining techniques applied to the problem of functional dependency discovery are rule learning and association mining. This technique applies Bayes net to such type of area of application. The proposed technique can reveal functional dependencies within a reduced search space. But the disadvantage of this approach is that its computational complexity is very high.

In [3], Y.V.Sreevanil, Prof. T. Venkat Narayana Rao proposed a new method based on rough set theory. This enables to examine and to eliminate all unnecessary knowledge from the knowledge base by preserving only that part of the knowledge which is really useful. This paper gives some insight into rough sets which can be used to know data dependencies and extraction of knowledge. The ideas envisaged and depicted here are useful in the domain which deal huge collection of databases to analysis and take rational decisions in the areas such as banking, stock markets, medical diagnosis etc.

In [4] Hong Yao, Howard J.Hamilton and Cory J. Butz suggested algorithm FD_MINE to discover FDs.FD_Mine combines domain knowledge of DBMS i.e. closure property, equivalence class etc. and data mining algorithm like Apriori algorithm [23] for discovering frequent itemsets can be used to find FDs from a dataset by setting the minimum support to 2/n, where n is the number of instances in the dataset. To prune redundant candidates, relationships among the FDs are analysed. The FD_Mine algorithm only examines and discover FDs that cannot be inferred from the discovered FDs using the theory of relational databases. The FD_Mine algorithm prunes redundant candidates by using relevant aspect of relational database theory and pruning rules. FD_Mine reduces the data and candidates by discovering equivalences, it has to perform fewer FD check than TANE. FD_Mine has fewer edges in its semi – lattice than TANE. But the drawback of FD_Mine algorithm is that it was restricted to find only simple FDs it was unable to discover other data dependencies like multivalve dependencies and conditional dependencies. In addition, if equivalences are increased at higher levels, the FD-Mine algorithm might be slowed down.

In [5] Yka Hutala, Juha Karkkainen, Pasi Porkka and Hannu Toivonen present TANE algorithm for FD discover.This approach consider partition of relation and determine valid dependencies from partitions. This algorithm searches for dependencies in breadth first or level wise manner. In this method sorting is performed repetitively and then compares tuples in order to determine whether or not these tuples satisfies FD definition. The drawback of this method is that it does not use the already discovered FDs for obtaining new FDs .This approach is inefficient as each and every time sorting and tuple comparison for every values of candidate attribute is needed to perform in order to examine whether or not FD holds. This method provide best performance if dependencies are relatively small but as the size of dependencies is increase to approx. one half of no. of attributes, the number of dependencies is exponential in the number of attribute and condition becomes bad or poor. In short this method is very sensitive to the number of tuples and attributes when dependencies are larger ,i.e when the dataset are large this algorithm will proves impractical and not compute the partitions efficiently.

IV. PROBLEM STATEMENT

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 discovered unique FDs and prune redundant candidate FD from dataset in such way that the search space ,time complexity for computation should be reduced.

V. OBJECTIVE

The main objective of our research is to generate frequent functional dependencies from database with a minimum time complexity and prune redundant functional dependencies. Our approach modify FD_Mine algorithm and the main aim behind this modification is to reduce time complexity and the number of functional dependencies to be checked on a dataset without leading to the mining functional dependencies from data loss of any useful information.

VI. PROPOSED METHODOLOGY

A. FD Generation Algorithm

1) Take a database and its attributes (A_1,A_2….A_m).

2) Performing the computation of partition i.e. perform grouping for no. of tuples that have same values for an attribute.

3) Compute non trivial closure, key set and equivalent set of candidate attributes.

- For these ,join the remaining attribute with the candidate attribute .Compute partition for all that combination .If cardinality of candidate attribute equals to the any of the combination of attribute then add that attribute to non trivial closure of candidate attribute and add FD in FD set.

- If union of candidate attribute and non trivial closure of candidate attribute equals to total attribute in database relation R then add candidate attribute to Key Set
- If cardinality of candidate attribute equals to the cardinality of any of remaining attribute then add FD obtain by that attribute to EQ set.

4) Perform generation of candidate FD for the next level on the basis Candidate Set obtain from previous level.

B. Example

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

TABLE I

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

1) Partition of attribute A,B,C and D are P_A={(1,2,3,4,7),(5,6)},P_B={(1,7),(2,5),(3,6),(4)} ,P_C={(1,2,3,7),(4,5,6)} and P_D={(1,2,3,4,7),(5,6)}

2) Compute nontrivial closure of attribute A, For these compute partition of P_{AB}={(1,7),(2),(5),(3),(6),(4)},P_{AC}={(1,2,3,7),(4),(5,6)} and P_{AD}={(1,2,3,4,7),(5,6)}.Since $|P_A|=|P_{AD}|$.Therefore Closure'(A)=D.Add A->D in FD Set

3) Since $|P_A|=|P_D|$, therefore attribute A is equivalent to attribute D.Add A<->D in EQ Set. Remove D from Candidate Set. Now Candidate Set=(A,B,C)

4) For Key identification take Unions of candidate attribute A and closure' (A) i.e. A U Closure' (A)=(A,D).Since attributes in (A U Closure'(A)) not equals to attribute in R.Therefore no key is identified.

5) Repeat step 2,3 and 4 for Candidate attribute (B,C)) i.e. Finally we get FD Set=A->D,EQ Set=A<->D ,Key Set=Null, Candidate Set=(A,B,C).

6) Perform two attribute combination of candidate set(A,B,C) i.e. New candidate set=(AB,AC,BC).Repeat step 1-5 for new candidate set. Then finally we get EQ Set=(A<->D),FD Set=(A->D, AB->C, BC->A, D->A),Key Set=null

C. Algorithm

Main Procedure

Input: database D and its attribute A_1 , A_2 ,….,A_n
Output: Frequent FD_Set, Candidate Set for next Level, EQ_Set, Key_Set

1). Initialization step

Set R= A_1, A_2 ,….,A_n
Set FD_Set = φ
Set EQ_Set = φ
Set Key_Set=Ø
Set Can_Set= {A_1, A_2,….,A_n }

2). Iteration step

While Can_Set? Φ Do {
For all $A_i \in$ Can_Set Do {
FD_Set = ComputeKey_EQclass_NTClosure (A_i)
}
GenNxtLevelCanSet (Can_Set)
}

3). Display FD_SET, EQ_Set, Key_Set

Procedure ComputeKey_EQclass_NTClosure (Xi)

```
Procedure ComputeKey_EQclass_NTClosure (Xi)
{
  For each Y ⊂ R - Xi
  Do
  {
        If (|π Xi | = | π XiY | then
```



```
                        Add Y to closure' [Xi]
                        Add Xi → Y to FD_Set
                        Add XiY to T_List

                        If (R = Xi U Closure' [Xi] ) then
                                Add Xi to Key_Set
                                Remove Xi from Can_Set
                        EndIf

                        If (| π Y | = | π YXi |) then
                                Add Xi to closure' [Y]
                                Add Y →Xi to FD_Set
                                Add Xi ↔Y to EQ_Set
                                Remove Y from Can_Set
                        End If
                End If
        Next candidate attribute
    }// end procedure
Procedure GenNxtLevelCanSet (Can_Set)
{
For each Ai ∈ Can_Set do
  For each Aj ∈ Can_Set do
                        If (Ai [1]=Aj[1], …, Ai[k-2] = Aj[k-2] and Ai[k- 1] < Aj[k-1]) then
                        {
                                Set Aij = Ai join Aj
                                If ∃  Aij ∈ T_List then delete Aij
                                Else
                                {
                                        Compute the partition Π Aij of Aij
                                        If (R = Aij U Closure' [Aij]) then
                                        Add Aij to          Key_Set

                                        Else
                                        {
                                        Add Aij to Can_Set
                                        set Closure' (Aij) = Closure' (Ai ) U  Closure'(Aj)
                                        }
                                }
                        Endif
} // end procedure
```

VII. RESULT

Actual Time Requirement for both Algorithm (FD_generation and FD_Mine) for some UCI Dataset are shown in Table II.These table specify time complexity between two algorithm and time complexity for FD_generation algorithm for given UCI dataset is low as comparisons to FD_Mine algorithm.

TABLE II

DATASET NAME	NO. OF ATTRIBUTE	NO. OF TUPLES	TIME COMPLEXITY OF FD_GENERATION (MIN)	TIME COMPLEXITY OF FD_MINE (MIN)
Abalone	9	4177	1.73	5.55
Breast Cancer	11	699	4.14	15.21
Iris	5	150	0.0004	0.03
Glass	11	214	1.07	3.45

VIII. CONCLUSION

We propose algorithm to discover Functional dependencies using the properties like key property, equivalent set and non trivial closure property. Here we are modifying previous FD_Mine algorithm so as to reduce time complexity, redundancy and generate frequent FDs.

Firstly, we examine the deficiencies of previous algorithm. We find out time complexity of FD_Mine i.e. $4n^3+n^2$ which is so high and if equivalences at higher level increases ,FD_Mine algorithm slows down. Secondly we find out reason behind this high time complexity. FD _Mine uses 5 procedures i.e.(computeNontrivalclosure(),ObtainFD and Key(),ObtainEQSet(),PruneCandidate(),GeneratenextLevelCandidate())in order to discover FDs and for pruning candidate set. Because of these complexity in terms of number of nested loops and time is increase to very high extent.

As we already mention that our main focus of modifying FD_Mine is to reduce time complexity, prune redundant FDs and keep FDs in reduced form. So in order to achieve our goal first we are using equivalent property and key property as we does not need to check all attribute to discover FDS and second we are using one procedure (compute_key_EQ_NTClosure) to discover FD Set, EQ Set, key set and one procedure (GenNxtLevelCanSet) to perform pruning of candidate set for next level. Time complexity of our approach is $2n^3$.

ACKNOWLEDGMENT

The Abalone, Breast Cancer, Iris and Glass data have been obtained from the UCI dataset.

REFERENCES

[1] 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.

[2] Nittaya kerdprasop and Kittisak kerdprasop data engineering research unit "Functional dependency discovery via bayes net analysis" university avenue, nakhon ratchasima 30000 Thailand, World Scientific and Engineering Academy and Society (WSEAS) Stevens Point, Wisconsin, USA ©2011 Pages 253-258

[3] Y.V.Sreevani1, Prof. T. Venkat Narayana Rao "Identification and Evaluation of Functional Dependency Analysis using Rough sets" (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 1, No. 5, November 2010.

[4] Yao, H., H.J. Hamilton and C.J. Butz, 2002."FD_MINE: Discovering Functional Dependencies in a Database Using Equivalences" 2002. IEEE International Conference on Data Mining (ICDM02), IEEE Computer Society, Maebashi City, Japan, ISBN 0-7695-1754-4,Dec. 9-12, 2002, pp. 729-732.

[5] Huhtala, Y., Kärkkäinen, J., Porkka, P., and Toivonen, H., "TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies." The Computing Journal, 42(2):100-111 (1999).

[6] Maier, D., The Theory of Relational Databases, Computer Science Press, 1983.

[7] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis,"Conditional Functional Dependencies for Data Cleaning," Proc.IEEE 23rd Int'l Conf. Data Eng. (ICDE), pp. 746-755, 2007.

[8] L. Golab, H. Karloff, F. Korn, D. Srivastava, and B. Yu, "On Generating near-Optimal Tableaux for Conditional Functional Dependencies," Proc. Very Large Databases (VLDB) Conf., pp. 376-390, 2008.

[9] G. Cormode, L. Golab, K. Flip, A. McGregor, D. Srivastava, and X.Zhang, "Estimating the Confidence of Conditional Functional Dependencies," Proc. SIGKDD Int'l Conf., pp. 469-482, 2009.

[10] J. Bauckmann, U. Leser, and F. Naumann, "Efficiently Computing Inclusion Dependencies for Schema Discovery," Proc. Second Int'l Workshop Database Interoperability, 2006.

[11] A.Koeller and E.A. Rundensteiner, "Heuristic Strategies for Inclusion Dependency Discovery," On the Move to Meaningful Internet Systems 2004: Proc. Int'l Conf. CoopIS, DOA, and ODBASE,pp. 891-908, 2004.

[12] F. De marchi, S. Lopes, and J.-M. Petit, "Efficient Algorithms for Mining Inclusion Dependencies," Proc. Eighth Int'l Conf. Extending Database Technology (EDBT), pp. 199-214, 2002.

[13] M. Kantola, H. Mannila, K.-J. Raiha¨, and H. Siirtola, "Discovering Functional and Inclusion Dependencies in Relational Databases,"Int'l J. Intelligent Systems, vol. 7, no. 7, pp. 591-607, 1992.

[14] M. Vincent, J. Liu, and C. Liu, "Redundancy Free Mappings from Relations to Xml," Proc. Fourth Int'l Conf. Web-Age Information Management (WAIM), pp. 55-67, 2003.

[15] C. Yu and H.V. Jagadish, "Efficient Discovery of Xml Data Redundancies," Proc. 32nd Int'l Conf. Very large Data Bases (VLDB '06), pp. 103-114, 2006.

[16] M. Kantola, H. Mannila, K.-J. Ra¨iha¨, and H. Siirtola, "Discovering Functional and Inclusion Dependencies in Relational Databases,"Int'l J. Intelligent Systems, vol. 7, no. 7, pp. 591-607, 1992.

[17] H. Mannila and K.-J. Ra¨iha¨, "On the Complexity of Inferring Functional Dependencies," Discrete Applied Math., vol. 40, pp. 237-243, 1992.

[18] S. Bell, "Discovery and Maintenance of Functional Dependencies by Independencies," Proc. Workshop. Knowledge Discovery in Databases (KDD '95), pp. 27-32, 1995

[19] F. Chiang and R.J. Miller, "Discovering Data Quality Rules," VLDB Endowment, vol. 1, no. 1, pp. 1166-1177, 2008.

[20] S. Lopes, J.-M. Petit, and L. Lakhal, "Efficient Discovery of Functional Dependencies and Armstrong Relations," Proc. Seventh Int'l Conf. Extending Database Technology (EDBT): Advances in Database Technology, vol. 1777, pp. 350-364, 2000.

[21] C. Wyss, C. Giannella, and E. Robertson, "Fastfds: A Heuristic- Driven, Depth-First Algorithm for Mining Functional Dependencies from Relation Instances—Extended Abstract," Proc. Third Int'l Conf. Data Warehousing and Knowledge Discovery (DaWaK '01),pp. 101-110, 2001.

[22] N. Novelli and R. Cicchetti, "Fun: An Efficient Algorithm for Mining Functional and Embedded Dependencies," Proc. Eighth Int'l Conf. Database Theory (ICDT), pp. 189-203, 2001.

[23] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. A., and Verkamo, I., "Fast Discovery of Association Rules". Fayyad, U. M., Gregory P. S., Smyth, P.,Uthurusamy, R. (eds.): In *Advances in Knowledge Discovery and Data Mining*,AAAI/MIT Press, 1996, 307-328.

