*International Journal of Engineering Research and Development*
***e**-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com*
*Volume 3, Issue 5 (August 2012), PP. 56-61*

# A Fast and Efficient Method to Find the Conditional Functional Dependencies in Databases

Vijaya Lakshmi[1], Dr. E. V. Prasad[2]
[1]*Department of CSE, JNT University Kakinada, India*
[2]*Professor of CSE & Registrar, JNT University Kakinada, India*

***Abstract—Conditional functional dependencies (CFDs) are the extension of functional dependencies (FDs) by supporting patterns of semantically related constants. These CFDs are very useful to frame the data cleaning rules in relational databases. The CFDs have been proven more effective than FDs in detecting and repairing dirtiness of data. However, finding the CFDs is a difficult task and it also involves rigorous manual effort. This paper proposes a fast and efficient method called BestCFD to find the CFDs from relations. This algorithm works effectively find the CFDs compared to the other algorithms like CTANE. The proposed method is implemented in Java with SQl database connectivity and tested for different datasets. The results validate the effectiveness of this method for finding CFDs in databases.***

***Keywords—Conditional Functional Dependencies, CFDs, BestCFD, Data Quality, Data Cleaning Rules***

## I. INTRODUCTION

Real world data tend to be incomplete, noisy and inconsistent. Data cleaning routines attempt to fill in missing values, smooth out noise while identifying outliers and correct inconsistencies in the data. Conditional functional dependencies (CFDs) are the extension of functional dependencies (FDs) by supporting patterns of semantically related constants. These CFDs were recently introduced for data cleaning. The CFDs have been proven more effective than FDs in detecting and repairing dirtiness of data [1][2]. The CFDs are expected to be adopted in the place of standard FDs by the data cleaning tools [3][4][5]. However, finding the CFDs is a difficult task and it also involves rigorous manual effort.

Different data quality tools are surveyed in [6][7]. Data cleaning-rule discovery is critical to commercial data quality tools [8]. To develop the CFD based data cleaning rules, it is necessary to develop methods to automatically find the CFDs. This paper proposes a fast and efficient method called BestCFD to find the CFDs from relations. This algorithm effectively find the CFDs compared to the level wise algorithms like CTANE[9]. This algorithm is developed from the inspiration of FastFD to find standard FDs [10]. To understand the need for conditional functional dependencies, consider an instance of customer data of a company.

**Example 1**: A company maintains a relation of customer records: Customer (NAME, COUNTRY, CITY, STREET, ZIPCODE, CCODE, ACODE, PHONENO). Each customer tuple contains the NAME, address information (Name of Country COUNTRY, Name of City CITY, Street STRET, Postal Code ZIPCODE) and telephone information( Country code CCODE, area code ACODE, phone number PHONENO) of a customer. An instance of customer data of a company is given in the below Fig.1.

| | NAME | COUNTRY | CITY | STREET | ZIPCODE | CCODE | ACODE | PHONENO |
|---|---|---|---|---|---|---|---|---|
| $t_1$ | Mike | CA | NYC | Tree Ave. | 10012 | 01 | 607 | 1111123 |
| $t_2$ | Rick | CA | NYC | Tree Ave. | 10012 | 01 | 607 | 1111123 |
| $t_3$ | Joe | US | NYC | Elm Str. | 01202 | 01 | 212 | 2234342 |
| $t_4$ | Jim | US | NYC | Elm Str. | 02404 | 01 | 212 | 2234342 |
| $t_5$ | Ben | US | PHI | Oak ave. | 19014 | 01 | 215 | 3344555 |
| $t_6$ | Iam | UK | EDI | High St. | EH4 1DT | 44 | 131 | 4455666 |

***Fig 1:*** An instance of customer data of a company

The traditional functional dependencies (FDs) on the above customer relation are:
$f_1$: [CCODE, ACODE, PHONENO] →[STREET, ZIPCODE, COUNTRY]
$f_2$: [COUNTRY, ZIPCODE] →[CITY]
Recall all semantics of FD: $f_1$ requires that customer records with the same country code, area code and phone number also have the same street, postal code and country. Similarly, $f_2$ requires that two customer records with the same country and zip code also have the same city. The traditional FDs are to hold on all the tuples in the relation. On the other side, the following constraint is supported to hold only when the country is UK. That is, for the customers in the UK, ZIPCODE determines the STREET.
$\phi_3$: [COUNTRY=UK, ZIPCODE] →[STREET]

In other words, $\phi_3$ is an FD that is to hold on the subset of tuples that satisfies the pattern "COUNTRY=UK", rather than on the entire customer relation. It is generally not considered as an FD in the standard definition since $\phi_3$ includes a pattern with data values in its specification.
One may think more accurate than [CCODE, ACODE, PHONENO] →[COUNTRY] in $f_1$, another traditional functional dependency could be defined as [CCODE] →[COUNTR]. Unfortunately, this FD does not hold on customer relation due to a few exceptions: customers with the same country code of 01 may come from either US or Canada, etc. In real world data, such exceptions often occur and prevent modeling the data using FDs. While these constraints cannot be modeled by FDs, they may be enforced by:
$\phi_{1a}$: [CCODE=44] →[COUNTRY=UK]
Similar to $\phi_3$, many constraints hold on a subset of the tuples instead of all the tuples in a relation. Another example on the customer relation is [CCODE, ACODE] ] →[CITY], which does not hold in US but holds in UK, China and most other countries.
$\phi_{4a}$: [CCODE=44, ACODE] →[CITY]
In US, although it does not hold in general, it still holds for same area codes.
$\phi_{4b}$: [CCODE=01, ACODE=212] →[CITY=NYC]
$\phi_{4c}$: [CCODE=01, ACODE=215] →[CITY=PHI]
The following constraints again not considered as FDs.
$\phi_{1b}$: [CCODE=01, ACODE=607, PHONENO] →[STREET, ZIPCODE, COUNTRY=US]
$\phi_{2a}$: [COUNTRY=US, ZIPCODE=10012] →[CITY=NYC]
$\phi_{2b}$: [COUNTRY=US, ZIPCODE=19014] →[CITY=PHI]
The $\phi_{1b}$ refine the standard FD $f_1$ given above, while $\phi_{2a}$ and $\phi_{2b}$ refines the FD $f_2$. This refinement is essentially enforces a binding of semantically related data values. It can be observed that the tuples $t_1$and $t_2$ in Fig.1 do not violate $f_1$, they violate its refined version $\phi_1$, since the country cannot be CA if the country code and area code are 1 and 607, respectively.

In this example, the constraints $\phi_{1a,b}$, $\phi_{2a,b}$, $\phi_3$, $\phi_{4a,b,c}$ capture a fundamental part of the semantics of the data. However, they cannot be expressed as standard FDs and are not considered in previous work on data cleaning. In response to the practical need for such constraints, a novel extension of traditional FDs is introduced referred to as Conditional Functional Dependencies (CFDs) that are capable of capturing the notion of "correct data" in these situations. A CFD extends an FD by incorporating a pattern tableau that enforces binding of semantically related values.

## II. CONDITIONAL FUNCTIONAL DEPENDENCIES(CFDS)

Consider a relation schema R defined over a fixed set of attributes, denoted by attr(R). For each attribute A $\epsilon$ attr(R), its domain is specified in R, denoted as dom(A).
A CFD φ on R is a pair (R : X →Z, Tp), where (i) X, Z are sets of attributes in attr(R), (ii) X → Z is a standard fd, referred to as the FD embedded in φ; and (iii) Tp is a tableau with attributes in X and Z, referred to as the pattern tableau of φ, where for each A in X U Z and each tuple t $\epsilon$ Tp, t[A] is either a constant 'a' in dom(A), or an unnamed variable '_' that draws values from dom(A).

If A occurs in both X and Z, we use $t[A_L]$ and $t[A_R]$ to indicate the occurrence of A in X and Z , respectively, and separate the X and Z attributes in a pattern tuple with '||'. We write φ as (X → Z, Tp) when R is clear from the context, and denote X as LHS(φ) and Z as RHS(φ).

**Example 2:** The constants $\phi_{1a,b}$, $\phi_{2a,b}$, $\phi_3$, $\phi_{4a,b,c}$ on the customer table given in Example 1 can be expressed as CFDs $\varphi_1$ (for $\phi_{1a}$ and $\phi_{1b}$), $\varphi_2$ (for $\phi_{2a}$ and $\phi_{2b}$), $\varphi_3$ (for $\phi_3$), and $\varphi_4$ (for $\phi_{4a}$, $\phi_{4b}$ and $\phi_{4c}$). This is shown in Fig. 2. If we represent both data and constraints in a uniform tableau format, then at one end of the spectrum are relational tables which consists of data values without logic variables, and at the other end are traditional functional constraints which are defined in terms of logic variables but without data values, while the CFDs are in between.

(a) Tableau $T_1$ of $\phi_1$ = ([CCODE, ACODE, PHONENO] → [STREET, ZIPCODE, COUNTRY], $T_1$)

| CCODE | ACODE | PHONENO | STREET | ZIPCODE | COUNTRY |
|---|---|---|---|---|---|
| - | - | - | - | - | - |
| 44 | - | - | - | - | UK |
| 01 | 607 | - | - | - | US |

(b) Tableau $T_2$ of $\phi_2$ = ([COUNTRY, ZIPCODE] → [CITY], $T_2$)

| COUNTRY | ZIPCODE | CITY |
|---|---|---|
| - | - | - |
| US | 10012 | NYC |
| US | 19014 | PHI |

(c) Tableau $T_3$ of $\phi_3$ = ([COUNTRY, ZIPCODE] → [STREET], $T_3$)

| COUNTRY | ZIPCODE | STREET |
|---|---|---|
| UK | - | - |

(d) Tableau $T_4$ of $\phi_4$ = ([CCODE, ACODE] → [CITY], $T_4$)

| CCODE | ACODE | CITY |
|---|---|---|
| 44 | - | - |
| 01 | 212 | NYC |
| 01 | 215 | PHI |

***Fig. 2:*** Example CFDs

## III. PROPOSED METHOD FOR FINDING CFDS (BestCFD)

The ptoposed method, BestCFD for the discovery of Conditional Functional Dependencies (CFDs) in databases is explained below.

Given an instance r and a support threshold k, the BestCFD method finds a canonical covers of all minimum CFDs φ such that sup(φ,r) ≥k in depth-first way.

For each attribute A in attr(R), BestCFD looks for all CFDs of the form φ =(Y→A, $t_p$) such that Y ⊆ attr(R)\{A}, φ is minimal and sup(φ,r) ≥k. These CFDs are denoted by Covers(A,r,k). All k-frequent minimal CFDs in r can then be obtained as $U_{A\epsilon attr(R)}$Covers(A,r,k).

The function Covers(A,r,k) first takes out the set of the k-frequent free itemsets $Fr_k(r)$ of r, in which itemsets are kept in the ascending order with respect to their sizes. To efficiently recover the elements in $Fr_k(r)$, the Covers(A,r,k) function indexes those itemsets in a hash table.

For each $(X,t_p)$ in $Fr_k(r)$, Covers function maintains the set of minimal difference sets produced from all tuples in $r_{tp}$ and are denoted as $D_A^m(r_{tp})$.

For a given $(X,t_p)$ $\epsilon Fr_k(r)$, Covers function recursively calls MinCovers function to find a minimal cover Y of $D_A^m(r_{tp})$.

The function MinCovers finds the minimal covers by traversing all subsets attr(R)\{A} in a depth-first way. We assume an ordering $<_{attr}$ on attr(R). All subsets of attr(R)\{A} are then enumerated in a depth-first, left-to-right fashion based on the given attribute ordering.

## IV. RESULTS AND ANALYSIS

A Java based GUI tool, shown in Fig. 3, has been developed based on the proposed algorithm for the discovery of conditional functional dependencies. The developed toll has been tested for different datasets on an Intel core i3, 3.10 GHz computer with 3 GB of main memory on Windows 7 operating system. The datasets used in this work are taken from the UCI machine learning repository (http://archive.ics.uci.edu/ml/), namely, the Wisconsin breast cancer (WBC) and Chess datasets. The parameters of the datasets used are given in the following table 1.

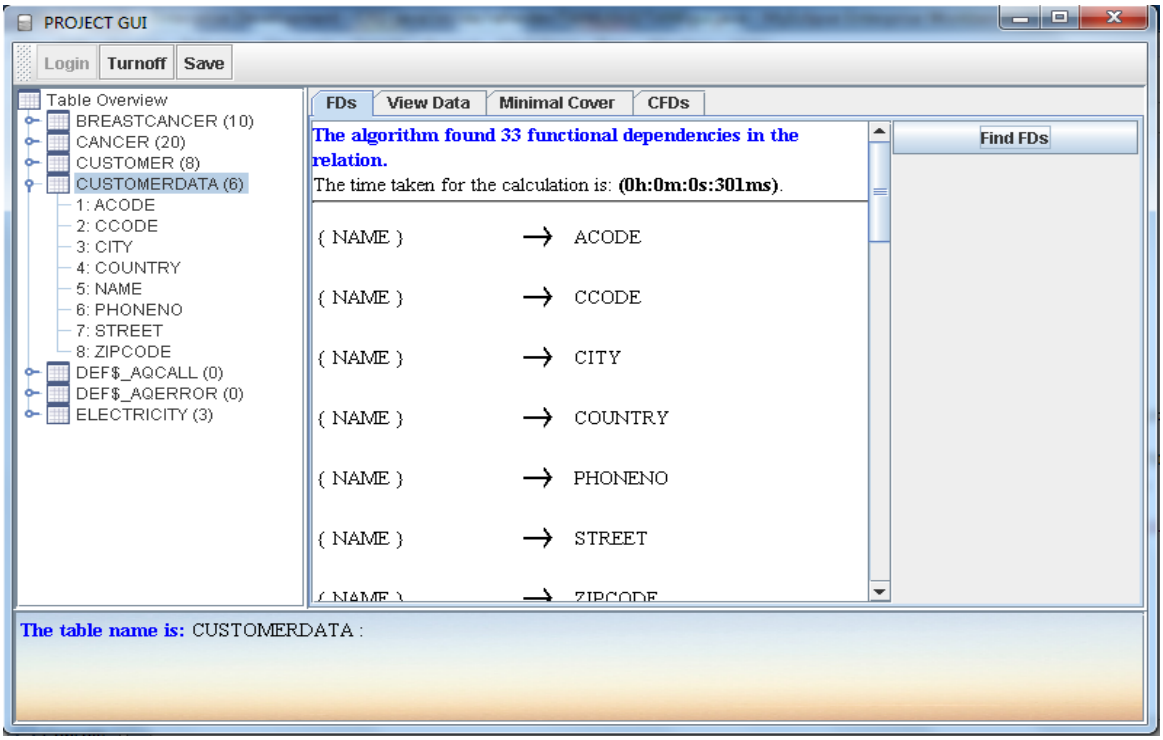


***Fig. 3 :*** Tool for the discovery of CFDs

***Table 1:*** Parameters of datasets

| *Dataset* | *Arity* | *Size (No. of tuples)* |
|---|---|---|
| Wisconsin breast cancer (WBC) | 11 | 699 |
| Chess | 7 | 28,056 |
| Tax | 14 | 20,000 |

A synthetic dataset for tax records generated by populating the database is used to find the scalability of the proposed scheme. The following Fig. 4, depicts the response time of the proposed BestCFD algorithm and CTANE for arity of 7, CF of 0.7, support of 0.1% and for different values of database size. From Fig. 4, it can be observed that the BestCFD takes less time and works better compared to CTANE. The Fig. 5, shows the response time for database size of 20,000, CF of 0.7, support of 0.1% and for different values of arty from 7 to 31. The Fig. 5 clearly shows that the BestCFD works well even when the arity is large. The fig. 2, also shos that the CTANE does not work well is the arity is large.

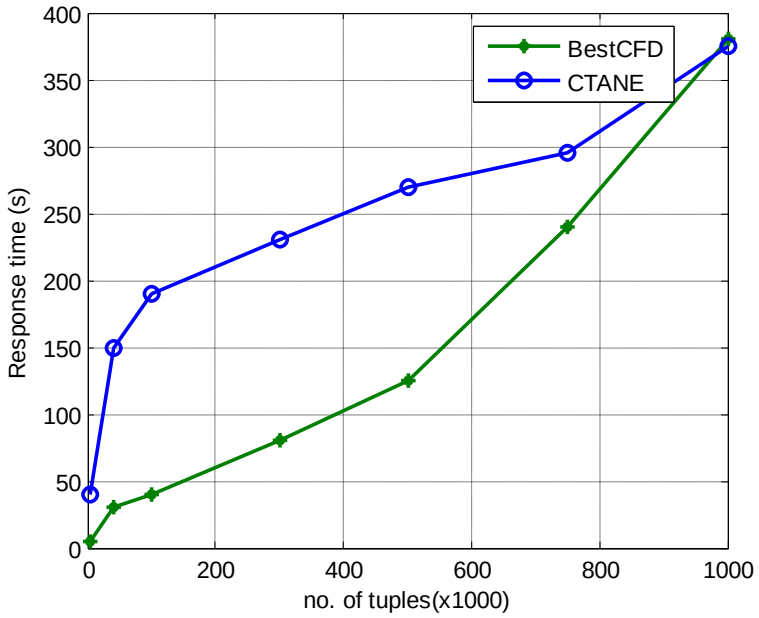


***Fig. 4:*** Response time for different data sizes

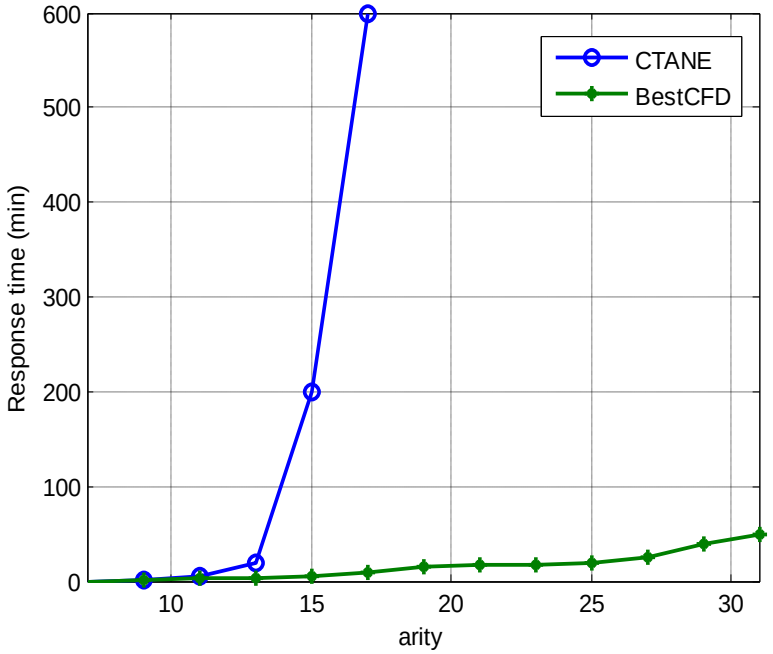


***Fig. 5:*** Response time for different number of attributes (arity)

The Fig. 6 is achieved for Wisconsin Breast Cancer dataset for different values of support. From Fig. 6, t can be seen that the CTANE is sensitive to the support value and the performance of CTANE increases with the increase of support. It can also observed that the BestCFD is less sensitive to the support. Fig. 7 shows the number of CFDs generated for Wisconsin Breast Cancer dataset for different values of support. The number of CFDs decreases with the increases of support. The level wise algorithm, CTANE works well when the arity of a relation is small and the support threshold is large. But it does not work well when the arty increases. The BestCFD works well even when the arity is large.

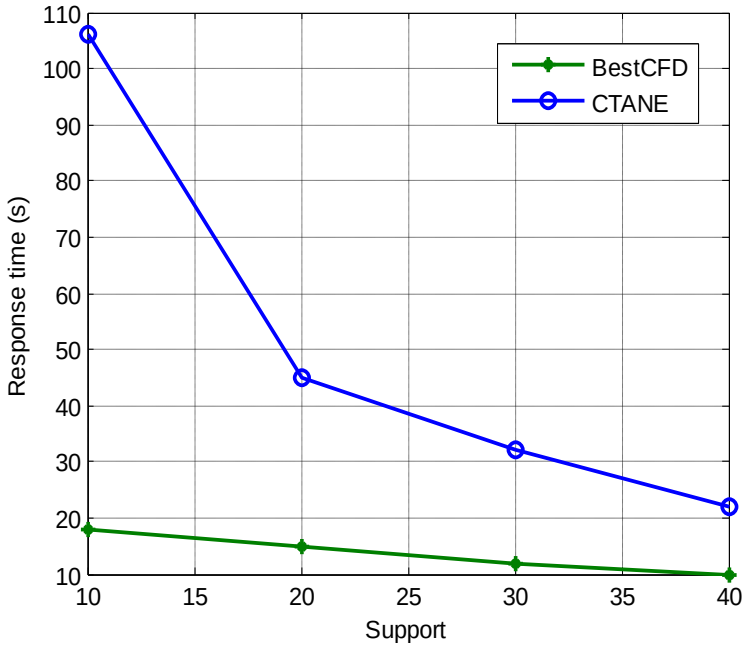


***Fig. 6:*** Variation of response time with support for WBC dataset

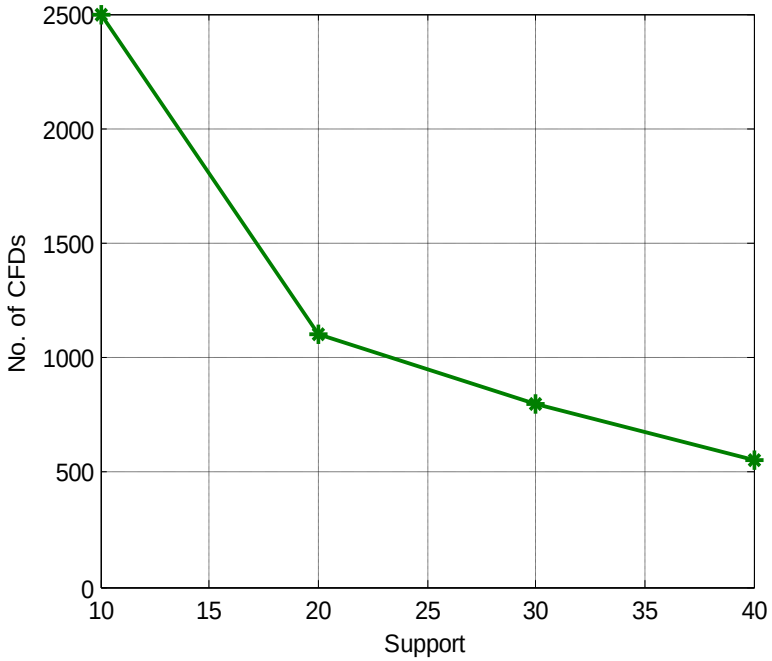


***Fig. 7:*** Variation of no. of CFDs with support for WBC dataset

## V. CONCLUSION

The CFDs are very useful in detecting and repairing dirtiness of data. This paper proposes a fast and efficient method called BestCFD to find the CFDs from relations. This algorithm works based on the depth-first approach and effectively find the CFDs compared to the level wise algorithms like CTANE. A Java based GUI tool has been developed based on the proposed algorithm for the discovery of conditional functional dependencies. This tool is tested for different datasets to assess its performance. The test results are compared with CTANE to validate the applicability of this method for CFD discovery. The level wise algorithms work well when the arity of a relation is small and the support threshold is large. But it does not work well when the arty increases. The proposed scheme, BestCFD works well even when the arity is large.

## REFERENCES

[1]. W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for capturing data inconsistencies," TODS, vol. 33, no. 2, June 2008.

[2]. G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma, "Improving data quality: Consistency and accuracy," in VLDB, 2007.

[3]. M. Arenas, L. E. Bertossi, and J. Chomicki, "Consistent query answers in inconsistent databases," TPLP, vol. 3, no. 4-5, pp. 393–424, 2003.

[4]. J. Chomicki and J. Marcinkowski, "Minimal-change integrity maintenance using tuple deletions," Information and Computation, vol. 197, no. 1-2, pp. 90–121, 2005.

[5]. J. Wijsen, "Database repairing using updates," TODS, vol. 30, no. 3, pp. 722–768, 2005.

[6]. C. Batini and M.Scannapieco, Data Quality: Concepts, Methodologies and Techniques. Springer, 2006.

[7]. E. Rahm and H. H. Do, "Data cleaning: Problems and current approach." IEEE Data Eng. Bull., vol. 23, no. 4, pp. 3-13, 2000.

[8]. Gartner, "Forecast: Data Quality tools, worldwide, 2006-2011," 2007.

[9]. Wenfei Fan, Floris Greerts, Jianzhong Li and Ming Xiong, "Discovering Conditional Functional Dependencies," IEEE transactions on Knowlwdge and Data Enginering, Vol 23, No.5, May 2011, pp. 1-15.

[10]. C. M. Wyss, C. Giannella, and E.L. Robertson, "FastFDs: A heuristic-driven,depth-first algorithm for mining functional dependencies from relation instances – extended abstract," in DaWak,2001.