

Inconsistent Relational Data Cleaning By Detecting Conditional Functional Dependencies

Challa Neehar
M.Tech (CSE) Student
Dept of CSE
QISCET, India
challaneehar@gmail.com

T.V.Sai Krishna
Assoc Professor
Dept of CSE
QISCET, India
tvsai.kris@gmail.com

Abstract—Conditional functional dependency (CFD) is an integrity constraint. CFDs were used to eliminate redundancy. Traditional functional dependencies (FDs) were recently replaced by CFDs for data cleaning. FDs were mainly used for schema design where as CFDs were aimed at capturing the consistency of data by using patterns of semantically related constants. Inconsistent Relational data can be detected by using CFDs. The discovery problem is more difficult in the case CFDs when compared to FDs also mining patterns in CFDs will introduce more challenges. For the purpose of CFD discovery we are providing three methods they are CFDMINER, CTANE and FASTCFD. CFDMINER is used for discovering constant CFDs, it employs item set mining on both free and closed item sets for constant CFD discovery. Constant CFD discovery is essential for the process of data cleaning and integration. CTANE is used for discovering general CFDs. It uses breadth first approach or level wise approach for discovering general CFDs it works well when the DBSIZE of the relation is large but does not scale good relatively when the arity of the relation is large. FASTCFD is also used for discovering general CFDs uses depth first approach for discovering general CFD. FASTCFD works more efficiently than CTANE when the arity of the relation is large. But it does not scale well when the DBSIZE of the relations is large

Keywords: Semantics, Redundancy, integrity constraint, constant CFD, general CFD

I. INTRODUCTION

Data cleaning is an important aspect in data engineering. The difference between standard functional dependencies (FDs) and CFD is they

enforce patterns of semantically or meaning fully related constants. Compared to FDs, CFDs are more effective than FDs in discovering and repairing inconsistencies (dirtiness) of data [2], [1]. CFDs are going to be used by data-cleaning tools which currently employing standard FDs

CFD-based cleaning methods should necessary adopt some techniques that can automatically discover and learn cfd from sample data. The practical problem involved in the discovery of CFD is to find canonically covered CFDs in a sample instance r_1 of a relational schema R_1 . More over the canonical covered CFDs should be nontrivial, left reduced and minimal[3] for the reduction of redundancy.

Example 1: The following relational schema cust is taken from [10]. It specifies a customer in terms of the customer's phone (country code (CC), area code (AC), phone number (PN)), name (NM), and address (street (STR), city (CT), zip code (ZIP)). An instance r_0 of cust is shown in Fig. 1. Traditional FDs that hold on r_0 include the following:

$$f_1 : [CC, AC] \rightarrow CT$$
$$f_2 : [CC, AC, PN] \rightarrow STR$$

Here f_1 requires that two customers with the same country- and area-codes also have the same city; similarly for f_2 . In contrast, the CFDs that hold on r_0 include not only the FDs f_1 and f_2 , but also the following (and more):

- $\phi_0: ([CC, ZIP] \rightarrow STR, (44, I))$
- $\phi_1: ([CC, AC] \rightarrow CT, (01, 908 I MH))$
- $\phi_2: ([CC, AC] \rightarrow CT, (44, 131 I EDI))$
- $\phi_3: ([CC, AC] \rightarrow CT, (01, 212 I NYC))$

CC	AC	NM	STR	CT	ZIP
010	908	rick	Tree Av.	MH	0797
010	908	Joe	Tree Av.	MH	0797
010	212	Jim	5th Ave	NYC	0120
010	908	Ben	Elm St.	MH	07974
441	131	Ian	High Str.	EDI	EH4 D
441	131	Ian	High Str.	EDI	EH4 D
441	908	Sean	Port PI	MH	W1B J
010	131	Mike	3rd Str.	UN	0120

Table1. An instance r_0 of the cust relation

Observe that the pattern tuple in each of ϕ_1 – ϕ_3 consists of only constants in both its LHS and RHS. Such CFDs are referred to as *constant CFDs*.

A. Prior Work

For the purpose of database[8] design, data archiving, OLAP and data mining the discovery problem has been studying from the past two decades. It was first noted in [2], that the discovery problem is inherently exponential in the arity $|R|$ of the schema R of sample data r . previously TANE was employed for FD discovery which is a level wise algorithm, also uses pruning strategy. As level wise algorithms takes more time for FD discovery i.e exponential time in $|R|$ later depth first strategy was used in the place of level wise algorithm. Depth first strategy takes linear time in the size of the output[4]. In the time being for the purpose of constant CFD discovery heuristic algorithms have been brought forth. because constant CFD discovery is closely related to association rule mining. It was later found that CFDs have some potential applications in the arena of data cleaning this urged for the further investigation of CFD discovery. a) constant CFD discovery is important for object identification. one wants the efficient methods for the discovery of constant CFD alone and it has to be faster than normal CFD discovery. b) Level wise algorithms may not perform well with the arity of the relation. More effective methods have to be designed to deal with datasets of large arity. (c) Adding to above work some host of techniques have been found for association rule mining.

B. Contributions

In light of these considerations we provide three algorithms for CFD discovery: one for discovering constant CFDs,[5] and the other two for general CFDs and later experimental study of those algorithms were done with both real and synthetic data.

1) *Cfdminer*: It is our first algorithm, for constant CFDs discovery. It uses depth first approach and works well on both free and closed item sets. It efficiently discovers constant CFDs when compared to other algorithms

2) *Ctane*: It is the second algorithm which was used to discover general CFDs. It uses level wise search approach for discovering general CFD. It works well when the DBSIZE of the relation is large. Pruning strategy was used in discovering non redundant general CFD

3) *Fastcfd*: This is our third algorithm which was used to discover general CFDs. It employs depth first approach instead of level wise search. It works well when the arity of the relation is large using pruning strategy. It discovers minimal CFDs which were canonically covered.

4) *Experimental Study*: This is our final contribution. Experimental study is the study of the efficiency and effectiveness of the aforementioned algorithms. Both real data and synthetic data have been considered for this experimental study. Scalability of the algorithms have been tested varying (1) DBSIZE (2) arity (3) support threshold k (4) correlation factor. We found that CTANE works well with the size of the relation. Fast CFD works with some magnitude faster when the arity of the relation is large. CFD miner mines on both closed and open sets. Whereas CTANE and Fast CFD works only on closed itemsets.

C. Organization

Section 2 defines CFD definition, classification of CFDs and the discovery problem of CFDs. Whereas section 3 describes constant CFD discovery related algorithm i.e CFDMINER and free and closed item set sets. Section 4 describes about general CFD discovery pointing out two different algorithms they were CTANE and Fast CFD. Section 5 tells us about the experimental study, experimental settings, experimental results and scalability. Conclusion has been given in section 6 and after that references were given.

II. CFD

CFD is an acronym for conditional functional dependencies. As mentioned earlier it is an integrity constraint. CFDs are mainly used to eliminate data redundancy. CFDs and FDs are mainly differentiated by the extension that was posed by CFDs is that they extend standard FDs by enforcing patterns of semantically related constants.

A. Definition

Consider a relation schema R defined over a fixed set of attributes, denoted by $\text{attr}(R)$. For each attribute $A \in \text{attr}(R)$, we use $\text{dom}(A)$ to denote its domain. **CFDs:** A *conditional functional dependency* (CFD) φ over R is a pair $(X \rightarrow A, t_p)$, where (1) X is a set of attributes in $\text{attr}(R)$, and A is a single attribute in $\text{attr}(R)$, (2) $X \rightarrow A$ is a standard FD, referred to as the FD *embedded in* φ ; and (3) t_p is a *pattern tuple* with attributes in X and A , where for each B in $X \cup \{A\}$, $t_p[B]$ is either a constant 'a' in $\text{dom}(B)$, or an unnamed variable ' ' that draws values from $\text{dom}(B)$.

B. Classification of CFDs

The CFDs have been classified as (a) constant CFD (b) variable CFD (c) minimal CFD (d) frequent CFD.

(a) *Constant CFD:* A CFD $(X \rightarrow A, t_p)$ is called a *constant CFD* if its pattern tuple t_p consists of constants only, i.e., $t_p[A]$ is a constant and for all $B \in X$, $t_p[B]$ is a constant.

(b) *Variable CFD:* a CFD can be called as a *variable CFD* if $t_p[A] = _$, i.e., the RHS of its pattern tuple is the unnamed variable ' '.

(c) *Minimal CFD:* a minimal CFD is a CFD which is non trivial, left reduced and non redundant such that there are canonical covered

(d) *Frequent CFD:* a frequent CFD is a CFD that is made after the frequency of occurrence in a relational data. for example 'k' frequent CFD. Adding to the above there is another CFD called general CFD which can be either a variable CFD or constant[9].

C. Discovery problem for CFDs

Real life data is often dirty that contains errors and noise. So for a given relation r of schema R , if we execute an algorithm for detecting CFD that will return all the CFDs that hold on r . The set of CFDs returned by the

algorithm may contain trivial, redundant and unnecessarily large amount of data. But we don't need all that data for our purpose. For this reason we need to find canonical covered, non redundant data and the important thing is that the CFDs returned by the algorithm must be minimal.

D. Problem statement

A canonical cover of CFDs on r w.r.t to k is a set of minimal, k -frequent CFDs in r , such that set is equivalent to the set of all k -frequent CFDs that hold on r . Finally the discovery problem can be concluded as in a relational schema R for a given instance and support threshold, the discovery problem is to find a canonical covered CFDs. Supposedly canonical covered CFD should not contain non redundant CFD on r .

III. CONSTANT CFD DISCOVERY

A. Cfminer

CFDMINER is employed for detecting constant CFDs. CFD miner efficiently discovers constant CFD when compared to other algorithms. CFD miner algorithm is based on left reduced constant CFD and free & closed item sets. CFD miner discovers k -frequent minimal constant CFD. For CFD miner to work efficiently we employed gc growth algorithm that simultaneously mines closed and free item sets. The gc growth algorithm returns a mapping that associates with each k -frequent closed and free item sets. Constant minimal CFD discovery is important for the process of data integration. CFD miner techniques were used by fastCFD for better efficiency and optimization. CFD miner mines itemsets both closed and open which is closely related to association rule mining.

IV. GENERAL CFD DISCOVERY

A. General CFDs

General CFDs are those CFD which can be either a variable or a constant. General CFD discovery was found to be more useful than constant CFD discovery. Two algorithms were proposed for this purpose they were CTANE and FastCFD. Although both CTANE and FastCFD were used to mine general CFD they were using different search approaches to find CFD. CTANE uses level wise search approach where as FastCFD uses depth first search approach. They both operate only on closed item sets.

B. Ctane (a level wise algorithm)

CTANE is used to discover general CFD. It uses breadth-first or level wise approach to discover CFDs. It works or it discovers CFDs relatively well when the DBSIZE of the relation is large when compared to other algorithms. Also it does not scale well when the arity of the relation is large. It leverages closed item set mining. It does not operate on open sets. This algorithm is not as Fast as FastCFD as it traverses attribute set pattern in level wise way. It discovers k-frequent minimal CFDs for a support threshold 'k'. It uses pruning strategy. Pruning strategy make level wise search feasible in practice as it discovers only minimal CFDs eliminating redundancy in CFD discovery

C. Fast cfd(a depth first algorithm)

Fast CFD is used to discover general CFD unlike CFDMINER which is used to discover constant CFD. It uses depth first approach to discover CFDs. It discovers well when the arity of the relation is large. Arity is the number of columns in a relation. It also scales marginally well when the size of the relation is large but not as good as CTANE. Similar to CTANE it leverages on closed item sets. It cannot leverage on open sets. Coming to the efficiency point of view it is far better than CTANE when the arity of the relation is large pruning strategy makes level wise search feasible in practice. Redundancy elimination in discovering CFDs is another important feature in FastCFD..

V. EXPERIMENTAL ANALYSIS

We now present the experimental analysis of our algorithms CTANE, FastCFD and CFDMINER. We investigated the effect of some factors that could influence the scalability and the number of CFDs generated. The factors are 1) DBSIZE 2) correlation factor 3) arity 4) support threshold 'k'.

A. Experimental Settings

The experiments were conducted both on synthetic datasets and real life data. The algorithms which were mentioned above have been coded in c++. The program code has been tested with intel i5 processor. The system memory is 64 gb and the operating system is linux operating system. The entire program code run on main memory. Each experiment was repeated for 6 times and the average is presented here by setting the values of ARITY, DBSIZE, CF and K to some fixed magnitude [10].

B. Experimental Results

1) Scalability experiments: In this set of experiments we study the performance of our algorithms by varying DBSIZE, ARITY, CF, and support threshold 'k'.

a) Scalability w.r.t DBSIZE: Fixing the arity=7, CF=0.7 and support ratio which is defined as k/DBSIZE at 0.1% we have varied DBSIZE from 20k to 1 million tuples. CFDMINER which mines only constant CFD is manytimes faster than others. When DBSIZE is less than 1 million FastCFD performed well than CTANE later CTANE performed well constantly.

b) Scalability w.r.t 'K': Fixing CF=0.7, DBSIZE 100k and support ratio at 0.1% we have varies 'k' from 50 to 150. In this case primarily when k value is low Fast CFD performed just better than CTANE but when the value increases CTANE out performed all other algorithms. This shows that CTANE is highly sensible to the value of 'k'. Also we saw that minimal CFD discovery gets decreased when the value of 'k' increases.

c) Scalability w.r.t CF: Fixing dbsize=50k, k=50, arity=9 we have varied the value of cf from 0.3 to 0.7. Just in the case of 'k' primarily for the low value of cf fast cfd performed well than ctane. But as the value of cf increases ctane performed well than fast cfd and cfminer. Also we can see there is a slight decrease in the performance of FASTCFD can be observed when the value of cf is increased.

d) scalability w.r.t ARITY: Fixing cf=0.7, dbsize =20k, supp%=0.1% the arity has increased from 7 to 31. Ctane, as discussed earlier does not well with the arity of the relation. As expected fast cfd performed orders of magnitude better than CTANE when the arity of the relation is increased.

C. Real Data Experiments

We have conducted experiments on the real-life

Data sets	arity	size
Road accidents	10	700
Card game	8	27.056
Tax	14	19.086

data, including the card game, road accidents, and synthetic tax datasets. For every dataset, k value was varied. Figures 11, 12, 13 show the response times of CTANE and FastCFD when k value is varied, but the figures 14, 15 and 16

show the corresponding numbers of CFDs discovered by the above algorithms. Consistent with our previous experiments, CTANE is very sensitive to the support threshold k , and its performs well when k value increases. FastCFD is less sensitive to k , and its performs just well only when value of k increases. Both algorithms discover lesser number of CFDs as k increases.

D. Summary

Experimental results suggest the following. (1) Constant CFD discovery (CFDMINER) can be multiple orders of magnitude quicker than general CFD discovery (CTANE or FastCFD). (2) CTANE usually performs well with small arity of a relation and with large support threshold value, but it scales poorly when the arity increases. (3) FastCFD is more efficient than CTANE with large value of arity in a relation. (4) Our technique of optimization based on closed-itemset mining is effective: FastCFD significantly performs well especially when the arity of relation is large. Also from the above experiments we can see that the parameters arity and DBSIZE playing an important role in determining the scalability and the efficiency of producing CFDs. Also we have seen that CFD miner has very less to play in discovering CFDs. because CFDMINER is only intended to discover constant CFD only. whereas both CTANE and FastCFD were used in discovering both constant and variable CFD i.e. general CFD.

E. Graphical Results

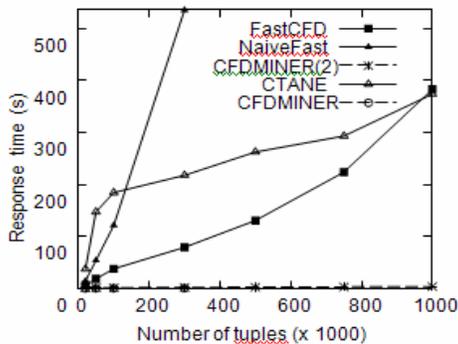


Fig. 1. Scalability w.r.t. DBSIZE

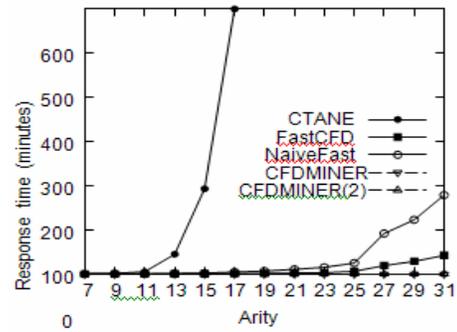


Fig. 2. Scalability w.r.t. ARITY

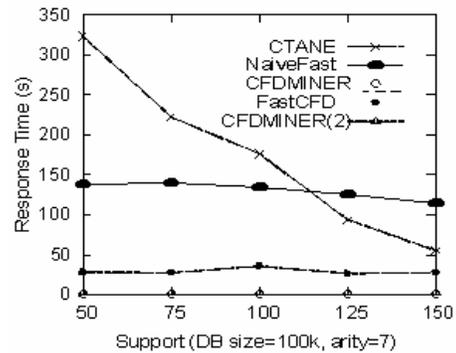


Fig. 3. Scalability w.r.t. K

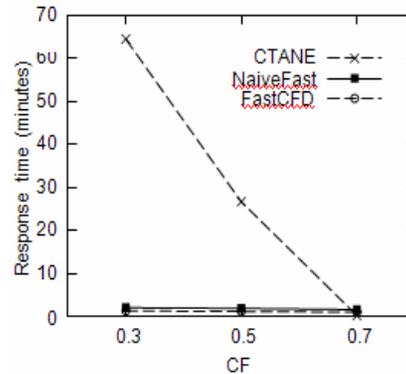


Fig. 4. Scalability w.r.t. CF

VI. CONCLUSIONS

Three algorithms have been developed and executed for discovering minimal CFDs they were 1) CTANE 2) FastCFD 3) CFDminer. All the above algorithms provide a set of tools for different applications which has to be chosen by user .1) CTANE is an algorithm which is used to find general minimal CFDs.It uses breadth first or level wise approach for detecting general minimal CFDs. 2) FastCFD is an algorithm which was developed for detecting general minimal CFDs.This algorithm uses depthfirst approach. 3) CFDminer is an algorithm for detecting minimal constant CFDs.Minimal constant CFDs are important for data integration and data cleaning[6].The above proposed algorithms have unique advantages relatively.when we want to find only minimal constant CFDs we can make use of CFDminer with out affording for general minimal CFDs.Also when we want all the CFDs to be detected, we can simply switch on to either fastCFD or ctane algorithms. The algorithms above are experimentally tested by varying some parameters. Scalability experiments were conducted on algorithms and found that CTANE performs well when the DBSIZE of the relation is large and fast CFD scales well when the arity of the relation is large, CFDminer performs order of magnitude well while detecting minimal constant CFD.We conducted experiments on both real-life and Synthetic data to evaluate the performance of above algorithms. We stressed our concentration towards detecting minimal constant CFDs.We detected CFDs using above algorithms which were non redundant .Finally we focussed for future to access quality[7] measures for CFDs .

REFERENCES

- [1] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma, "Improving data quality: Consistency and accuracy," in *VLDB*, 2007.
- [2] M. Arenas, L. E. Bertossi, and J. Chomicki, "Consistent query answers in inconsistent databases," *TPLP*, vol. 3, no. 4-5, pp. 393-424, 2003.
- [3] J. Chomicki and J. Marcinkowski, "Minimal-change integrity maintenance using tuple deletions," *Information and Computation*, vol. 197, no. 1-2, pp. 90-121, 2005.
- [4] J. Wijsen, "Database repairing using updates," *TODS*, vol. 30, no. 3, Pp 722-768, 2005.
- [5] C. Batini and M. Scannapieco, *Data Quality: Concepts, Methodologies and Techniques*. Springer, 2006.

- [6] E. Rahm and H. H. Do, "Data cleaning: Problems and current approaches." *IEEE Data Eng. Bull.*, vol. 23, no. 4, pp. 3-13, 2000.
- [7] Gartner, "Forecast: Data quality tools, worldwide, 2006-2011."
- [8] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases* Addison-Wesley, 1995.
- [9] L. Golab, H. Karloff, F. Korn, D. Srivastava, and B. Yu, "On generating near-optimal tableaux for conditional functional dependencies," in *VLDB*, 2008
- [10] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for capturing data inconsistencies," *TODS*, vol. 33, no. 2, June 2008.

AUTHORS PROFILE



Challa Neehar, pursuing M.Tech (CSE) from QIS College of Engineering and Technology, Ongole Affiliated to JNTU, Kakinada. His Interested areas include Data Ware Housing and Data Mining, Operating Systems Data Base Mangement Systems



Mr T.V.Sai Krishna currently working as Associate Professor in Dept of CSE. He received his B.Tech from JNTU Hyderabad and M.Tech from JNTU, Anantapur. He had 9 years of Teaching Experience and Life Member of ISTE. He has published several papers at various national and International Journals and Conferences. His Research areas include Image Processing, Data mining and Network Security.