International Journal of Computer Applications (0975 – 8887)
Volume 74– No.15, July 2013

Mining Constant Conditional Functional Dependencies for Improving Data Quality

D Devi Kalyani
Department of Computer Science Engineering
Gitam School of Technology,
Gitam University, Hyderabad, India.

ABSTRACT

This paper applies the data mining techniques in the area of data cleaning as effective in discovering Constant Conditional Functional Dependencies(CCFDs) from relational databases . These CCFDs are used as business rules for context dependent data validations. Conditional Functional Dependencies(CFDs) are an extension of Functional dependencies(FDs) which captures the consistency of data by supporting patterns of semantically related constants. Based on the hierarchy between FDs, CFDs and Association Rules :Union of Association Rules are CFDs, while union of CFDs are FDs. This paper proposes the algorithms used for Association Rule discovery to be reused for CCFD Mining i.e CFDs with constant patterns only . Three algorithms for CCFD mining namely CCFD-FPGrowth, CCFD-AprioriClose and CCFD-ZartMNR are provided in this paper. CCFD-FPGrowth uses FP-growth algorithm to find frequent itemsets and then generates rules as constant patterns from the set of frequent itemsets using modified Agrawal Association rule Generation algorithm. CCFD-AprioriClose uses Apriori algorithm to find frequent closed itemsets and then generates rules as constant patterns from the set of frequent closed itemsets using modified Agrawal Association rule Generation algorithm. CCFD-ZartMNR uses Zart algorithm to find closed itemsets and minimal generators and then generates minimal non-redundant rules from the set of closed itemsets. Experimental results on two real-world data sets show that this approach performs well across several dimensions such as recall, runtime and scalability.

General Terms

Data Quality, Data Cleaning, Data Mining

Keywords

Data Cleaning, Constant Conditional Functional Dependency(CCFD), Conditional Functional Dependency(CFD), Frequent Pattern Growth (FP) tree, Frequent Itemsets, Closed Itemsets

1. INTRODUCTION

A company's most important asset is information. A corporation's ability to compete, adapt, and grow in a business climate of rapid change is dependent on large extent on how well the company uses information to make decisions. Sharing information that is not clean and consolidated to the full extent can substantially reduce the effectiveness of a system.

Data cleansing, is the process of ensuring data quality in information systems. This process involves inspecting a single set of records or between multiple sets of data that need to be merged or that will work together. Data cleaning solutions will involve discovering erroneous data records, correcting data and duplicate matching.

Many data cleaning solutions are highly dependent on human input. The deliverable of the profiling phase – the first phase of a data quality assessment [13], is a set of metadata describing the source data which is then used as an input for the creation of data validation and transformation rules. However, the validation rules have to be confirmed or designed by a business user who is an expert in the business area being assessed. It is not always easy or straightforward to create such a set of business rules. The situation is very similar where duplicate matching is concerned. Even if business rules for record matching are provided, e.g. "Equal SSN's and dates of birth", it may be impossible to match duplicate records, as any of the data quality issues may occur thus preventing from exact matching. Therefore, if incorrect SSN's or dates of birth stored in different positional systems occur, exact-matching business rules may not mark the records as duplicates.

When attribute standardization and correction is considered, data cleaning solutions are only as good as the reference data they use. Reference data is a set of values which are considered to be valid for a given attribute, e.g. list of states, countries, pin codes etc

The main focus of this article is to use data mining to discover data validation rules from the dataset itself and use them for attribute value correction, object-identification problem.

Recently Conditional functional dependencies (CFDs) which are an extension of Functional dependencies(FDs) have been introduced to detect data inconsistencies and provides a context dependent cleaning solutions using SQL.[1],[2].

Example 1.1: Consider an example to explain the concept of Conditional Functional Dependency. Let Customer be a relation describing customers with attributes country code (CC), area code (AC), phone number (PN), name (NM), street (STR), city (CT) and pin code (PIN). A instance r_0 of Customer is shown in Figure 1.

r_0	CC	AC	PN	NM	STR	CT	PIN
t1	91	891	1111111	Raju	B.S.Layout	VIZAG	530016
t2	91	891	1111111	Kalyani	B.S.Layout	VIZAG	530016
t3	91	40	2222222	Rajini	Hydernagar	HYDERABAD	500072
t4	91	891	2222222	Malini	M.V.P	VIZAG	530016
t5	44	131	3333333	Rupa	High St.	EDI	EH4 1DT
t6	44	131	4444444	Ravi	High St.	EDI	EH4 1DT
t7	44	891	4444444	Ravi	Port PI	VIZAG	W1B 1JH
t8	91	884	2222222	Devi	Majestic	KAKINADA	500072

Fig. 1. An instance r_0 of the Customer relation

Traditional FDs that hold on r_0 include the following:
f1: [CC, AC] → CT
f2: [CC, AC,PN] → STR .
Here f1 requires that two customers with the same country and area-codes also have the same city; similarly for f2. In contrast, the CFDs that hold on r_0 include not only the FDs f1 and f2, but also the following:
$\emptyset_0$: ([CC, PIN] → STR, (44, _|| _))
∅1: ([CC, AC] →CT, (91, 891 || VIZAG))
∅2: ([CC, AC] → CT, (44, 131 || EDI))
∅3: ([CC, AC] → CT, (91, 40 || HYDERABAD))

In $\emptyset_0$, (44,_||_) is the pattern tuple that enforces a binding of semantically related constants for attributes (CC, PIN, STR) in a tuple. It states that for customers in the UK, PIN uniquely determines STR. It is an FD that only holds on the subset of tuples with the pattern "CC = 44", rather than on the entire relation r0. CFD ∅1 assures that for any customer in the INDIA (country code 91) with area code 891, the city of the customer must be VIZAG, as enforced by its pattern tuple (91, 891 || VIZAG); similarly for ∅2 and ∅3. These cannot be expressed as FDs.

More specifically, a CFD is of the form (X→A, tp), where X→A is an FD and tp is a pattern tuple with attributes in X and A. The pattern tuple consists of constants and an unnamed variable '_ ' that matches an arbitrary value. To discover a CFD it is necessary to find not only the traditional FD X → A but also its pattern tuple tp. With the same FD X → A there are possibly multiple CFDs defined with different pattern tuples, *e.g.,* ∅1-∅3.The pattern tuple in each of ∅1-∅3, consists of only constants in both its LHS and RHS. Such CFDs are referred to as Constant CFDs.

In this paper the main focus is to discover such constant conditional functional dependencies from dataset and form a pattern tableau for each unique FD. These pattern tableau's were later merged to a single pattern tableau ,which serves as entire set of data quality rules to be validated against the dataset to detect inconsistencies.

Contribution:
In this paper 3 new algorithms are proposed for Constant CFD mining. Experimental study shows that first algorithm CCFD-FPGrowth takes less time to compute Constant CFDs but generates redundant rules that can be implied by other rules discovered with same support and confidence. Second algorithm CCFD-AprioriClose discovers reduced set of Constant CFDs by mining closed itemsets and is better than CCFD-FPGrowth, but execution time is more compared to CCFD-FPGrowth. The third CCFD-ZartMNR performs better compared to the above 2 algorithms and generates only minimal non-redundant Constant CFDs but takes more time compared to others . The rules discovered by these algorithms serves as data quality rules. To my knowledge no previous paper on Constant CFD mining has applied these algorithms.

Paper Organization:
Section 2 defines CFD, Constant CFDs. Section 3 describes Constant CFD mining and presents CCFD-FPGrowth, CCFD-AprioriClose, CCFD-ZartMNR. Section 4 describes detecting Constant CFD Violations using SQL. Section 5 discusses Experimental study and results. Section 6 explains related work and Section 7 gives conclusions.

2. DEFINITIONS

2.1 CFD

Consider a relation R defined by a set of attributes, denoted by Attribute(R). For each attribute A ϵ Attribute(R), domain(A) is used to denote its domain.
A conditional functional dependency (CFD) φ over R is a pair (X → Y, tp), where (1) X, Y are a set of attributes in Attribute(R) (2) X→ Y is a standard FD, referred to as the FD embedded in φ; and (3) tp is a pattern tuple with attributes in X and Y, where for each B in X U {Y}, tp[B] is either a constant 'a' in dom(B),or an unnamed variable '_' that draws values from dom(B). X is denoted as LHS(φ) and Y as RHS(φ). X and Y attributes in a pattern tuple are separated with ' || '. Standard FDs are a special case of CFDs. Indeed, an FD X→Y can be expressed as a CFD (X→Y, tp), where tp[B] =_ for each B in X U {Y}. The semantics of CFDs are explained in paper [3]

2.2 Constant CFD:

A CFD (X→ Y, tp) is called a Constant CFD if its pattern tuple tp consists of constants only, i.e., tp[Y] is a constant and for all B ϵ X, tp[B] is a constant. From the CFDs given in Example 1.1, ∅1,∅2,∅3 are constant CFDs.

3. CONSTANT CFD MINING

Given an instance r of a relation schema R, an algorithm for Constant CFD(CCFD) mining aims to find Constant CFDs that hold on r. Instead of mining all CCFDs that hold on r, which may contain trivial and redundant CCFDs and is unnecessarily large, only a non-trivial and non-redundant set of CCFDs are returned. A CFD φ = (X → Y, tp) over R is said to be non-*trivial* if Y ∉ X.

Problem statement:
Given an instance r of a relation schema R and a support threshold s, confidence c, the discovery problem for Constant CFDs is to find a minimal non-redundant frequent CFDs with constant patterns in r.

Frequent CFD:
The support of a CFD φ = (X → Y, tp) in r, denoted by support(φ, r) is a relative support, and is defined to be the set of tuples t in r such that t[X] = tp[X] and t[Y] = tp[Y], i.e., tuples that match the pattern of φ. Frequent CFD is one whose relative support is above user specified threshold.

3.1 CCFD-FPGrowth Algorithm:

Given a minsupport (i.e support threshold s), first FrequentPattern(FP) Growth algorithm is applied to mine frequent patterns without candidate generation. From these frequent itemsets obtained, given a user specified confidence , association rules are retrieved using Faster algorithm given by Agrawal and Srikant. These rules discovered are used as Constant CFDs

3.1.1 Frequent- Pattern Tree Approach: Mining Frequent Patterns without Candidate Generation

Frequent-pattern tree (FP-tree) is an extended prefix-tree structure for storing compressed, crucial information about frequent patterns. This FP-tree is further mined for the complete set of frequent patterns by pattern fragment growth (FP-growth) method

Efficiency of mining is achieved with : a) a large database is compressed into a condensed, smaller data structure, FP-tree which avoids costly, repeated database scans, b) FP-tree-based mining adopts a pattern-fragment growth method to avoid the costly generation of a large number of candidate sets c) a partitioning-based, divide-and-conquer method is used to decompose the mining task into a set of smaller tasks for mining confined patterns in conditional databases, which dramatically reduces the search space.

Fp-growth method is proven to be efficient and scalable for mining both long and short frequent patterns, and is about an order of magnitude faster than the Apriori algorithm.

The FP-tree construction takes exactly two scans of the transaction database: The first scan collects the set of frequent items, and the second scan constructs the FP-tree

Mining frequent patterns using FP-tree:

Here three properties are used:
a) Node-link property:- For any frequent item a_i , all the possible patterns containing only frequent items and a_i can be obtained by following a_i's node-links, starting from a_i's head in the FP-tree header.

b) Prefix path property:- To calculate the frequent patterns with suffix a_i , only the prefix subpaths of nodes labeled a_i in the FP-tree need to be accumulated, and the frequency count of every node in the prefix path should carry the same count as that in the corresponding node a_i in the path.

c) Pattern growth property:- Let m be a frequent itemset in database, B be m's conditional pattern-base, and n be an itemset in B. Then m U n is frequent in database if and only if n is frequent in B.

The procedure for construction of FP-tree and Mining frequent patterns from FP-tree by pattern fragment growth can be seen in [5] . The advantages of FP-growth over Apriori becomes evident when the dataset contains an abundant number of mixtures of short and long frequent patterns. FP-growth can mine with support threshold as low as 0.05%, with which Apriori cannot work out within reasonable time.

Given user specified Confidence c, Apriori Fast Algorithm[9] is applied on frequent itemsets discovered by FP-growth algorithm, to generate required association rules that can be used as Constant CFDs.

It is observed that rule patterns discovered here is not a minimal cover, as some rules discovered are implied by other rules having the same confidence and min support. While trying to find minimal cover, closure properties are only applied to rules with same confidence and min support.

The association rules considered here are probabilistic in nature. The presence of a rule X=>A does not necessarily mean that X,Y => A also holds because the latter may not have same minimum support as earlier. Similarly,the presence of rules X =>Y and Y=>Z does not necessarily mean that X=>Z holds because the latter may not have same minimum confidence as earlier. These such rules cannot be considered as redundant.

3.2 CCFD-AprioriClosed:

Given a minsupport (i.e support threshold s), first Apriori algorithm is applied to mine frequent closed itemsets. From these frequent closed itemsets , we can obtain all frequent itemsets. The set of all frequent closed itemsets is sufficient to determine a reduced set of association rules. Given a user specified confidence , association rules are retrieved from frequent itemsets using Faster algorithm given by Agrawal and Srikant. These rules discovered are used as Constant CFDs.

3.2.1 AprioriClosed Algorithm:

Mining frequent closed itemsets has the same power as mining the complete set of frequent itemsets, but it may substantially reduce redundant rules to be generated and increase the effectiveness of mining. This algorithm makes use of closed itemset lattice instead of subset lattice, for finding frequent itemsets.

Consider an example transaction database D in Fig.2:

TID	ITEMS
1	A C D
2	B C E
3	A B C E
4	B E
5	A B C E

Fig.2 Transaction database D

Considering minsupport=2, A closed itemset is a maximal set of items common to a set of objects. For example, in the database D, the itemset {B,C,E} is a closed itemset since it is the maximal set of items common to the objects {2,3,5}. {B,C,E} is called a frequent closed itemset as support of {B,C,E}= 3 ≥ minsupport.
The below Fig.3 gives the closed itemset lattice of D with frequent closed itemsets for minsupport=2.

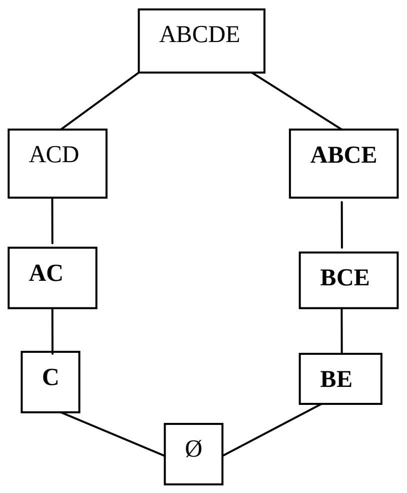

Fig.3 Closed Itemset Lattice of D

Closed itemset lattice minimizes the search space compared to subset lattice, thereby reducing the number of database passes and the CPU time involved in the generation of frequent itemsets.
Based on the closed itemset lattice properties as described in [6], we can generate all frequent itemsets from a database D through the following steps:

1. Discover all frequent closed itemsets in D, i.e itemsets that are closed and have support greater or equal to minsupport

2. Derive all frequent itemsets from the frequent closed itemsets found in step1

That is generate all subsets of the maximal frequent closed itemsets and derive their support from the frequent closed itemset supports.

This paper implemented the Pseudo code of Apriori-Close Algorithm for discovering frequent closed itemsets and deriving frequent itemsets from the frequent closed itemsets as described in [6]

Once Frequent itemsets and their support are calculated. We can generate valid association rules using Faster algorithm given by Agrawal and Srikant as below :

```
Input:
L//Large itemsets with their support   // obtained as shown in fig.4
C // user specified minimumConfidence
Output:
R  //Association Rules satisfying s and c
Steps:
R= Ø;
For each $l \in L$ do  //where L is a set of large itemsets of size ≥ 2 , l is any large itemset contained in L.
    For each $x \subset l$ such that $x \neq Ø$  do   // x is any item subset of  l
        If support(l)/support(x) ≥ C then
            R=RU {x=>(l-x)};
```
These rules obtained are considered as Constant CFDs having minsupport and confidence specified by user.

3.3 CCFD-ZartMNR

Given a minsupport (i.e support threshold s), we first apply Zart, a Multifunctional Itemset Mining Algorithm to mine frequent closed itemsets, and their associated minimal generators. From these frequent closed itemsets and minimal generators, given a user specified confidence , we retrieve set of Minimum Non Redundant(MNR)association rules. These rules discovered are used as Constant CFDs.

3.3.1 ZART Algorithm:

ZART identifies frequent closed itemsets and associate generators to their closures. This allows one to find minimal non-redundant association rules. Minimal non-redundant association rules (MNR) rules are lossless(should enable derivation of all strong rules), sound(should forbid derivation of rules that are not strong) and informative (should allow determination of rules parameters such as support and confidence). [21]

An association rule is strong if its support and confidence are not less than the user-defined thresholds minimum support and minimum confidence, respectively.

M. Kryszkiewicz has shown in [23] that minimal non-redundant rules (MNR) with the cover operator, and transitive reduction of minimal non-redundant rules (RMNR) with the cover operator and the confidence transitivity property are lossless, sound and informative representations of all strong association rules. From the definitions of MNR and RMNR it can be seen that we only need frequent closed itemsets and their generators to produce these rules.

Working steps of Zart :

1. First it identifies frequent itemsets and notes frequent generators.
2. Second,it separates frequent closed itemsets among frequent itemsets, like Apriori-Close [24]. The idea is that an itemset is not closed if it has a superset with the same support. Thus, if at the kth iteration an itemset has a subset of size $(k - 1)$ with the same support, then the subset is not a closed itemset. This way all frequent closed itemsets can be found.
3. The third step consists of relating generators to their closures. This can be done by gathering the non-closed generator subsets of the given closed itemset that have the same support.

To find frequent closed itemsets and associate generators to their closures this paper implemented the pseudo code given by algorithms 1, 2 and 3 stated in section 4 of paper [21]

MNR has the following form: the antecedent is a frequent generator, the union of the antecedent and consequent is a frequent closed itemset, and the antecedent is a proper subset of this frequent closed itemset. MNR also has a reduced subset called RMNR. Since a generator is a minimal subset of its closure with the same support, non-redundant association rules allow to deduce maximum information with a minimal hypothesis. These rules form a set of minimal non-redundant association rules, where “minimal” means “minimal antecedents and maximal consequents”. Among rules with the same support and same confidence, these rules contain the most information and these rules can be the most useful in practice [25]. For the generation of such rules the frequent closed itemsets and their associated generators are needed. Since Zart can find both, the output of Zart can be used directly to generate these rules.

The algorithm for finding MNR is the following: for each frequent generator P1 find its proper supersets P2 in the set of FCIs. Then add the rule r : P1 → P2 \ P1 to the set of MNR. The implementation for finding MNR is based on the description in [22].

4. GENERATING PATTERN TABLEAUX AND DETECTING CONSTANT CFD VIOLATIONS: (COMMON TO ALL 3 ALGORITHMS CCFD-FPGROWTH, CCFD-APRIORICLOSE, CCFD-ZARTMNR)

It is observed that association rules with 100% confidence cannot be used to clean the current dataset, but can only be used as data quality rules to clean future data. In this scenario, where we use the same dataset to generate validation rules and using these rules to correct data inconsistencies, we always have to take a confidence level <100%. The usefulness or interestingness of a rule is often application-dependent. The need for a user in the loop , to specify values for confidence and support and providing some interface to allow user guidance of the rule discovery process is always essential.

All constant patterns(AR's) discovered for each unique Functional dependency(X=>Y, where X and Y are set of attributes) are identified and consolidated to generate a single pattern tableaux T_p. Later all X attributes of pattern tableaux's are merged into a single pattern tableaux $T_{\sum}^{X}$ and all Y attributes of pattern tableaux's are merged into a single pattern tableaux $T_{\sum}^{Y}$,as described in[4], which captures all constant conditional functional dependencies.The SQL techniques to detect Single-tuple violations of cfd [4] are reused here to detect Constant CFD violations.

Problem Statement:
Given a instance r of R relation schema and a set Σ of constant CFDs on r, it is to find all the inconsistent tuples in r, i.e., the tuples that violate some constant CFD in Σ

Checking Constant CFD Violations Using SQL:
Consider the merged tableaux $T_{\sum}^{X}$ and $T_{\sum}^{Y}$ from a set Σ of Constant CFDs over a relation schema *R* and let r be an instance of *R*. Then, the following SQL query can be used to detect inconsistent tuples of r violating any of the pattern tuple.

select *t* from *R t*, $T_{\sum}^{X}$ t_p^X , $T_{\sum}^{Y}$ t_p^Y where t_p^X.id = t_p^Y.id AND $t[X1] = t_p^X[X1]$ AND . . . $t[Xn] = t_p^X[Xn]$ AND ($t[Y1] \neq t_p^Y[Y1]$ OR . . . $t[Yn] \neq t_p^Y[Yn]$).

5. Experimental Study:

In order to assess performances experimental study is conducted on the three algorithms described in section 3, for discovering constant CFDS : CCFD-FPGrowth, CCFD-AprioriClose, CCFD-ZartMNR.
Experiments studied the effects of the following factors on the scalability and the number of constant CFDs produced: (1) the support threshold s, (b) the size of a sample relation r, *i.e.,* the number of tuples in r, (3) the arity of r, *i.e.,* the number of columns in r.

5.1 Experimental Settings:

The experiments used real datasets from the UCI machine learning repository(http://archive.ics.uci.edu/ml/), namely, the Wisconsin breast cancer(WBC) and Chess datasets.
The following table describes the characteristics of these datasets:

Dataset	**Arity**	**Size (Number of tuples)**
Wisconsin breast cancer (WBC)	11	699
Chess	7	28,056

The algorithms have been implemented in Java. The program has been tested on Intel Core 2 Duo Processor (2.2GHZ) with 3GB of memory running the Microsoft WindowsXP operating system. All algorithms run entirely in main memory. Each experiment was repeated over 5 times and the average is reported here

5.2 Experimental Results:

5.2.1 Experimental Results on Wisconsin breast cancer (WBC)

The experimental results on Wisconsin breast cancer (WBC) are shown below. The number of attributes here is 11, number of tuples is 699. Here first attribute is discarded from this dataset i.e Sample code number as this is not useful in rule discovery process. Hence we consider only 10 attributes. As the domain of the attributes from 2 to 10 is a value between 1-10,and class attribute value being 2 or 4,a rule of the form 1 1 => 2 does not give information about from which attribute domain this rule derived from.

Hence for all attributes(2-11) values, a common suffix is added and converted them to string The common suffix added to Clump Thickness is -A1, for Uniformity of Cell Size is –A2, and so on for Class is –A10. Now an example of the discovered rule is of the form 1-A2 1-A6 ==> 2-A10 [support = 0.406 (284/699) confidence= 100%] which means if two tuples have same Uniformity of CellSize and Bare Nuclei values(LHS)with 1 and 1 respectively , the Class attribute value(RHS)must be equal to 2.

By keeping confidence as constant(≥70%) and varied support threshold s (10% to 60%). Figures 6, 7, 8, 9 show the number of Constant CFDS discovered by all the three algorithms for constant confidence of 70%,80%,90%,100% respectively. While figures 10,11,12,13 show the corresponding response times (in seconds)of all three algorithms to generate constant CFDs. Here response time is the sum of the time needed to generate Frequent items and the time needed to generate Association Rules from Frequent items.

From Fig.6,7,8 and 9, we observed that when support% increases from 10 to 60 , the number of Constant CFDs (CCFDs) are reduced. While when confidence% increases from 70 to 100 and support is kept constant, we also observe a reduction in number of CCFDs generated by all 3 algorithms. The algorithm CCFD-ZartMNR always produced less number of Constant CFDs, which are minimal and non-redundant. The number of CCFDs generated by CCFD-FPGrowth are always high, as it is also producing some redundant CFDs that can be inferred by other rules having same support and confidence. The algorithm CCFD-Aprioriclosed produced a reduced non-redundant CCFDs compared to CCFD- FPGrowth.

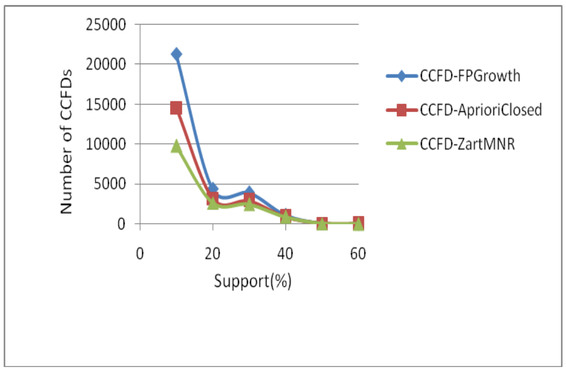

Fig .6 For constant confidence=70%

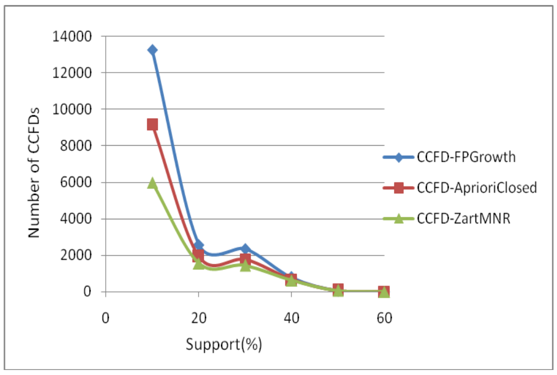

Fig .7 For constant confidence=80%

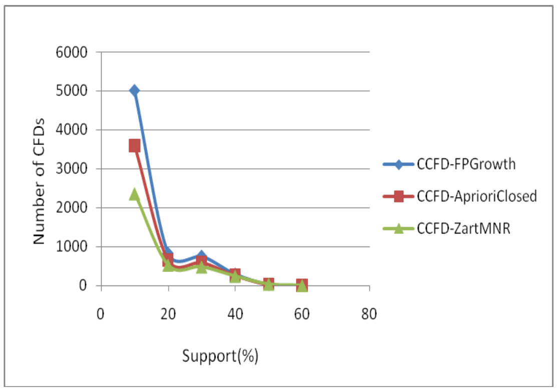

Fig .8 For constant confidence=90%

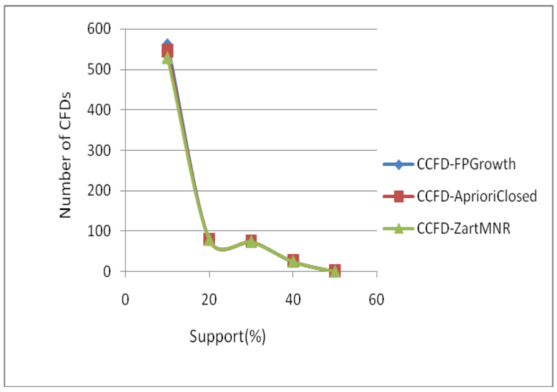

Fig .9 For constant confidence=100%

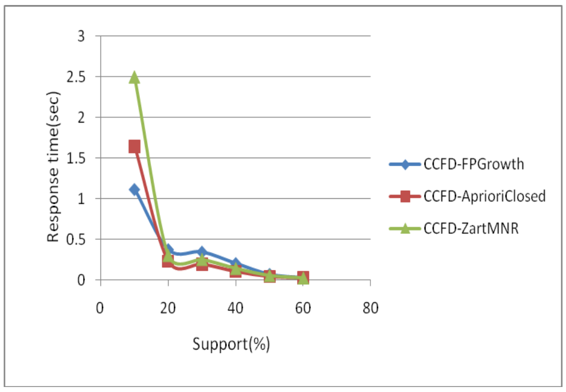

Fig .10 For constant confidence=70%

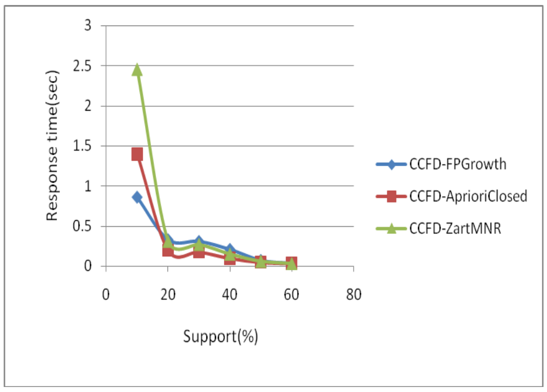

Fig .11 For constant confidence=80%

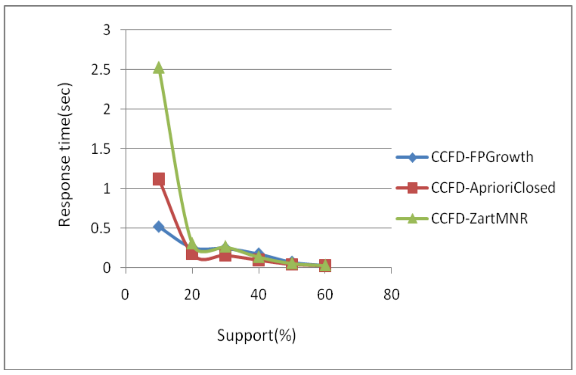

Fig .12 For constant confidence=90%

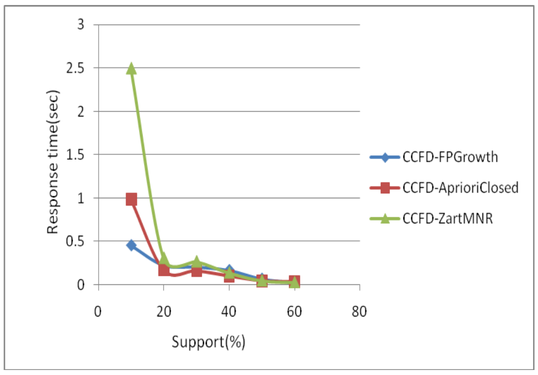

Fig .13 For constant confidence=100%

From Fig.10,11,12,and 13 we observed that when support% increases from 10 to 60 , the response time to generate CCFDs is reduced. From support%≥40 , all the four fig's show the same response time. While when confidence% increases from 70 to 100 and support is kept constant, we also observe a reduction in response time to generate CCFDs by all 3 algorithms. Among the 3 algorithms CCFD-FPgrowth took less time to generate CCFDs .This is because the time taken to compute Frequent itemsets by FPGrowth is less when compared to others. The algorithms CCFD-AprioriClosed and CCFD-ZartMNR incur an additional overhead for computing closed itemsets. Response time of CCFD-ZartMNR is always observed high, as it involves cost of computing closed itemsets and mining minimal non redundant rules.

The tuples of Wisconsin breast cancer (WBC) that violate Constant CFD. 1-A8 1-A9 ==> 1-A2 with support= 0.5(relative) and confidence= 0.82 by all three algorithms are 76 in number. First 10 tuples that violate the constant CFD are shown below in Table 3.

The pattern table for 1-A8 1-A9 ==> 1-A2 is given in Table 2.

Normal Nucleoli	Mitoses	Uniformity of Cell Size
1	1	1

Table.2 Pattern table for CCFD 1-A8 1-A9 ==> 1-A2

TUPLE ID	NORMAL NUCLEOLI	MITOSES	CELL SIZE
10	1	1	2
27	1	1	2
38	1	1	2
44	1	1	6
55	1	1	5
59	1	1	2
60	1	1	5
78	1	1	3
81	1	1	2
83	1	1	2

Table.3 Tuples which violated CCFD 1-A8 1-A9 ==> 1-A2

5.2.2 Experimental Results on Chess

In chess dataset all 7 attributes are considered. These attribute values are appended with common suffix : -A1 for first attribute,-A2 for second attribute and so on –A7 for seventh attribute.

By keeping confidence as constant(≥70%) and varied support threshold s (10% to 60%) Figures 14, 15, 16, 17 show the number of Constant CFDS discovered by all the three algorithms for constant confidence of 70%,80%,90%,100% respectively. while figures 18,19 show the response times (in seconds)of all three algorithms to generate constant CFDs for constant confidence 70%,100%.Here response time is the sum of the time needed to generate Frequent items and the time needed to generate Association Rules from Frequent items

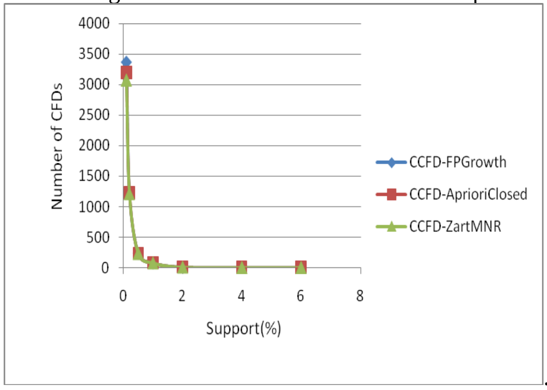

Fig .14 For constant confidence=70%

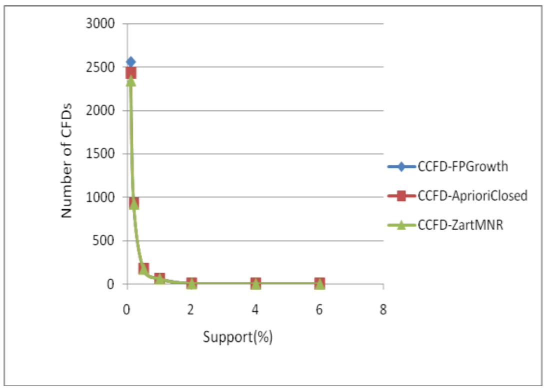

Fig .15 For constant confidence=80%

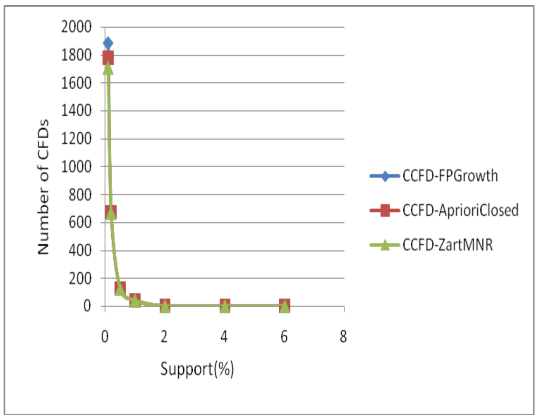

Fig .16 For constant confidence=90%

The tuples of chess dataset that violate Constant CFD fifteen-A7 ==> 1-A2 with support=0.071(relative) and confidence=0.92 by all three algorithms are 172 in number. First 10 tuples that violate the constant CFD are shown below in Table 5.

The pattern table for fifteen-A7 ==> 1-A2 is given in Table4

Optimal depth-of-win for White	White King rank
T Fifteen	1

Table.4 pattern table for CCFD fifteen-A7 ==> 1-A2

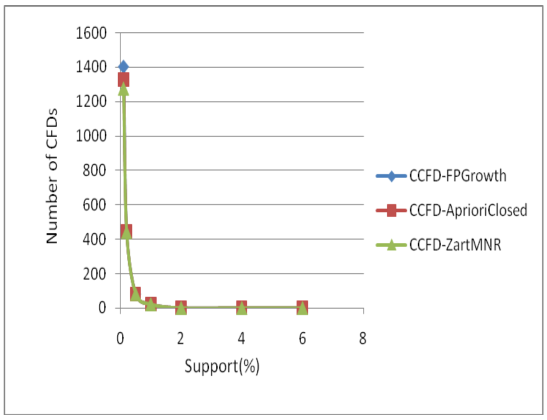

Fig .17 For constant confidence=100%

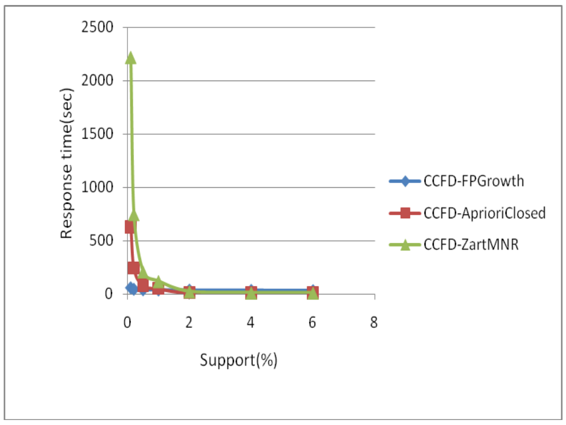

Fig .18 For constant confidence=70%

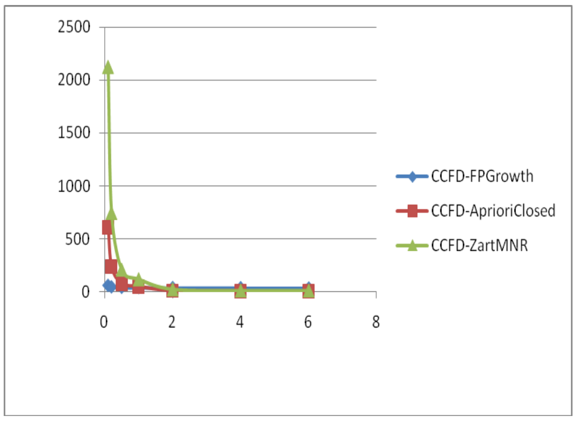

Fig .19 For constant confidence=100%

TUPLEID	Optimal depth-of-win for White	White King rank
26690	fifteen-A7	2-A2
26691	fifteen-A7	2-A2
26692	fifteen-A7	2-A2
26693	fifteen-A7	2-A2
26694	fifteen-A7	2-A2
26695	fifteen-A7	2-A2
26696	fifteen-A7	2-A2
26697	fifteen-A7	2-A2
26698	fifteen-A7	2-A2
26699	fifteen-A7	2-A2

Table.5 Tuples which violated CCFD fifteen-A7 ==> 1-A2

6. Related work:

In [1,5], CFDs has been proposed and studied mainly from a theoretical perspective, their underlying application being data cleaning. They revise classical problems (implication, consistency, axiomatization. . .) in data dependencies for CFDs.

In [12], authors study the characterization and the generation of pattern tableaux to realize the full potential of CFDs. In [14], a hierarchy of CFDs, FDs and ARs has been proposed along with some theoretical results on pattern tableaux equivalence. They use the work of [7], based on horizontal decomposition of a relation, as a way to represent and reason on CFDs.

Two important contributions have been made for CFD mining [8,3] while plenty of contributions have been proposed for FD inference and AR mining (see for example [10,11,15,16,17] in the context of this paper).

In [8], authors propose a tool for data quality management which suggests possible rules and identify conform and non-conform records. They present effective algorithms for discovering CFDs and dirty values in a data instance, but the CFDs discovered may contain redundant patterns.

In [3], authors proposed three methods to discover CFDs. The first one is called CFDMiner which mines only constant CFDs i.e. CFDs with constant patterns only. CFDMiner is based on techniques for mining closed itemsets [11]. The two other ones, called CTANE and FastCFD, were developed for general (non constant) CFDs discovery. CTANE and FastCFD are respectively extensions of well known algorithms TANE [15] and FastFD [18] for mining FDs.

7. CONCLUSIONS

In this paper all three algorithms discussed are implemented for discovering constant conditional functional dependencies(CCFDs) and studied their performance on two real world datasets.

These CCFDs are useful in data cleaning and towards enforcing semantic data consistency. Furthermore it is also shown how to identify exceptions to these rules using single-tuple violations using SQL. Of all 3 algorithms CCFD-ZartMNR proved to be more efficient as it generates only minimal non-redundant rules but execution time is slightly more when compared to others. The further study is aimed to develop algorithms which generates minimal non-redundant CCFDs with less execution time.

8. 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] Wenfei Fan , Floris Geerts , Jianzhong Li , Ming Xiong. Discovering Conditional Functional Dependencies. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL.23, NO. 5, May 2011.

[4] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for data cleaning. In Proceedings of ICDE'07, April 15-20, Istanbul, Turkey, pages 746_755, 2007.

[5] Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach Data Mining and Knowledge Discovery Journal , 8, 53–87, 2004

[6] Discovering Frequent Closed Itemsets for Association Rules, Nicolas Pasquier, Yves Bastide, Rafik Taouil, Lotfi Lakhal. ICDT '99 Proceedings of the 7th International Conference on Database Theory, Pages 398 - 416 , Springer-Verlag London, UK ©1999

[7] Paul De Bra and Jan Paredaens. Conditional dependencies for horizontal decompositions. In Proceedings of the 10th Colloquium on Automata, Languages and Programming, pages 67_82, London, UK, 1983. Springer-Verlag.

[8] Fei Chiang and Renée J. Miller. Discovering data quality rules. PVLDB,1(1):1166_1177, 2008.

[9] Fast Algorithms for Mining Association Rules, Rakesh Agrawal, RamaKrishnan Srikant, In Proc. 20th Int. Conf. Very Large Data Bases, VLDB (December--JanuaryMay~ 1994), pp. 487-499

[10] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993, pages 207_216. ACM Press, 1993.

[11] Nicolas Pasquier, Yves Bastide, Ra_k Taouil, and Lot_ Lakhal. Discovering frequent closed itemsets for association rules. In ICDT, pages 398_416, 1999.

[12] Lukasz Golab, Howard Karlo_, Flip Korn, Divesh Srivastava, and Bei Yu.On generating near-optimal tableaux for conditional functional dependencies.Proc. VLDB Endow., 1(1):376_390, 2008.

[13] A. Maydanchik *Data Quality Assessment* Technics Publications,336pp, ISBN-13: 9780977140022

[14] Raoul Medina and Lhouari Nourine. A unified hierarchy for functional dependencies, conditional functional dependencies and association rules. In ICFCA, Lecture Notes in Computer Science, pages 235_248. Springer, 2009

[15] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. The Computer Journal, 42(2):100_111, 1999.

[16] Stéphane Lopes, Jean-Marc Petit, and Lot_ Lakhal. Efficient discovery of functional dependencies and armstrong relations. In EDBT 2000, volume 1777 of LNCS, pages 350_364, Konstanz, Germany, 2000. Springer.

[17] Noël Novelli and Rosine Cicchetti. Fun: An efficient algorithm for mining functional and embeddeddependencies. In Proceedings of the 8th International Conference on DatabaseTheory (ICDT'01), volume 1973 of Lecture Notes in Computer Science, pages 189_203, 2001.

[18] Catharine Wyss, Chris Giannella, and Edward Robertson. Fastfds: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances extended abstract. Data Warehousing and Knowledge Discovery, pages 101_110, 2001.

[19] G.Birkhoff. Lattices theory. In coll.pub. XXV, Volume 25. American Mathematical Society,1967. Third edition.

[20] B.A. Davey and H.A.Priestley. Introduction to Lattices and Order. Cambridge University Press,1994. Fourth edition.

[21] L. Szathmary, A. Napoli, and S. O. Kuznetsov. ZART: A Multifunctional Itemset Mining Algorithm. In Proc. of the 5th Intl. Conf. on Concept Lattices and Their Applications (CLA '07), pages 26{37, Montpellier, France, Oct 2007

[22] Marzena Kryszkiewicz: Representative Association Rules and Minimum Condition Maximum Consequence Association Rules. PKDD 1998: 361-369

[23] Kryszkiewicz, M. Concise representations of association rules. In: Pattern Detection and Discovery. (2002) 92–109

[24] Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Closed set based discovery of small covers for association rules. In: Proc. 15emes Journees Bases de Donnees Avancees, BDA. (1999) 361–381

[25] Pasquier, N.: Mining association rules using formal concept analysis. In: Proc. Of the 8th International Conf. on Conceptual Structures (ICCS ’00), Shaker-Verlag (2000) 259–264