Future Computing and Informatics Journal

Volume 1
Issue 1 (2016) *2016, Issues 1–2*

Article 2

2016

Fixing rules for data cleaning based on conditional functional dependency

Asmaa Abdo
Faculty of Computers and Information, Menoufia University, Egypt,
engasmaa.saad.mis@ci.menofia.edu.eg

Available online at www.sciencedirect.com

ScienceDirect

Future Computing and Informatics Journal 1 (2016) 10–26
http://www.journals.elsevier.com/future-computing-and-informatics-journal/

Fixing rules for data cleaning based on conditional functional dependency

Rashed Salem, Asmaa Abdo*

Information Systems Department, Faculty of Computers and Information, Menoufia University, Egypt

Received 3 February 2017; revised 14 March 2017; accepted 16 March 2017
Available online 6 April 2017

Abstract

Most existing databases suffer from data inconsistencies. Enhancing data quality efforts are necessary to resolve this issue. In this paper, two techniques are proposed for mining accurate conditional functional dependencies rules from such databases to be employed for data cleaning. The idea of the proposed techniques is to mine firstly maximal closed frequent patterns, then mine the dependable conditional functional dependencies rules with the help of lift measure. Moreover, data repairing algorithm is proposed for fixing inconsistent tuples found in the database exploiting the generated rules. An extensive experimental is conducted study to confirm the effectiveness of the proposed techniques compared with existing technique on both real-life and synthetic medical data sets.

Keywords: Data quality; Data repairing; Data mining; Data inconsistency; Data cleaning

1. Introduction

Organizations today are facing increasing challenges in growing of their data. Ensuring data quality is a critical challenge in most of currently application domains. Like most public and private organizations have a significant aspect of their data, which are used for data management purposes. Henceforward, the existence of inconsistent issues data decreases their assessment making them misinformed or even harmful [26].

The value of data depends on its quality that is more difficult to be defined. Unlike manufactured products, data do not have physical characteristics that allow quality to be easily assessed. Quality is thus a function of intangible properties. In this context, quality means that data is "fitness for use" or "potential for use". Fitness for use involves not only the statistical quality concepts of variance and bias, but also other characteristics such as consistency and de-duplication that determine how effectively statistical information are used [23]. Data quality is an essential characteristic that determines the reliability of data for decision making in organizations. The quality of data is an increasingly pervasive problem, as data in real world databases quickly degenerates over time and affects the results of the mining [15,32].

Dirty data costs US business billions of dollars annually, which lead to poor decisions making [7]. Fig. 1 indicates a connection between costs inflicted by poor data quality and costs of ensuring high-quality of data. There is a tradeoff between costs inflicted by poor data quality and costs of assuring data quality. Increasing poor quality of data affects negatively in resulting high-quality data that is lead to think about how to ensure and achieve the quality of data.

Therefore, it is important to focus on detecting and repairing data inconsistency problems as data cleaning process. Specifically, guaranteeing high-quality dependable data is a competitive advantage for all industries, which necessitates accurate data scrubbing solutions.

The term data cleaning (or data scrubbing) is considered the critical phase of data preprocessing. It really works for

* Corresponding author.
E-mail address: engasmaa.saad.mis@ci.menofia.edu.eg (A. Abdo).
Peer review under responsibility of Faculty of Computers and Information Technology, Future University in Egypt.

http://dx.doi.org/10.1016/j.fcij.2017.03.002

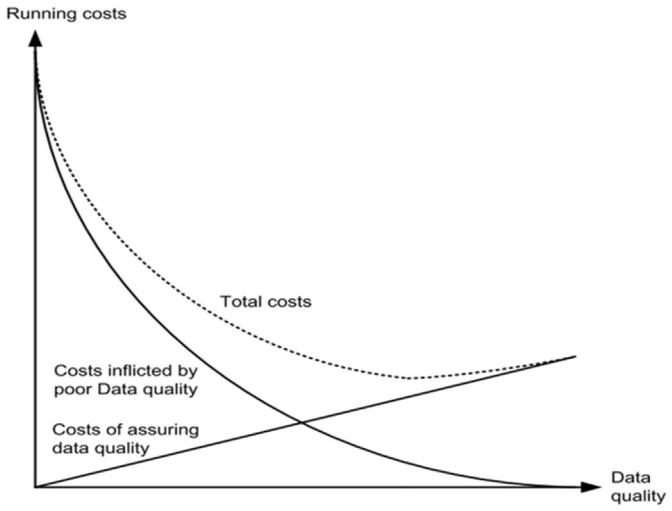

Fig. 1. Total costs acquired by quality of data in the organization [13].

maintaining inconsistent records before mining and examining data [26]. Data cleaning is essentially found in several applications such as data warehousing, data quality management and knowledge discovery in databases.

Data mining is an area of research which means the process of finding hidden and unknown patterns in databases and using this fact to build models. Data mining provides methods and technologies to transform huge amounts of data to be processed and analyzed into useful information for decision making purposes [29]. Data mining is becoming more popular and essential topic in current data cleaning algorithms, because of manual data cleansing is also exhausting process, time-consuming, and itself prone to errors [21].

Undoubtedly, the majority of data cleaning techniques in the literature apply record matching [10], which compare probably erroneous records with master cleaned records. It is still mandatory to cope with data inconsistency issue exploiting the data themselves without the necessity for master data. While fixing data inconsistencies, ensuring integrity constraints is performed including Functional Dependencies (FDs) and, Conditional Functional Dependencies (CFDs) [4,8]. Identifying inconsistent values is a major step in the data cleaning process.

Currently, growing data in most application domains considers a significant challenge for ensuring consistent and accurate data quality, to be employed for data management purposes. The incorrect data recorded electronically leads to poor quality data. Herein, we are interested in generating dependable accurate data quality rules, which then used for repairing data inconsistencies in several application domains.

Contributions. The main contributions of this paper involve proposing techniques for enhancing data cleaning process. More specifically, paper contributions include:

I. Proposing ICCFD_Miner technique for generating dependable data cleaning rules based mainly on frequent closed patterns.
II. Proposing MICCFD_Miner technique for enhancing the performance of first proposed ICCFD_Miner technique in generating data cleaning rules. The enhancement is realized on extracting maximal frequent patterns and their associated generators instead of using closed frequent patterns. It acts as effective search strategy mechanism to reduce the size of the search space domain.
III. Proposing T_Repair algorithm for given generated rules and input tuples (t) of relation for validating repairing errors from given dataset.

Organization. This paper is organized as follows: Section 2 discusses related work. In Section 3, background and definitions of data quality concepts are presented. Then, Section 4 highlights data cleaning in medical applications. Section 5 presents the proposed ICCFD_Miner, MICCFD_Miner, and T_Repair techniques. Section 6 discusses the experimental study and results conducted for different medical datasets. Finally, Section 7 concludes the proposed work and highlights the future trends.

2. Related work

Unfortunately, despite the urgent need for precise and dependable techniques for enhancing data quality and data cleaning problems, there is no vital solution up to now to these problems. There has been little discussion and analysis about enhancing data consistency. However, most of the recent work focus on record matching and duplicate detection [2].

Database and data quality researchers have discussed a variety of integrity constraints based on Functional Dependencies (FD) [5,12,20,35]. In Ref. [35], authors propose an FD_Mine algorithm that discovers functional dependency from given relation. A survey and comprehensive comparison of seven algorithms for discovering functional dependencies are discussed in Ref. [27]. Surveyed algorithms include TANE, FUN, FD_Mine, DFD, Dep-Miner, FastFDs, FDEP as indicated extensively in Ref. [27]. Nevertheless, traditional FDs are developed mainly for schema design but are often not able to detect the semantic values errors of data. Other researchers focus on the extension of FD, they have proposed what is so-called Conditional Functional Dependencies (CFD) and Conditional Inclusion Dependencies (CID) for capturing errors in data. Algorithms that proposed for discovering CFDs rules from relation include: CFD Miner algorithm for discovering constant conditional functional dependencies, CTANE algorithm that extends TANE to discover general CFDs, and FastCFD for discovering general CFDs by employing a depth-first search strategy instead of the level-wise approach as used in CTANE algorithm [6].

Several data quality techniques are proposed to clean messy tuples from databases [9], as researchers aim to find critical information missing from databases. In Ref. [9], authors propose three models to specify relative information completeness of databases from which both tuples and values may be missing. Statistical inference approaches are studied in Ref. [24], which

infer missing information and correct errors automatically. These approaches tackle missing values to enhance the quality of data. From technological part, several open source tools are developed for handling messy data. Open Refine and Data Wrangler are two open source tools for working with missing data for cleaning them as detailed in Ref. [18].

Moreover, there are a variety of data transformation methods such as commercial ETL (Extract, Transformation, and Loading) tools [31]. Extraction methods focus on extracting data from homogeneous and/or heterogeneous data sources. Transformation methods purpose is to store data in proper format or structure for querying and analysis purpose. Loading methods concern with load data into a single data source repository such data warehouse or other unified data source depending on the requirements of the organization. These tools are developed for data cleaning to support any changes in the structure, representation or content of data.

The usage of editing rules in combination with master data is discussed in Ref. [8]. Such rules can find certain fixes by updating input tuples with master data. According to constraints, editing rules have dynamic semantics and are relative to master data. Given an input tuple t that matches a pattern, editing rules tell us which attributes of given tuple t should be updated and what values from master data should be assigned to them. This approach requires defining editing rules manually for both relations, *i.e.*, master relation and input relation, which is very expensive and time-consuming. Repairing use heuristic solution based on minimum cost function of two updates that not always provide with a deterministic fix. Editing rules require users to examine every tuple, which is expensive.

Furthermore, lots of work are proposed in the literature relying on domain specific similarity and matching operators, such works include record matching, record linkage, duplicate detection, and merge purge [2,7,14]. These approaches define two functions; namely, match and merge [1]. While match function identifies duplication of records, the merge function combines the two duplicated records into one.

Finally, it is concluded that existing methods do not guarantee deterministic and reliable fixes to the inconsistency problem. Such methods do not work properly when detecting errors in critical data such as Electronic Medical Records (EMRs) of healthcare systems. While most exiting works depend on either the availability of cleaned master data to match and fix data inconsistencies or conditional functional dependencies for capturing errors in data, the traditional functional dependencies based techniques can be extended to generate more dependable and accurate fixing rules from data themselves without need to master data. In this paper, the constant CFD is reutilized as a special case of association rules [6,17], and is exploited to resolve the problem of detecting and repairing inconsistencies errors from databases.

3. Preliminary

There are several terminologies from the literature which are related to the proposed techniques. The definitions of such terminologies are provided, including functional dependencies, conditional functional dependencies, closed frequent patterns, maximal frequent patterns, constant CFD problem, and pruning search space.

Functional Dependencies (FD) is defined as a constraint between two sets of attributes in relation from a database. FD over Relation(r) is represented as $X \rightarrow Y$ where each X value is associated with precisely one Y value. Herein, X is considered as the determinant and Y is considered as the dependent. FD describes a relationship between all possible combinations of attribute value pairs, and employed for working on schema design that do not capture semantic of data for data cleaning [35].

Conditional Functional Dependencies (CFD) is an extension of FDs, which aims to detect inconsistencies of data between tuples in a single relation. The CFD φ on relation r *is* a pair $(X \rightarrow Y, tp)$, where $X \rightarrow Y$ is a standard FD on r; and tp is the pattern tuple of φ with attributes in X and Y. For each attribute A in $X \cup Y$, $tp[A]$ is either a constant in the domain of A, or an unnamed variable '_' [10].

In this example, records from customer relationships table are addressed, such records having the following attributes Country Code (CC), Area Code (AC), Phone Number (PN), Name (NM), Street (STR), City (CT) and Zip Code (ZIP) as shown in Table 1. The purpose of this example is to explain difference between FDs and CFDs rules.

Traditional FDs that holds on Table 1 is as the following:

$$f_1 : [\mathrm{CC}, \mathrm{AC}] \rightarrow \mathrm{CT}$$

$$f_2 : [\mathrm{CC}, \mathrm{AC}, \mathrm{PN}] \rightarrow \mathrm{STR}$$

f_1 mean that if two customers have the same country and area code, they also have the same city, similarly for f_2.

The following are CFDs that hold on Table 1 is

$$\varphi_0 : ([\mathrm{CC}, \mathrm{ZIP}] \rightarrow \mathrm{STR}, (32, -||-))$$

$$\varphi_1 : ([\mathrm{CC}, \mathrm{AC}] \rightarrow \mathrm{CT}, (40, 872||\mathrm{UN}))$$

$$\varphi_2 : ([\mathrm{CC}, \mathrm{AC}] \rightarrow \mathrm{CT}, (32, 222||\mathrm{VIZAG}))$$

$$\varphi_3 : ([\mathrm{CC}, \mathrm{AC}] \rightarrow \mathrm{CT}, (40, 101||\mathrm{EDI}))$$

Rules are classified to φ_0 as variable CFD, where φ_1, φ_2, φ_3 are constant CFD. In this work, we are concerned with such type of constant CFD. CFD φ_0 asserts that zip uniquely determines STR. This FD holds on a subset of tuples with the

Table 1
Sample records from customer relation.

	CC	AC	PN	NM	STR	CT	ZIP
t_1	40	872	6666666	Mike	Port PI	UN	08422
t_2	40	872	6666666	Rick	Port PI	UN	08422
t_3	40	101	9999999	Joe	M.V.P	EDI	01202
t_4	40	872	9999999	Jim	Majestic	UN	08422
t_5	32	222	4444444	Ben	6rd Str	VIZAG	EH4 1DT
t_6	32	222	7777777	Ian	6rd Str	VIZAG	EH4 1DT
t_7	32	872	7777777	Ian	XXX	UN	W1B 1JH
t_8	40	131	9999999	Sean	5rd Str	MH	01332

pattern CC = 32, rather than on the entire relation. CFD φ_1 assures that for any tuple if customer's country code = 40 and area code = 872, then the customer's city is UN, similarly for φ_2 and φ_3. CFDs φ_0, φ_1, φ_2, φ_3 indicate that these rules cannot be able to be represented by FDs.

Closed Frequent Patterns, Pattern is frequent closed if it is not included in a proper superset having the same support. A generator Y of a frequently closed pattern X is a pattern constraint with having the same support as X, and it does not have any subset having the same support. The set of closed frequent patterns is lossless, which it contains uniquely the complete information regarding its corresponding frequent patterns [6]. From the set of closed frequent patterns, it is straightforward to derive both the identities and supports of all frequent patterns without mining the database again. At the same time, closed frequent patterns can themselves be orders of magnitude smaller than all frequent patterns, especially on dense databases.

For example, from Table 1 ([CC, AC, CT, ZIP], (40, 827, UN, 08422)) is a closed pattern set with support equal to 3. This closed pattern has two generators patterns, ([CC, AC], (40, 827)) and ([ZIP], (08422)), both having support equal to 3 as well.

Maximal Frequent Patterns, are called maximal because they have no frequent supersets. In other words, patterns are maximal frequent if none of its immediate supersets is frequent. Mining maximal frequent pattern help discovers the long patterns in dense databases. Extracting maximal frequent patterns has become an important issue because the set of maximal frequent patterns not only uniquely define patterns, but also the number of maximal frequent patterns can be significantly smaller than the number of closed frequent patterns. The set of maximal frequent patterns is thus a subset of the set of frequent closed patterns, which is a subset of all frequent patterns. This patterns extracted by applying effective search mechanism to reduce the size of search space domain. Moreover, the set of these patterns is a minimal set, i.e., the smallest set from which all frequent patterns [28].

Constant CFD Problem is the problem of discovering a minimal set of frequent constant CFDs that include non-redundant CFD. This also mean to discover conditional functional dependencies with constant patterns only [19,30].

Pruning the Search Space: search space is still a major problem in the area of pattern extraction from the dataset in data mining research. Thus, the pruning technique from machine learning techniques is necessary to reduce the size of search space domain. The objective of pruning is to remove infrequent nodes from search space using predefined support threshold value. The pruned search space is significantly smaller than the unpruned search space. Pruning used to extract constant conditional functional dependencies from all patterns in the dataset [19].

4. Data cleaning in healthcare: motivation

The medical domain is one from several applications in information systems which involves dirty and inconsistent data. The healthcare data management systems should ensure the high-quality data of patient records, in particular critical decisions are based on such records. The incorrect data recorded electronically about patients may lead to wrong treatment and prescriptions, which may cause several problems including death [22,29]. Fig. 2 indicates the consequence of poor data quality on Electronic Medical Records (EMRs).

Electronic Medical Records (EMRs) are one of the major automated concern of current hospitals [3]. EMRs need to be ensured against quality to obtain user satisfaction that effect on the resulting of the overall system effectiveness. User satisfaction about his electronic medical data is considered as one of the measures to ensure quality in EMRs. In Fig. 3, a schematic diagram for data cleaning process in electronic medical records is presented.

Furthermore, healthcare stakeholders and services providers need to acquire the high quality of medical data for both delivering medical services as well as treating clinical decisions to improve healthcare quality. They need such data with accurate quality to take the full advantage of the decision-making process. In particular, acquiring accurate and dependable information about diseases treatment is based on precise and consistent data stored about patient [11,33]. Therefore, healthcare service research aimed at precise, complete and consistent statistical information for practicing the healthcare services within a society.

Example 1: Consider a sample of data from the thyroid dataset as shown in Table 2. Such data involves several inconsistencies from medical context. The proposed techniques aim to discover some dependable data quality rules for both detecting and fixing such inconsistencies.

5. Proposed techniques

There are several drawbacks in the traditional techniques, e.g., traditional FDs, which developed mainly for schema design. Some other techniques produce redundant non-dependable rules and not scalable in large datasets with a large number of attributes. Moreover, other techniques require users involving in the cleaning process, which is expensive and

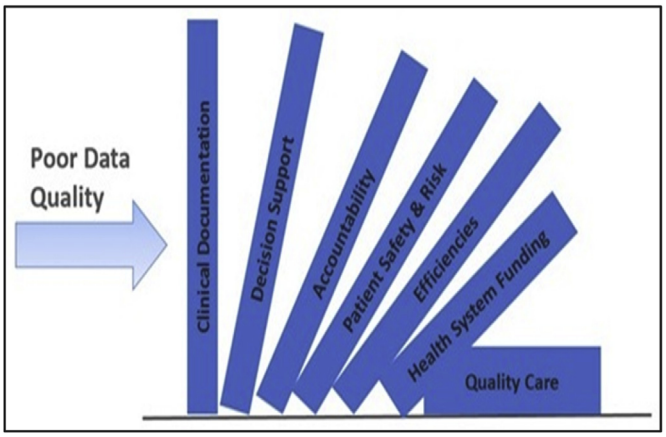

Fig. 2. Poor data quality in electronic healthcare.

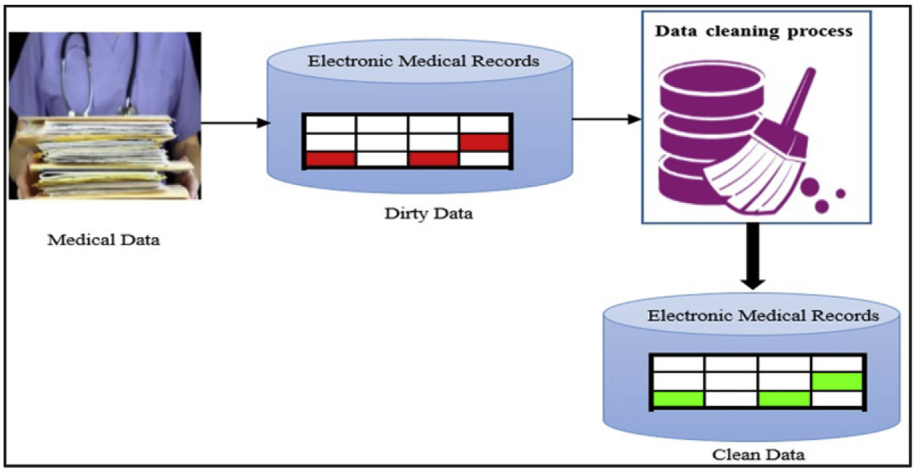

Fig. 3. Schematic diagram for data cleaning process in electronic medical records.

Table 2
Sample records from Thyroid dataset.

Patient number	Pregnant	Hypopituitary	Antithyroid medication	TBG measured
98	FALSE	FALSE	TRUE	FALSE
162	FALSE	FALSE	FALSE	FALSE
164	FALSE	FALSE	FALSE	FALSE
175	FALSE	FALSE	TRUE	FALSE
183	FALSE	FALSE	FALSE	FALSE
214	FALSE	FALSE	TRUE	FALSE
218	FALSE	FALSE	FALSE	FALSE
261	FALSE	FALSE	FALSE	FALSE
742	FALSE	FALSE	TRUE	FALSE
1253	FALSE	FALSE	FALSE	FALSE

is not suitable. The proposed techniques are designed to overcome the drawbacks of the traditional techniques and achieve the quality of data in a variety of application domains.

In this section, the proposed techniques for enhancing and ensuring the quality of data are presented. The three techniques, i.e., ICCFD-Miner, MICCFD-Miner and T-Repair, are discussed respectively.

5.1. The proposed ICCFD_Miner technique

The first proposed technique is called Interest Constant Conditional Functional Dependencies technique abbreviated as ICCFD_Miner. It is proposed for discovering interest minimal non-redundant constant CFD Rules. The ICCFD_-Miner relies on generating frequent closed patterns as well as their associated generators consequently. The associated generators can be defined as the closure of frequent closed patterns. The ICCFD_Miner focuses on generating precise, dependable, interest minimal and non-redundant constant CFD data quality rules, which cover all set of rules with respect to identified thresholds of the support and confidence.

Let us assume that there are a tuple t of a relation R, minimum support, and minimum confidence thresholds, the ICCFD_Miner enable discovering Interest-based Constant Conditional Functional Dependencies (ICCFDs) rules. The discovered ICCFDs certify finding interest minimal non-redundant dependable Constant Conditional Functional Dependencies (CCFDs) rules with constant patterns in t. Fig. 4 shows the pseudo-code of the algorithm which detailed as:

Input: The inputs to the ICCFD-Miner technique are both dataset and predefined thresholds, *i.e.*, minimum support and minimum.

Step 1: The user-defined minimum support threshold is used to generate the list of frequent closed patterns and their associated generators according to their closure. The ICCFD_Miner utilizes closed frequent patterns and their associated generators rather than all frequent patterns to minimize the search space and to save the time of rules generation [17]. Fig. 5 shows the search space domain.

The *Support CFD* φ: $(X \rightarrow Y)$ is defined as the number of records in the dataset that contain both patterns X and Y to the total number of records in the database T. The greater values of Support validate the correlation among patterns. Support of a CFD φ: $(X \rightarrow Y)$ whereas X is generator pattern and Y is closed patterns [4], defined as

$$\text{support}(X \rightarrow Y) = \frac{\text{support}(X \cup Y)}{T} \quad (1)$$

Step 2: The user defined confidence threshold is used the set of minimum interest non-redundant constant conditional functional dependencies data quality rules. The support and confidence are utilized for generating these rules in the literature work, however the ICCFD_Miner utilizes not only support and confidence measures but also the interest (or so-called lift) measure to generate more reliable and dependable rules. The form of rules for each frequent generator pattern X finds its proper supersets Y from a set of frequent closed patterns.

The *Confidence CFD* is defined as the number of records in the dataset that satisfy CFD and contain the generator and consequence patterns divided by a number of records that satisfy left-hand side of the rule.

$$\text{confidence}(X \rightarrow Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X)} \quad (2)$$

Pseudo Code of Proposed ICCFD_Miner

Input: Dataset (D), Thresholds (Support, Confidence).
Output: ICCFD_Miner Rules.

```
1. for each v_i[n]∈ D
2.     Supp_Calc ← Compute_SupportCount (v_i[n]);
3.         If (Support_Count (v_i[n]) ≥ Support)
4.             Then C ← Compute_Closed (v_i[n]);
5.              G_C ← Find_Generator (C);
6.              Conf_Calc ← Compute_Confidence (C, C/G_C);
7.              Lift_Calc ← Compute_Lift (C, C/G_C);
8.                 If (Conf_Calc ≥ Confidence && Lift_Calc > 1)
9.                     Then Generate (ICCFD_Miner Rules);
10.                    else ;
11.                    return ICCFD_Miner Rules;
12.                end if
13.             else ;
14.         end if
15. end for each
```

Fig. 4. First proposed ICCFD-Miner algorithm.

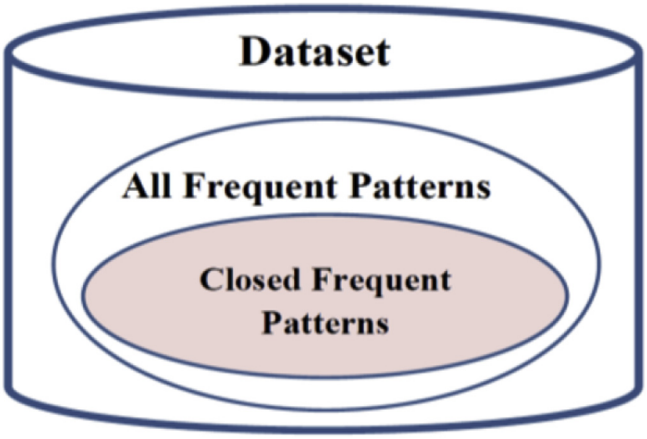

Fig. 5. Search space domain of proposed ICCFD_Miner technique.

The reliability of the rule is measured using confidence whereas the value is a real number between 0 and 1.0 [25]. The shortcoming of the confidence measure is that it neglects the support of the right-hand side of rules. Consequently, the data quality measure called Lift (Interest) is utilized to generate more dependent rules when defined it as greater than 1.

The *Lift CFD* is defined as measuring the degree of compatibility of the left-hand side and right-hand side of rules [16]. The Lift value is configured greater than 1 to ensure generating dependent rules. The lift of the CFD rule φ: (X → Y) is computed as:

$$\text{lift}(X \rightarrow Y) = \frac{\text{confidence}(\varphi)}{\text{support}(Y)} \quad (3)$$

Finally, the ICCFD_Miner optimizes the rules generation process compared with the most related methods.

ICCFD_Miner algorithm. The algorithm ICCFD_Miner generates the desired rules given dataset, minimum support, and minimum confidence thresholds as shown Fig. 4. It starts by iterating over each field value $v_i[n]$ in the dataset to compute the frequency of each field value according to Equation (1). Then, it checks if the calculated frequency exceeds the given support threshold, it computes the closed frequent patterns of such values. The generated closed frequent patterns are stored in *C* hash map as key (line 4). Moreover, the generators of closed frequent patterns are computed and stored in *G_C* of the hash map as values (line 5). Both confidence and lift measures of (C, G_C) are calculated in line 6 and 7, respectively according to Equations (2) and (3). The algorithm checks if the calculated confidence exceeds the given threshold and the calculated lift greater than 1, then ICCFD_Miner rules are generated and returned to the repository. The generated rules follow the form *G_C* as the left-hand side of the rule (antecedent) and *C/G_C* as right-hand side of rule (consequence).

Now, let us addressing an example of generated rule form from the dataset utilized in Section 4 as follows:

$$\varphi: \{\textbf{Pregnant} = \textbf{FALSE} \wedge \textbf{Hypopituitary} = \textbf{FALSE}\} \rightarrow \{\textbf{Antithyroid\_Medication} = \textbf{FALSE} \wedge \textbf{TBG\_Measured} = \textbf{FALSE}\}$$
$$\textbf{with min} - \textbf{support} = 0.98,\ \textbf{min} - \textbf{confidence} = 0.99,\ \textbf{and lift} > 1.$$

"If she is not pregnant and not has hypopituitary then it must not take any antithyroid medication and its TBG measured value should be false".

The mentioned rule is a important and well-known rule to physicians when pre-scribing drug to the patient. Furthermore, such rules can be employed statistically to determine a total number of patients that deal antithyroid medication. Table 3 shows the set of tuples that violate the mentioned generated rule, whereas error values are shaded.

5.2. The proposed MICCFD_Miner technique

The second proposed technique is called Minimal Interest Constant Conditional Functional Dependencies Technique abbreviated as MICCFD_Miner. Since the overall performance of rules generation is based on the fast extraction of patterns from data. MICCFD_Miner mainly based on mining maximal frequent patterns and their associated generators, applying the effective pruning mechanism to reduce the size of search space domain.

Maximal frequent patterns (MFPs) are usually much smaller than the set of frequent patterns and also smaller than a set of frequent closed patterns. In other words, maximal frequent patterns are subset of closed frequent patterns that are subset of frequent patterns. The MFP patterns are called maximal because they have no frequent supersets. Moreover, the set of these patterns is a minimal set, i.e., the smallest set from which all frequent patterns.

Moreover, the MICCFD_Miner employs Interest (lift) measure as a filter to generate accurate, and dependable interest data quality rules. It also prunes other rules that are neither statistically significant nor meaningful. The generated output MICCFD rules are utilized for both detecting and repairing data inconsistencies from large datasets. The MICCFD-Miner technique is depicted in pseudo-code presented in Fig. 6.

Table 3
Records that violate the generated rule.

Patient number	Pregnant	Hypopituitary	Antithyroid medication	TBG measured
98	FALSE	FALSE	TRUE	FALSE
162	FALSE	FALSE	FALSE	FALSE
164	FALSE	FALSE	FALSE	FALSE
175	FALSE	FALSE	TRUE	FALSE
183	FALSE	FALSE	FALSE	FALSE
214	FALSE	FALSE	TRUE	FALSE
218	FALSE	FALSE	FALSE	FALSE
261	FALSE	FALSE	FALSE	FALSE
742	FALSE	FALSE	TRUE	FALSE
1253	FALSE	FALSE	FALSE	FALSE

Besides the dataset both the minimum support and the minimum confidence are the main input of the MICCFD-Miner. Subsequentially, the first step using the support threshold to extracte a list of maximal frequent patterns and their associated generators. The MICCFD-Miner technique then utilizes such extracted maximal frequent patterns instead of working on discovering all frequent patterns or frequent closed patterns, reducing the size of search space domain as shown in Fig. 7.

The second step depends on extracted patterns and their associated generators from the first step. Then, the confidence threshold is utilized to generate the set of minimum interest non-redundant constant CFD data quality rules. The form of rules for each frequent generator pattern X finds its proper supersets A from a set of maximal frequent patterns. Then, from X and A add rule antecedent (Generator) $\rightarrow$ consequence (maximal/generator) as φ: X $\rightarrow$A. Finally, generated rules are output to repository MICCFD-Miner rules.

MICCFD_Miner algorithm. The algorithm MICCFD_-Miner generates the desired rules given dataset, support, and confidence thresholds as shown Fig. 6. It starts by iterating over each field value $v_i[n]$ in the dataset to compute the frequency of each field value according to Equation (1). Then, it checks if the calculated frequency exceeds the given support threshold, it computes the maximal frequent patterns of such values. The generated maximal frequent patterns are stored in M hash map as key (line 4). Moreover, the generators of maximal frequent patterns are computed and stored in G_M of the hash map as values (line 5). Both confidence and lift measures of (M, G_M) are calculated in line 6 and 7, respectively according to Equations (2) and (3). The algorithm checks if the calculated confidence exceeds the given threshold and the calculated lift greater than 1, then MICCFD_Miner rules are generated and returned to the repository. The generated rules follow the form G_M as the left-hand side of the rule (antecedent) and M/G_M as the right-hand side of the rule (consequence).

Discussion: The ICCFD_Miner technique and MICCFD_-Miner technique are mainly distinguished in step 1. The MICCFD_Miner technique is considered as enhancement of the ICCFD_Miner in memory space complexity measurement.

5.3. The proposed T_Repair algorithm

Since the most recent repairing techniques fix data manually, there is a need to fix detected errors and inconsistent data

Pseudo Code of Proposed MICCFD_Miner

Input: Dataset (D), Thresholds (Support, Confidence).
Output: MICCFD_Miner Rules.

```
1. for each v_i[n]∈ D
2.     Supp_Calc ← Compute_SupportCount (v_i[n]);
3.         If (Support_Count (v_i[n]) ≥ Support)
4.             Then M ← Compute_Maximal (v_i[n]);
5.             G_M ← Find_Generator (M);
6.             Conf_Calc ← Compute_Confidence (M, M/G_M);
7.             Lift_Calc ← Compute_Lift (M, M/G_M);
8.                 If (Conf_Calc ≥ Confidence && Lift_Calc > 1)
9.                     Then Generate (MICCFD_Miner Rules);
10.                    else ;
11.                    return MICCFD_Miner Rules;
12.                end if
13.            else ;
14.        end if
15. end for each
```

Fig. 6. Second proposed MICCFD-Miner algorithm.

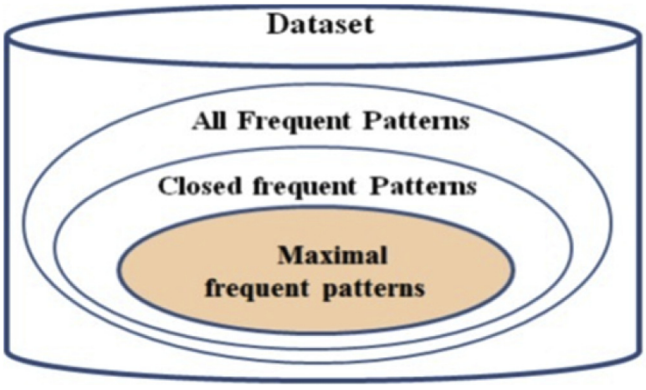

Fig. 7. Search space domain of proposed MICCFD_Miner Technique.

automatically. For this purpose, the third proposed technique called Tuple Repairing abbreviated as T_Repair is proposed. The T_Repair algorithm exploits rules generated from MICCD_Miner technique, to be applied on the dataset and repair errors found in tuples. Finally, the algorithm leaves data in a consistent state. The pseudo-code of Fig. 8 shows the main steps of T_Repair algorithm.

T_Repair algorithm. The algorithm T_Repair produces repaired tuples from given relation as shown in Fig. 8. Given the set of MICCFD rules and tuples of relation *R*, the algorithm iterates over each tuple in relation to be tested against each rule. It also clears the counter for all rules, and initializes a tuple to be repaired to false. Next, it iterates for each rule of MICCFD rules repository and increases the counter of rules. The algorithm checks if left-hand side of the rule matches with tuple field(s), it also checks the right-hand side of the rule, if match skip the tuple as it validated to take true value. Otherwise update tuple field (s) and set updated to true. Finally, it returns updated tuples. It repeats this process until end of tuples from the relation.

6. Experimental study

In this section, an experimental study for validating the proposed techniques is presented. The experimental study is conducted on both real-life and synthetic datasets from one of the critical domains, *i.e.*, medical domain. The exploited datasets have large amount of information about patients and their status. These datasets are used for evaluating the performance of the three proposed techniques i.e., ICCFD_Miner, MICCFD_Miner, and T_Repair.

The proposed techniques are used for generating dependable rules from these datasets, and they also are employed for fixing errors found automatically to yield data into a consistent state. The cleaned data become on demand data access for decision making and management purposes, enabling accurate decisions based on their exact quality of data. The proposed techniques are evaluated using several measures including:

- **Accuracy**: generating accurate dependable rules.
- **Response time measure.**
- **Efficiency**: extracting maximal frequent patterns.
- **Memory space consumption.**

Pseudo Code of Proposed T_Repair Algorithm

Input: A set $\sum(\alpha_m)$ of MICCFD Rules Repository, tuple (t_i) of Relation R(t).
Output: A repaired tuple (t_i`[n]).

```
1. for each t_i[n]∈ R(t)    do  counter(α_m)=0 ;      updated := false;
2.     for each α ∈ ∑(α_m)     do   counter(α)++;
3.         if L.H.S(α) matches t_i[n] ;
4.           then if  R.H.S(α) matches t_i[n];
5.                    then skip t_i[n] as it validated to take true value;
6.                    else update t_i[n]=R.H.S(R);     updated := true;
7.                        return t_i`[n];
8.                end if
9.         end if
10.    end for each
11. end for each
```

Fig. 8. Third proposed T_Repair algorithm.

- **Scalability with dataset size and arity of relation.**
- **Certainty Factor (CF):** investigate the correctness of output generated rules.
- **Validation**: efficiency of T_Repair algorithm using MICCFD-Miner rules repository.

6.1. Experimental setting

Extensive experiments are conducted using both synthetic and real-life datasets. Such datasets are downloaded from the repository of UCI machine learning (http://achive.ics.uci.edu/ml/) namely, Thyroid, Primary-tumor, Cardiotocography, Mushroom, Adult, Audiology, Pima_diabets, and Cleveland-14-heart. Table 4 shows the number of attributes and the number of instances in each dataset.

All experiments are implemented using Java (JDK1.7). The implementation is executed on a machine with an Intel Pentium Dual T3400 processor and 2 GB of memory running on Windows 7 OS. Each experiment is repeated over 5 times and the average is reported.

6.2. Experimental results

In this section, the results from the experimental study over three proposed techniques are demonstrated and discussed.

Table 4
Datasets description.

Dataset name	Arity (Number of columns)	Size (Number of instances)
Thyroid (hypothyroid)	30	3772
Primary-tumor	18	339
Cardiotocography	23	2126
Mushroom	23	8124
Pima_diabets	9	768
Adult	15	32,561
Audiology	70	226
Cleveland-14-heart	14	303

Several experiments are conducted on datasets from medical applications domain, each of them is carried out for evaluating the proposed techniques using one of the mentioned measures.

6.2.1. Experiment-1

These experiments aim to measure the accuracy of generating accurate and dependable rules, and also aim to measure the response time. The accuracy and response time measure of rules generated from first proposed ICCFD-Miner technique, which are minimum and non-redundant are compared to CCFD-ZartMNR algorithm [17] over five datasets, i.e., Thyroid, Cardiotocography, Primary-tumor, Cleveland-14-heart, and Pima_diabets as follows:

A. Experiment 1 generates the dependable rules from Thyroid dataset. Recall that Thyroid dataset includes 30 attributes and 3772 tuples of patient data describing the hypothyroid diagnoses. The results conducted on this dataset are shown in Figs. 9 and 10.

In Fig. 9, by changing threshold values of minimum support and minimum confidence, it is noticed that the proposed ICCFD_Miner technique outperforms the CCFD-ZartMNR algorithm in generating the accurate interest minimal non-redundant rules. For instance, given min-support = 0.97 and min-confidence = 0.99, the number of generated rules are 85 and 220 by the proposed ICCFD-Miner and the CCFD_ZartMNR algorithms, respectively.

In Fig. 10, results show that the ICCFD_Miner also outperforms the CCFD-ZartMNR algorithm in generating rules with different values of support and confidence values with respect to response time.

B. The experimental results conducted on Primary-tumor diseases dataset are shown in Figs. 11 and 12. Recall that this dataset contains 18 attributes and 339 records.

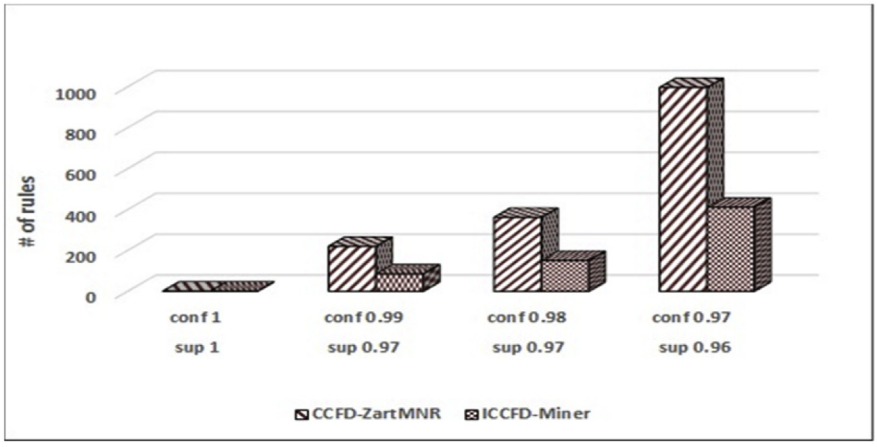

Fig. 9. Total number of rules generated from Thyroid dataset.

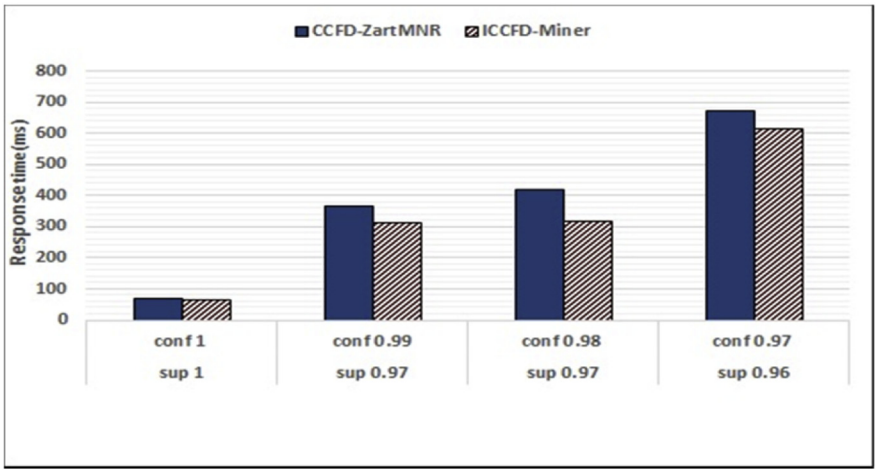

Fig. 10. Response time measure of generating rules from Thyroid dataset.

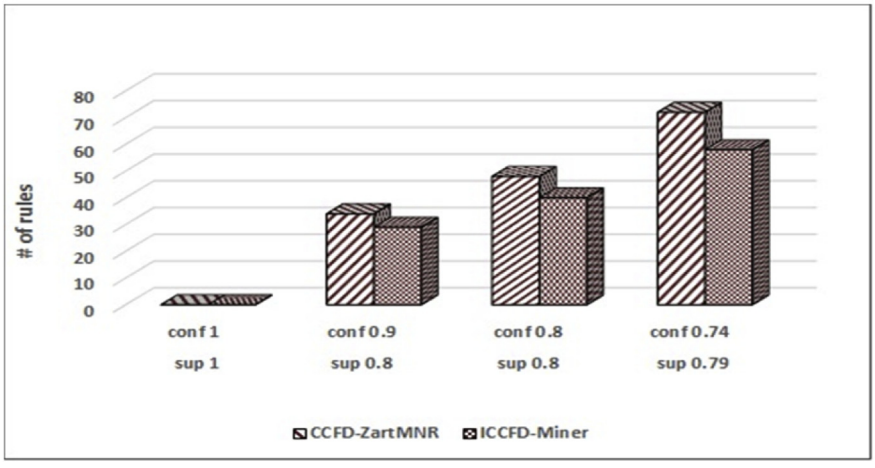

Fig. 11. Total number of rules generated of Primary-tumor dataset.

Figs. 11 and 12 validate the efficiency of the proposed ICCFD-Miner technique against CCFD-ZartMNR algorithm on Primary-tumor dataset with respect to the number of rules generated and response time.

C. The effectiveness of the proposed ICCFD-Miner technique against CCFD-ZartMNR algorithm over Cardiotocography disease patient data is evaluated. Recall that this dataset contains 23 attributes and 2126 records. Results

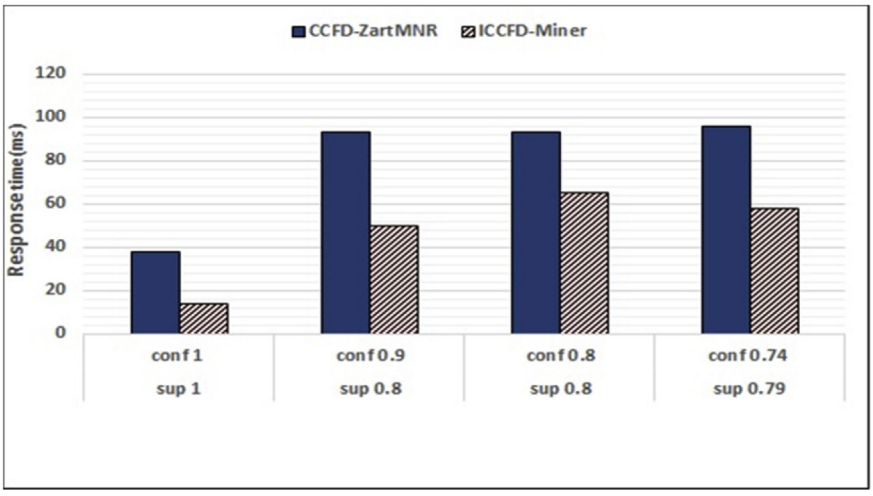

Fig. 12. Response time measure of generating rules from Primary-tumor dataset.

prove that the proposed technique is more efficient than another algorithm as shown in Figs. 13 and 14.

6.2.2. Experiment-2

The second experiment is conducted to evaluate the efficiency of extracting maximal frequent patterns from the second proposed MICCFD-Miner technique compared to the existing CCFD-ZartMNR [17] technique. Also, memory space consumption from second proposed MICCFD-Miner technique compared to existing CCFD-ZartMNR technique is measured.

The experiment is conducted on the mushroom dataset. Recall that this dataset contains 23 attributes and 8124 records. Mushroom dataset describes mushroom physical characteristics, whereas each mushroom species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. The results on MICCFD-Miner compared to the existing technique CCFD-ZartMNR over mushroom dataset are shown in Figs. 15 and 16.

Fig. 15 shows the efficiency of the reduction of the number of patterns generated by the second proposed MICCFD-Miner technique over another CCFD-ZartMNR technique without loss of information. The *x*-axis is the different support values, and the *y*-axis is the total number of extracted patterns, and set fixed constant confidence value equal to 1. For example, with minimum support value equal to 0.4 and confidence value equal to 1, the number of maximal frequent patterns extracted from the proposed MICCFD_Miner technique is 41 patterns, but a number of closed patterns generated from CCFD-ZartMNR technique are 140. Such results show the effectiveness of extracting maximal frequent patterns using MICCFD_Miner as the first step in generating accurate and dependable rules instead of extracting frequent closed patterns from current existing techniques.

However, Fig. 16 demonstrates the memory space consumption of the number of patterns extracted from the second proposed MICCFD_Miner technique compared to CCFD-ZartMNR technique. The *x*-axis is the support values, and the *y*-axis is the memory space consumption in megabytes

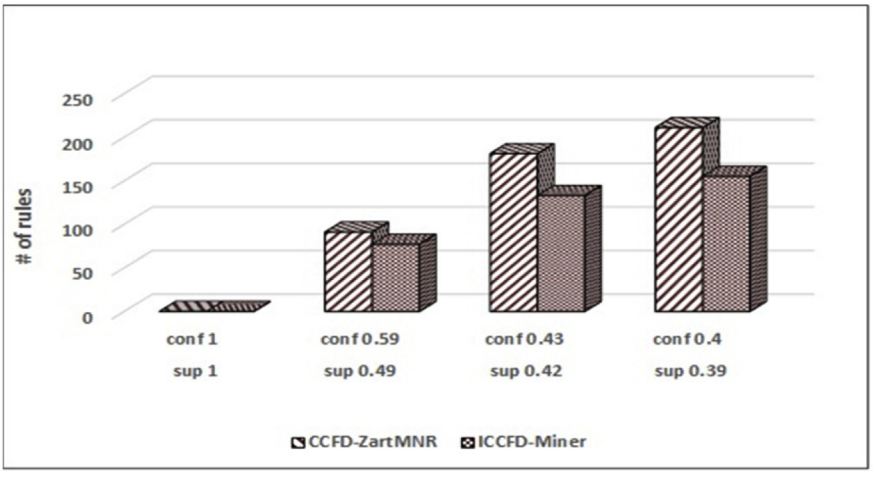

Fig. 13. Total number of rules generated from Cardiotocography dataset.

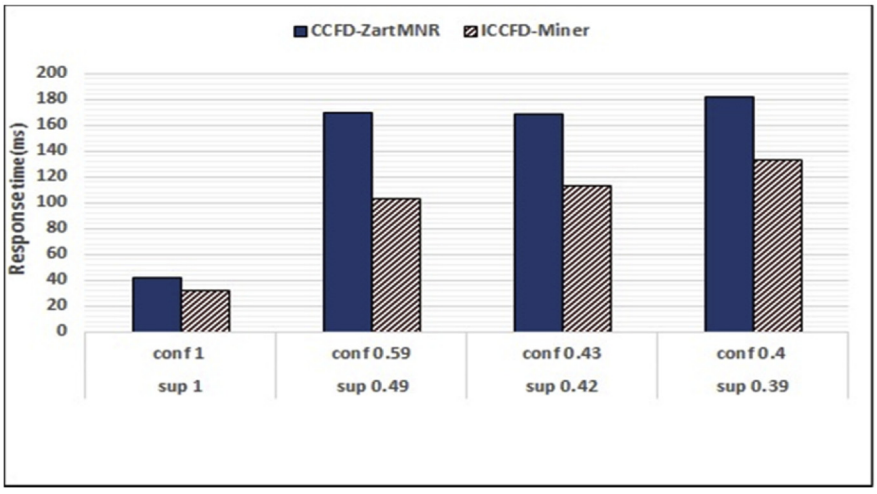

Fig. 14. Response time measure of generating rules from the Cardiotocography dataset.

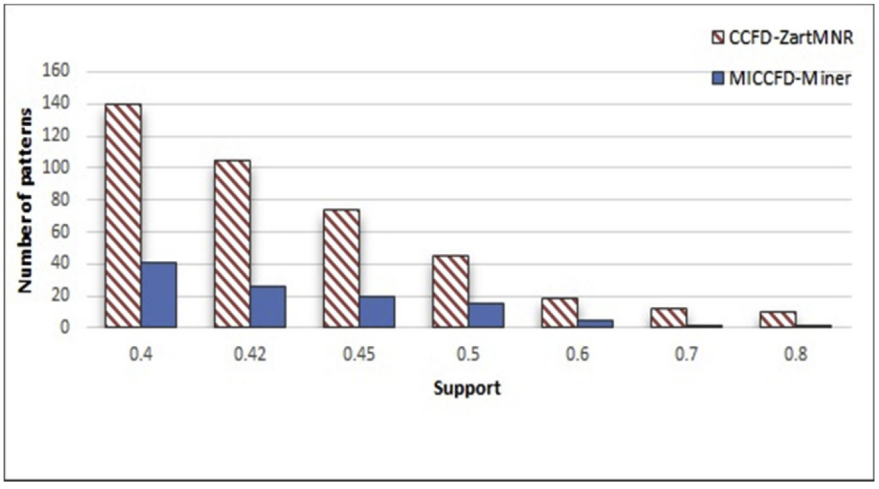

Fig. 15. Total number of maximal frequent patterns extracted from mushroom dataset.

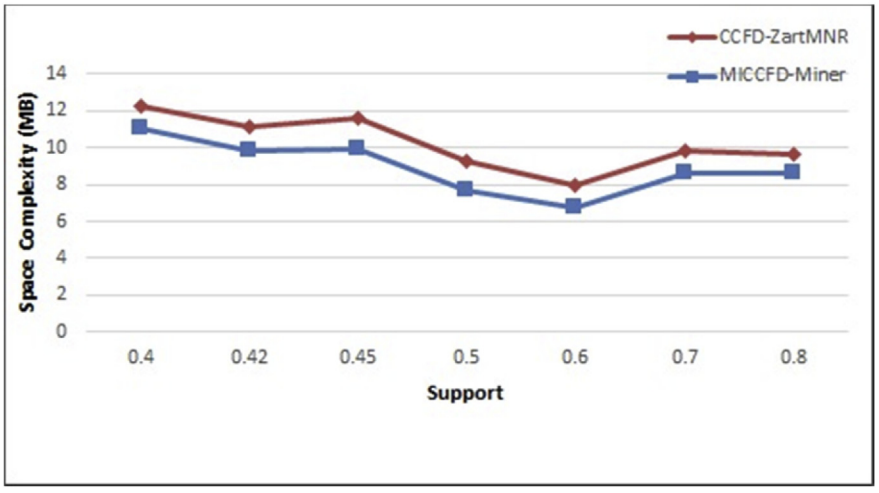

Fig. 16. Space complexity measure about mushroom dataset.

(MB), at fixed constant confidence value equal to 1. Note that, focusing on maximal frequent patterns yields optimization of the memory space, which shown in Fig. 16.

6.2.3. Experiment-3

The experiment aims to evaluate the scalability of the second proposed MICCFD_Miner technique compared to CCFD-ZartMNR technique [17], by varying the dataset size and the arity of relation. This experiment is performed on two datasets, *i.e.*, Adult and Audiology.

A. In this experiment, the scalability of the second proposed MICCFD_Miner technique when increasing the number of tuples at fixed number of attributes compared to CCFD-ZartMNR technique is evaluated. Results from the Adult dataset are shown in Fig. 17. Recall that this dataset contains 15 attributes and 32,561 records. It describes prediction about whether income exceeds $50K/yr or not based on census data. While the arity is fixed to 15, minimum confidence to 0.95 and minimum support to 0.5, the dataset size (number of tuples) is increasing from 5 k to 30 k tuples. The response time of the second proposed MICCFD_Miner technique compared to the existing CCFD-ZartMNR technique by varying number of tuples is reported in Fig. 17, which shows linear time scaling of our proposed MICCFD_Miner technique over existing CCFD-ZartMNR technique. The *x*-axis is the number of tuples *1000 and *y*-axis is the measure of response time in milliseconds (ms).

This experiment concludes that the proposed MICCFD_-Miner technique outperforms another existing CCFD-ZartMNR technique in scalability of increasing number of tuples with less response time.

B. The evaluation of the scalability by increasing the number of attributes at fixed number of tuples for the second proposed MICCFD_Miner technique compared to CCFD-ZartMNR technique is performed. Results conducted on audiology dataset, which contains 70 attributes and 226 instances. The results of this experiment are depicted in Fig. 18.

While fixing database size to 226 tuples, minimum confidence to 1, and minimum support to 0.99, the arity is

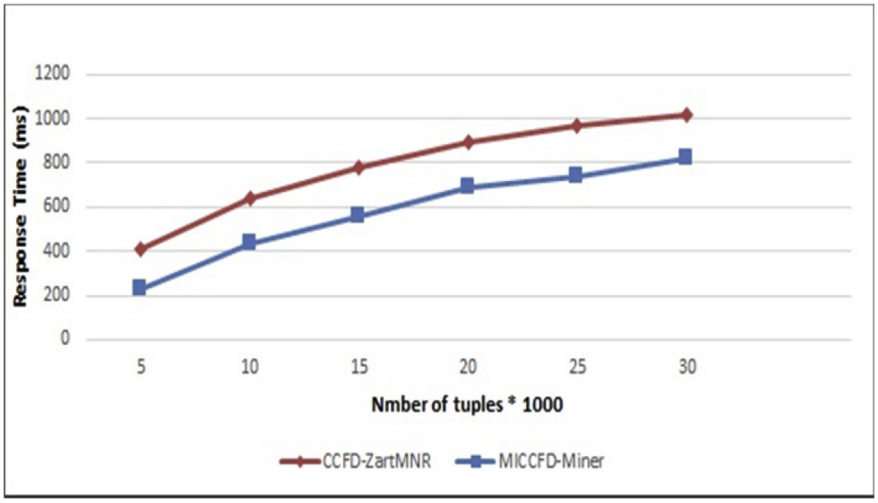

Fig. 17. Scalability per number of tuples on the adult dataset.

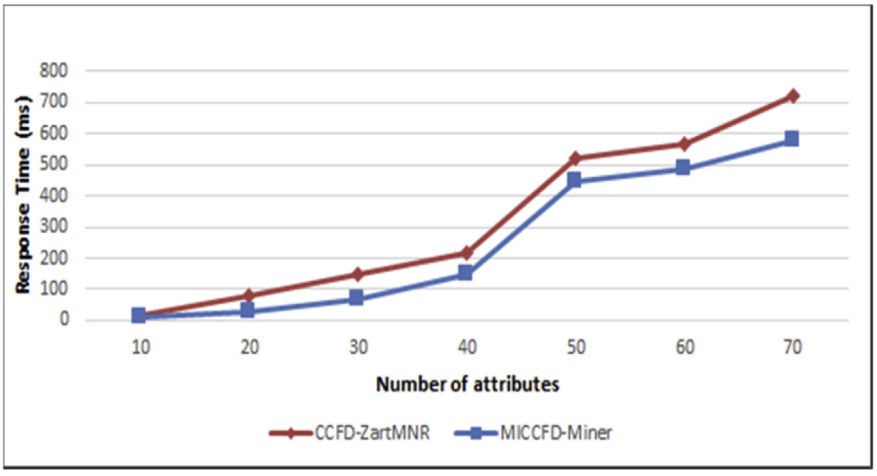

Fig. 18. Scalability per number of attributes in audiology dataset.

increasing from 10 to 70 attributes. Fig. 18 shows the response time measure of the second proposed MICCFD_Miner technique outperforms the existing CCFD-ZartMNR technique by varying number of attributes.

6.2.4. Experiment-4

In this experiment, the correctness of output generated rules from the two proposed techniques, *i.e.*, ICCFD-Miner and MICCFD-Miner, compared to CCFD-ZartMNR technique is investigated by using certainty factor measure. Certainty factor (CF) is defined as a criterion to measure the strength of generated rules and verify redundancy elimination of rules without affect the quality of the other rules, as described in Equation (4) [34].

$$\text{Certainty Factor}(X \rightarrow A) = \frac{(p(A|X) - p(A))}{1 - p(A)} \quad (4)$$

Interval results of the certainty factor measure have the range [-1:1], whereas values range as follows:

(i) CF values equal 0, which means no evidence of rules.
(ii) CF values less than 0, which means negative evidence of rules.
(iii) CF values greater than 0, which means favor for the correctness of output generated rules.

The certainty factor is applied to six medical datasets including Thyroid, Primary-tumor, Clevend-14-heart, Mushroom, Cardiotocography, and Audiology datasets. The results of the two proposed techniques over these datasets are shown in Table 5. These results assure and validate the output of generated rules from the two proposed techniques. It indicates that such techniques always generate precise rules with certainty factor greater than 0 compared to CCFD-ZartMNR technique. The CCFD-ZartMNR generates some rules that have negative evidence (less than zero).

6.2.5. Experiment-5

In this experiment, the ICCFD-Miner and MICCFD-Miner techniques are compared against CCFD-ZartMNR technique to measure the consumption of memory space. This experiment is conducted on Primary-tumor dataset, which contains 18 attributes and 339 records. Results of this experiment are shown in Figs. 19 and 20.

Fig. 19 indicates that second proposed MICCFD-Miner technique be efficient in saving memory space than the first proposed ICCFD-Miner technique and CCFD-ZartMNR technique. Fig. 20 also shows that the second proposed MICCFD-Miner technique outperforms other two techniques in response time. It is also noticed that there is not a significant difference between both proposed techniques.

6.2.6. Experiment-6

In this experiment, the efficiency of T_Repair algorithm is evaluated using MICCFD-Miner output generated rules repository. The T_Repair algorithm is validated on the previously mentioned medical datasets. Results of this experiment are indicated in Figs. 21 and 22 and Table 6.

While Fig. 21 shows the numbers of tuples updated from each dataset after applying T_Repair algorithm, Fig. 22 shows the response time measure for applying the T_Repair algorithm. It is noticed that response time is based mainly on the size of relation and number of errors updated. Table 6 shows the percentage of fixed tuples for each dataset numerically.

6.3. Results discussion

Results of the conducted experiments show that both the proposed ICCFD-Miner and MICCFD-Miner techniques outperform CCFD_ZartMNR algorithm. The outperformance of the proposed ICCFD-Miner technique occurs due to its implying lift measure when generating the accurate minimal interest dependable rules for detecting data inconsistency problems. Furthermore, the proposed ICCFD_Miner technique focuses on closed patterns with their associated generators, *i.e.*, supersets of closed patterns, as a search space for generating the accurate, interest minimum non-redundant and reliable rules. Since the MICCFD-Miner technique focuses on maximal frequent patterns, it reduces the number of patterns generated compared with CCFD-ZartMNR technique without loss of information. This reduction results in better performance with respect to consumed memory and response time. Moreover, the degree of good quality of generated rules by the proposed techniques are ensured by the certainty factor measure.

7. Conclusions and future works

In this paper, both the ICCFD-Miner and MICCFD-Miner techniques have presented for generating accurate, dependable rules for resolving data inconsistency problems in databases. Moreover, the T_Repair algorithm are proposed which

Table 5
Number of rules generated with certainty factor measure.

Dataset name	Arity (# of Columns)	Size (# of Instances)	Number of rules generated (CF $\leq$0) from CCFD-ZartMNR	Number of rules generated (CF > 0) from proposed techniques (ICCFD_Miner, MICCFD_Miner)
Thyroid (hypothyroid)	30	3772	220	85
Primary-tumor	18	339	34	29
Cleveland-14-heart	14	303	10	9
Mushroom	23	8124	9	6
Cardiotocography	23	2126	35	32
Audiology	70	226	487	87

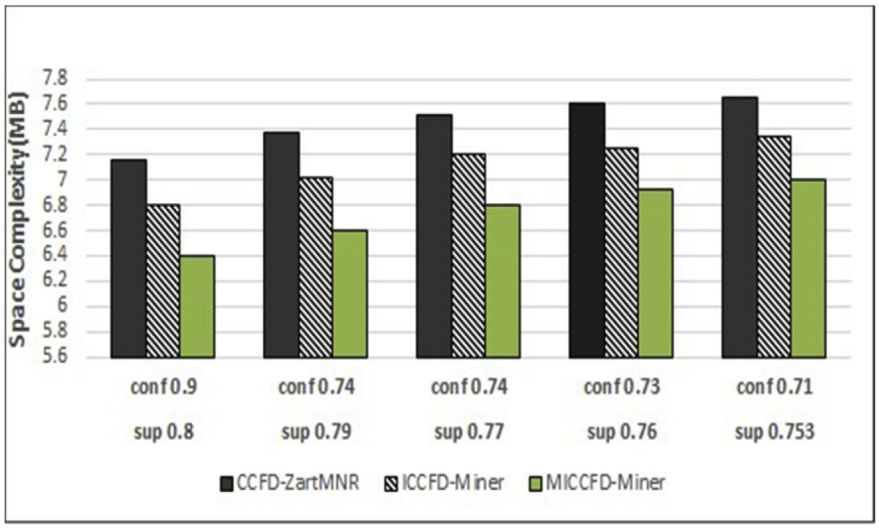

Fig. 19. Space complexity measure over three techniques on Primary-tumor dataset.

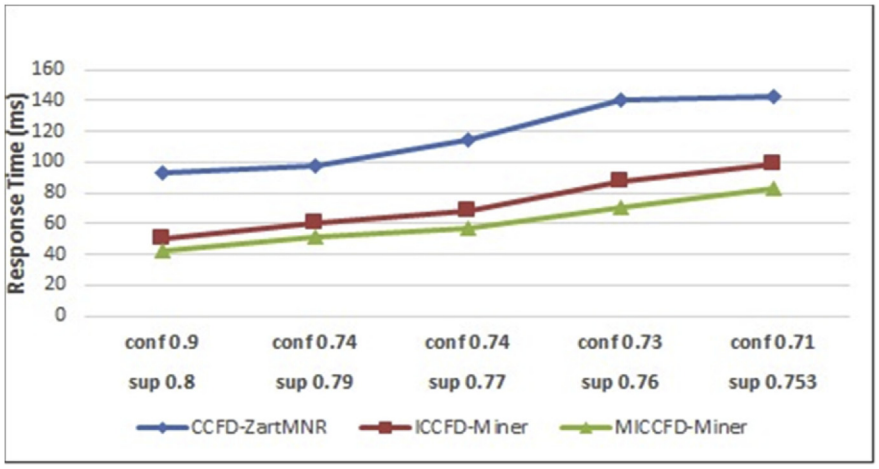

Fig. 20. Time Complexity measure over three techniques on Primary-tumor dataset.

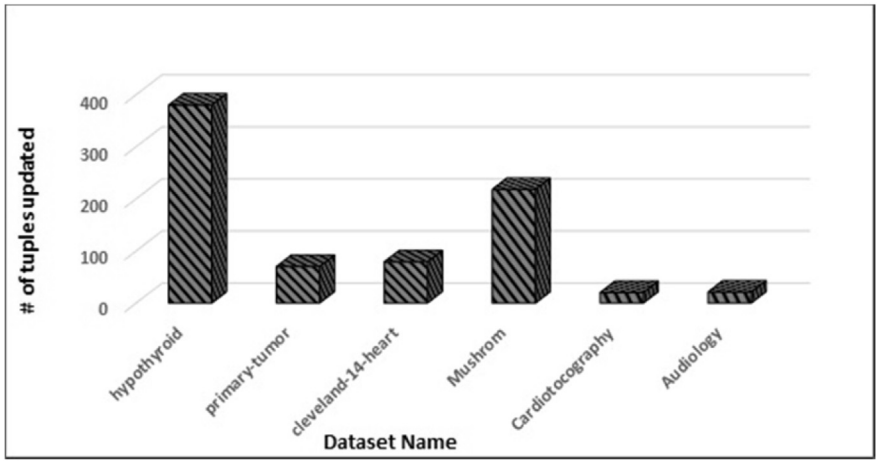

Fig. 21. Number of tuples updated by applying T_Repair technique over different datasets.

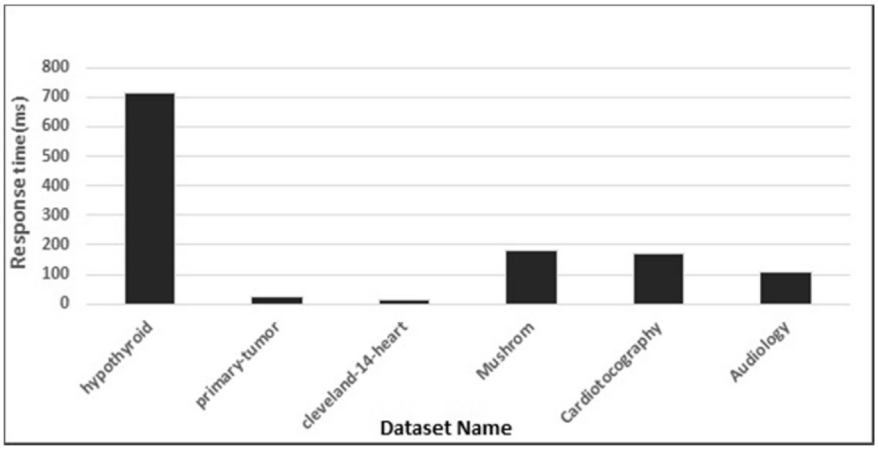

Fig. 22. Response Time measure of applying T_Repair algorithm over different datasets.

Table 6
Number of tuples updated by T_Repair algorithm over different medical datasets.

Dataset name	Arity (# of Columns)	Size (# of Instances)	Number of updated tuples	Percentage of updated tuples
Thyroid (hypothyroid)	30	3772	380	10.07%
primary-tumor	18	339	70	20.64%
cleveland-14-heart	14	303	79	26.07%
Mushroom	23	8124	218	2.68%
Cardiotocography	23	2126	20	0.94%
Audiology	70	226	21	9.29%

takes the action of fixing the inconsistent tuples. The both proposed techniques and the algorithm are included in a generic framework for enhancing data quality. Such framework is evaluated and verified using several datasets coming from the healthcare domain. A variety of experiments is conducted to test the performance of the proposed techniques including the response time, storage space, and the database scalability. Results show that the proposed techniques outperform the competitor technique, i.e., CCFD-ZartMNR.

Finally, future work includes proposing an incremental algorithm for maintaining and updating rules over data streams without re-scanning original database, as well as repairing streaming data. Moreover, the performance of T_Repair technique needs to be considered, it can be enhanced by using indexing and outer join to detect unmatched tuples which need to be fixed.

References

[1] Benjelloun O, Garcia-Molina H, Menestrina D, Su Q, Whang SE, Widom J. Swoosh: a generic approach to entity resolution. VLDB J Int J Very Large Data Bases 2009 Jan 1;18(1):255–76.
[2] Bharambe D, Jain S, Jain A. A survey: detection of duplicate record. Int J Emerg Technol Adv Eng 2012 Nov;2(11):298–307.
[3] Chang IC, Li YC, Wu TY, Yen DC. Electronic medical record quality and its impact on user satisfaction — healthcare providers' point of view. Gov Inf Q 2012 Apr 30;29(2):235–42.
[4] Chiang F, Miller RJ. Discovering data quality rules. Proc VLDB Endow 2008;1(1):1166–77.
[5] Cong G, Fan W, Geerts F, Jia X, Ma S. Improving data quality: consistency and accuracy. In: Proceedings of the 33rd international conference on very large data bases. VLDB Endowment; 2007 Sep 23. p. 315–26.
[6] Fan Wenfei, Geerts Floris, Li Jianzhong, Xiong M. Discovering conditional functional dependencies. Knowl. Data Eng. IEEE Trans 2011; 23(5):683–98.
[7] Fan W, Ma S, Tang N, Yu W. Interaction between record matching and data repairing. J Data Inf Qual 2014 May 1;4(4):16.
[8] Fan W, Li J, Ma S, Tang N, Yu W. Towards certain fixes with editing rules and master data. VLDB J Int J Very Large Data Bases 2012 Apr 1; 21(2):213–38.
[9] Fan W, Geerts F. Capturing missing tuples and missing values. In: Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM; 2010. p. 169–78.
[10] Fan W, Geerts F. Foundations of data quality management. Synth Lect Data Manag 2012;4(5):1–217.
[11] Groves Peter, Kayyali Basel, Knott David, Van Kuiken S. The "big data" revolution in healthcare. McKinsey Q 2013;2(1):1–19.
[12] Hartmann S, Kirchberg M, Link S. Design by example for SQL table definitions with functional dependencies. VLDB J Int J Very Large Data Bases 2012 Feb 1;21(1):121–44.
[13] Haug A, Zachariassen F, Van Liempd D. The costs of poor data quality. J Ind Eng Manag 2011 Jul 4;4(2):168–93.
[14] Herzog TN, Scheuren FJ, Winkler WE. Data quality and record linkage techniques. Springer Science & Business Media; 2007 May 23.
[15] Huang Yu, Chiang F. Towards a unified framework for data cleaning and data privacy. In: Web information systems engineering–WISE 2015. Springer; 2015. p. 359–65.
[16] Hussein N, Alashqur A, Sowan B. Using the interestingness measure lift to generate association rules. J Adv Comput Sci Technol 2015 Jan 1;4(1): 156.
[17] Kalyani DD. Mining constant conditional functional dependencies for improving data quality. Int J Comput Appl 2013;74(15):12–20.
[18] Larsson, P. Evaluation of Open Source Data Cleaning Tools: Open Refine and Data Wrangler, http://courses.cs.washington.edu/courses/cse544/13sp/final-projects/p12-plarsson.pdf.

[19] Li J, Liu J, Toivonen H, Yong J. Effective pruning for the discovery of conditional functional dependencies. Comput J 2013 Mar 1;56(3): 378–92.
[20] Liu J, Li J, Liu C, Chen Y. Discover dependencies from data — a review. IEEE Trans Knowl Data Eng 2012 Feb;24(2):251–64.
[21] Maletic JI, Marcus A. Data cleansing: a prelude to knowledge discovery. In: Data mining and knowledge discovery handbook. US: Springer; 2009. p. 19–32.
[22] Mans Ronny S, van der Aalst Wil MP, Vanwersch RJ. Data quality issues. In: Process mining in healthcare. Springer; 2015. p. 79–88.
[23] Martin Elizabeth, Ballard G. Data management best practices and standards for biodiversity data applicable to bird monitoring data. 2010. http://www.prbo.org/refs/files/12058_Martin2010.pdf.
[24] Mayfield C, Neville J, Prabhakar S. ERACER: a database approach for statistical inference and data cleaning. In: Proceedings of the 2010 ACM SIGMOD international conference on management of data. ACM; 2010 Jun 6. p. 75–86.
[25] Medina R, Nourine L. A unified hierarchy for functional dependencies, conditional functional dependencies and association rules. In: Formal concept analysis. Springer; 2009. p. 98–113.
[26] Mezzanzanica Mario, Boselli Roberto, Cesarini Mirko, Mercorio F. Automatic synthesis of data cleansing activities. In: Proceedings of the 2nd international conference on data technologies and applications; 2013. p. 138–49.
[27] Papenbrock Thorsten, Ehrlich Jens, Marten Jannik, Neubert Tommy, Rudolph Jan-Peer, Schonberg Martin, et al. Functional dependency discovery: an experimental evaluation of seven algorithms. Proc VLDB Endow 2015;8(10):1082–93.
[28] Quan J, Liu Z, Chen D, Zhao H. High-efficiency algorithm for mining maximal frequent item sets based on matrix. In: Computational intelligence and communication networks (CICN), 2012 fourth international conference on. IEEE; 2012 Nov 3. p. 930–3.
[29] Shafique Umair, Majeed Fiaz, Qaiser Haseeb, Mustafa IU. Data mining in healthcare for heart diseases. Int J Innov Appl Stud 2015;10(4): 1312–22.
[30] Stefan B. Addressing internal consistency with multidimensional conditional functional dependencies. Manag Data 2010:1–20.
[31] Vassiliadis P, Simitsis A. Extraction, transformation, and loading. 2009.
[32] Wang J, Tang N. Towards dependable data repairing with fixing rules. In: Proceedings of the 2014 ACM SIGMOD international conference on Management of data. ACM; 2014. p. 457–68.
[33] Weiskopf, Gray Nicole, Weng C. Methods and dimensions of electronic health record data quality assessment: enabling reuse for clinical research. J Am Med Inf Assoc 2013;20(1):144–51.
[34] Xu Y, Li Y, Shaw G. Reliable representations for association rules. Data Knowl Eng 2011 Jun 30;70(6):555–75.
[35] Yao H, Hamilton HJ. Mining functional dependencies from data. Data Min Knowl Discov 2008;16(2):197–219.