Acta Cybernetica **25** (2022) 647–675.

Strongly Possible Functional Dependencies for SQL*

Munqath Alattar[ab] and Attila Sali[ac]

Abstract

Missing data is a large-scale challenge to research and investigate. It reduces the statistical power and produces negative consequences that may introduce selection bias on the data. Many approaches to handle this problem have been introduced. The main approaches suggested are either missing values to be ignored (removed) or imputed (filled in) with new values [14]. This paper uses the second method. Possible worlds and possible and certain keys were introduced in [22, 25], while certain functional dependencies (c-FD) were introduced in [23] as a natural complement to Lien's class of possible functional dependencies (p-FD) by [26], and Weak and strong functional dependencies were studied in [25]. The intermediate concept of strongly possible worlds introduced in a preceding paper [3] and results in strongly possible keys (spKey's) and strongly possible functional dependencies (spFD's) were studied. Also, a polynomial algorithm to verify a single spKey was given and it was shown that it is NP-complete in general to verify an arbitrary collection of spKeys. Furthermore, a graph-theoretical characterization was given for validating a given spFD $X \rightarrow_{sp} Y$.

We show, that the complexity to verify a single strongly possible functional dependency is NP-complete in general, then we introduce some cases when verifying a single spFD can be done in polynomial time. As a step toward axiomatization of spFD's, the rules given for weak/strong and certain functional dependencies are checked. Appropriate weakenings of those that are not sound for spFD's are listed. The interaction between spFD's and spKey's and certain keys is studied. Furthermore, a graph theoretical characterization of implication between singular attribute spFD's is given.

*Research of the second author was partially supported by the National Research, Development and Innovation Office (NKFIH) grants K–116769, K–132696 and SNN-135643. This work is also supported by the National Research, Development and Innovation Fund (TUDFO/51757/2019-ITM, Thematic Excellence Program), the BME NC TKP2020 grant of NKFIH Hungary and it is also connected to the scientific program of the "Development of quality-oriented and harmonized R+D+I strategy and functional model at BME" project, supported by the New Hungary Development Plan (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002).

[a]Department of Computer Science and Information Theory, Budapest University of Technology and Economics, Hungary

[b]Information Technology Research and Development Center, University of Kufa, Iraq, E-mail: m.attar@cs.bme.hu, ORCID: 0000-0002-4673-4902

[c]Alfréd Rényi Institute of Mathematics, Budapest, Hungary, E-mail: sali.attila@renyi.hu, ORCID: 0000-0002-4837-6360

DOI: 10.14232/actacyb.287523

Keywords: strongly possible functional dependencies, strongly possible keys
`NULL` values, data imputation, matchings in bipartite graphs, list coloring

1 Introduction

Incomplete data in databases are allowed in many of the modern systems. For example, when raw data is collected from different sources in data warehousing, some of the attributes may be available in some of the sources and may not be in some others. For this reason, it is important to treat constraints over incomplete tables. Aliriza et al. [12] showed that encountering up to half of the data values missing is common. So using analysis methods that work only with complete data tables makes mining the data more complicated.

Incompleteness occurs in database tables for different reasons. Date [10] identified seven types of `NULL`'s as follows: value not applicable, value unknown, value does not exist, value undefined, value not valid, value not supplied, and value is the empty set. We consider the second, third, and seventh types as this paper deals with data consumption in an incomplete database table. We consider the symbol N/A for the other types of missing data and we assume it belongs to each domain and it is considered as a regular domain value for comparisons and analysis.

Missing values problem complicates data analysis. It also causes a loss of data efficiency and effectiveness [14]. Although some data analysis approaches get over the problem of incomplete databases, still most approaches require complete databases. To overcome the missing data problem, there are mainly two main methods, either ignoring the incomplete tuples or filling them with domain values chosen by some heuristics. [14].

Possible worlds, introduced by Köhler et al. [22], are obtained by replacing each missing data value with a value from its attribute's domain, which can be infinite. For an incomplete table, every possible world is a complete table that may contain some duplicated tuples. They defined a possible (certain) keys as a key that are satisfied by some (every) possible world of a (non-total) database table. For example, the table in Table 2a satisfies the possible key $\{Course\ Name\}$ as there is a possible world that satisfies it, and every possible world of the table satisfies the certain key $\{Course\ Name, Year, Semester\}$. Furthermore, no possible world of the table satisfies the key $\{Lecturer\}$. Weak (strong) functional dependencies were defined by [25] as FD's that are satisfied by some (every) possible world. The table in Table 1a satisfies the strong FD $\Box(Name\ mrg\_status \rightarrow gender)$ because every possible world satisfies it, and the weak FD $\diamond(mrg\_status \rightarrow gender)$, as there exist some possible worlds, but not all, satisfying the implication.

Most often, especially when the attribute's domain is not known, there is no reason to consider any other attribute value than the already existing values of the table. Types of cars, diagnoses of patients, applied medications, dates of exams, course descriptions, etc, are some examples. For that, the strongly possible world is

Table 1: Possible and Strongly Possible Worlds

(a) Incomplete Table

Name	**gender**	**mrg_status**	**age**
⊥	female	married	32
Sarah	female	⊥	⊥
David	⊥	divorced	38
James	male	single	⊥
James	male	widower	47

(b) Possible World

Name	**gender**	**mrg_status**	**age**
30	female	married	32
Sarah	female	lawyer	high
David	Apple	divorced	38
James	male	single	-12
James	male	widower	47

(c) Strongly Possible World

Name	**gender**	**mrg_status**	**age**
David	female	married	32
Sarah	female	single	32
David	male	divorced	38
James	male	single	38
James	male	widower	47

defined as a possible world achieved by substituting each occurrence of `NULL` with a value from the corresponding attribute's existing values [3]. Strongly possible key(FD) is a key(FD) that satisfied by a strongly possible world. Values that are shown in each attribute of a table represent a part of that attribute's domain. When the attribute domain is unknown, choosing values out of its real domain to fill in a `NULL` may distort the data. For example, it would be unsuitable using value other than the ones appearing in the *marriage status* attribute to fill the ⊥ in the second row in Table 1a. Other values like numbers, symbols, or any other strings with distant meanings may cause distortion. Therefore, a more meaningful and

semantically acceptable possible (strongly possible) world would be provided using one of the already shown values in the attribute. As the possible world in Table 1c is preferred to the one in Table 1b. Strongly possible worlds were introduced by [3] where strongly possible keys (spKey's) were studied. The natural extension to strongly possible functional dependencies was investigated in [4].

1.1 Contributions

In the present paper, we study spFD's that were earlier introduced in [4, 5] as a generalized version of functional dependency constraint using the visible domain concept and we continue the work started in [3]. We provided a graph-theoretical condition for the spFD satisfaction in [4]. We give a list of axioms of weak/strong and certain FD's that are sound for spFD's, as well and describe several possible sound weakenings for those that are not sound. Interactions between spKeys and strongly possible FDs/certain FDs are investigated, and also the spFD's between singular attributes are studied. The main contributions of this paper are as follows:

- We analysed the properties of strongly possible keys and introduced a (worst-case exponential time) algorithm to verify a single spKey in [3]. A polynomial-time solution for the same problem is given and we show that verifying an arbitrary system of strongly possible keys is NP-complete, thus resolving an open problem from [3].
- A graph-theoretical characterization was given in [4] to verify when spFD $X \rightarrow_{sp} Y$ holds in an SQL table T containing `NULL`'s. We use that here to show that the satisfaction problem for a single spFD $X \rightarrow_{sp} Y$ is NP-complete.
- We give a list of the axioms of Weak/Strong and Certain FD's that are sound for spFD's, with several possible weakenings or restrictions that keep soundness for those that are not sound.
- An interaction between the spKeys and sp/c-FDs is studied and the weakening and transitivity rules are introduced.
- We analyzed the case when the spFD's are restricted to singular attributes. A complete characterisation of the implication problem of this special case is given based on the fact that all the possible extensions of any `NULL` occurrences are shown in the table.

1.2 Paper structure

The organization of this paper is as follows. Section 2 contains the necessary definitions. Section 3 discusses related work. Algorithmic and complexity questions of strongly possible keys and strongly possible functional dependencies are introduced in Section 4. In section 5, some spFD's implication properties are studied. First the case if one side of an spFD is fixed is treated. Then sound implication rules for

spFD's are listed. The proof of their correctness are given in [5]. Then the special case of spFD's between single attributes is treated using graph theoretical methods. Finally, Section 6 contains concluding remarks and future research directions.

2 Definitions

In this section, we recall some basic definitions introduced in [4]. Let $R = \{A_1, A_2, \ldots A_n\}$ be a relation schema. The set of all the possible values for each attribute $A_i \in R$ is called the domain of A_i and denoted by $D_i = dom(A_i)$ for $i =$ 1,2,... n.

An instance $T = (t_1,t_2, \ldots t_s)$ over R is a list of tuples that each tuple is a function $t : R \to \bigcup_{A_i \in R} dom(A_i)$ and $t[A_i] \in dom(A_i)$ for all A_i in R. Note that this definition of tuples emphasizes that the order of the attributes in schema R is irrelevant and by taking a list of tuples we use the *bag semantics* that allows several occurrences of the same tuple. For a tuple $t_r \in T$ and $X \subset R$, let $t_r[X]$ be the restriction of t_r to X.

It is assumed that $\perp$ is an element of each attribute's domain that denotes missing information. t_r is called V-total for a set V of attributes if $t_r[A] \neq \perp$, $\forall A \in V$. Also a tuple t_r is a total tuple if it is a R-total. t_1 and t_2 are *weakly similar* on $X \subseteq R$ denoted as $t_1[X] \sim_w t_2[X]$ defined by Köhler et al. [22] if:

$$\forall A \in X \quad (t_1[A] = t_2[A] \text{ or } t_1[A] = \perp \text{ or } t_2[A] = \perp).$$

Furthermore, t_1 and t_2 are *strongly similar* on $X \subseteq R$ denoted by $t_1[X] \sim_s t_2[X]$ if:

$$\forall A \in X \quad (t_1[A] = t_2[A] \neq \perp).$$

For the sake of convenience, we write $t_1 \sim_w t_2$ if t_1 and t_2 are weakly similar on R and use the same convenience for strong similarity. Thus, if t_1 and t_2 are both weakly similar to a `NULL`-free tuple t, then $t_1 \sim_w t_2$. This is true because both of the non-total tuples t_1 and t_2 could be extended to be equal to the total tuple t.

Let $T = (t_1, t_2, \ldots t_s)$ be a table instance over R. $T' = (t'_1, t'_2, \ldots t'_s)$ is a *possible world* of T, if $t_i \sim_w t'_i$ for all $i = 1, 2, \ldots s$ and T' is completely `NULL`-free. That is, we replace the occurrences of $\perp = t[A_i]$ with a value from the domain D_i different from $\perp$ for all tuples and all attributes.

Weak functional dependency $\diamond X \to Y$ holds in T if there exists a possible world T' such that $T' \models X \to Y$ in the classical sense, that is functional dependency $X \to Y$ holds in T' meaning that if $t'_i[X] = t'_j[X]$ then $t'_i[Y] = t'_j[Y]$ is satisfied, for all pairs of tuples $t'_i, t'_j \in T'$. *Strong functional dependency* $\Box X \to Y$ holds in T if functional dependency $X \to Y$ holds in all possible worlds T' of T. X is a *possible key* if there exists a possible world T' such that X is a key in T', and X is a *certain key* if it is a key in every possible world of T. The following was proven in [22].

Theorem 1. *$X \subseteq R$ is a certain (possible) key iff*

$$\forall t_1, t_2 \in T\colon t_1[X] \not\sim_w t_2[X]\,(t_1[X] \not\sim_s t_2[X]).$$

Table 2: Complete and Incomplete Datasets

(a) Incomplete Dataset

Course Name	**Year**	**Lecturer**	**Credits**	**Semester**
Mathematics	2019	$\perp$	5	1
Datamining	2018	Sarah	7	$\perp$
$\perp$	2019	Sarah	$\perp$	2

(b) Complete Dataset

Course Name	**Year**	**Lecturer**	**Credits**	**Semester**
Mathematics	2019	Sarah	5	1
Datamining	2018	Sarah	7	2
Datamining	2019	Sarah	7	2

2.1 Strongly possible worlds

The concepts of *visible domain* and *strongly possible world* were introduced in [3].

Definition 1. *The* visible domain *of an attribute A_i (VD_i) is the set of all distinct values except $\perp$ that are already used by tuples in T:*

$$VD_i = \{t[A_i] : t \in T\} \setminus \{\perp\} \text{ for } A_i \in R$$

For example, the visible domain of the *Credits* attribute in Table 2a is $\{5, 7\}$. So, for any dataset with no information about the attributes' domains, we define their structure and domains by the data itself. It is more reliable and realistic when considering only what information we have in a given dataset and depending on extracting the relationship between data to overcome the missing data problem. Strongly possible worlds are obtained by using the visible domain values in place of the occurrence of `NULL`'s.

Definition 2. *A possible world T' of T is called* strongly possible world (spWorld) *if $t'[A_i] \in VD_i$ for all $t' \in T'$ and $A_i \in R$.*

We defined *strongly possible keys (spKey's)* and *strongly possible functional dependencies (spFD's)* in [3, 4] respectively by strongly possible worlds as follows.

Definition 3. *Strongly possible functional dependency $X \rightarrow_{sp} Y$ holds in table T over schema R if there exists an spWorld T' of T such that $T' \models X \rightarrow Y$. X is a strongly possible key if there exists an spWorld T' of T such that X is a key in T', in notation $sp\langle X\rangle$. Note that this is not equivalent to spFD $X \rightarrow_{sp} R$ since we use the bag semantics.*

In Table 2a, the spKey of the two attributes {Course Name, Year} is satisfied, and so is the spFD $CourseName \rightarrow_{sp} Semester$, as the strongly possible world in Table 2b shows it.

For a schema R, the *NULL-free subschema (NFS)* is a subset $R_S \subseteq R$ that corresponds to SQL's `NOT NULL` constraint. A table T over R satisfies NFS R_S if it is R_S-total, that means every tuple $t \in T$ is R_S-total ($\forall A \in R_S \colon t[A] \neq \perp$). If T satisfies NFS R_S, then we say T is over (R, R_s). Also, if Σ is a set of integrity constraints (for example spFD's), then a table T over (R, R_s) is an *Armstrong instance* of Σ if

(1) $T \models \sigma \iff \Sigma \models \sigma$ for any constraint σ, and

(2) $\forall A \in R \setminus R_S$, $\exists t \in T$ such that $t[A] = \perp$.

This is the classical Armstrong instance extended by requiring that if an attribute is not in the `NULL`-free subschema, then it certainly contains `NULL`'s.

3 Related work

Keys and functional dependencies are two main constraints that enforce the semantics of relational databases. Database tables of real database systems usually contain occurrences of null values and for some cases, this happens in candidate key columns. Many studies have been done on handling the missing values problem.

Aliriza et.al. introduced a framework of imputation methods in [14] and evaluated how the selection of different imputation methods affected the performance in [12]. Experimental analyses of several algorithms for imputation of missing values were provided by [16, 13, 2, 7]. An approach introduced by Zhang et.al. discussed and compared different approaches that use only known values [29].

Sree [11] suggests that it is necessary to substitute the missing values with values based on other information in the dataset to overcome the biased effects that affect the accuracy of classification. Likewise, for each `NULL` in an attribute, we use the attribute's existing values. Cheng et.al.[8] used a clustering algorithm to cluster data and calculate coefficient values between different attributes producing the minimum average error.

Interactions of functional dependencies and other integrity constraints with null values have long been studied. Early investigations focused on "fixing" the database using the Chase procedure, such as Grahne did in [15]. Imlienski and Lipski [19] also investigated the properties of Chase concerning `NULL`'s. Kiss and Márkus [21], provided a chase procedure for functional and inclusion dependencies to provide graph manipulation rules for dependency inference.

The two most used interpretations of `NULL`'s are "value unknown at present" and "no information". The first one leads to possible world semantics that is `NULL`'s are substituted by domain values to obtain total tables. A three-valued model of FD satisfaction is given by Vassilou [27]. It takes into account that all possible words of a table T are considered and a functional dependency either holds, does

not hold or may hold on T. This latter means that in some possible worlds it holds, and in some other ones it does not hold. Fuzzyness is naturally associated with imperfect information. For fuzzy relational databases, the truth value of FD's was studied by Achs and Kiss in [1] Levene and Loizou defined weak and strong functional dependencies based on possible world concept. A weak FD holds in some of the possible worlds and a strong FD holds in all possible worlds. A sound and complete axiom system is given for them in [25].

Both of unknown, as well as non-existing data can be treated by the "no information" approach. Lien [26] defined functional dependencies that hold if strong similarity on the LHS implies equality on RHS. The equality here means that when two tuples are equal on an attribute, if the attribute value is `NULL` in one tuple, then the other tuple must also be `NULL` in the same attribute. This dependency is same as the possible FD (p-FD) of Köhler and Link [23]. The novelty of the latter paper is the concept of certain functional dependencies. A c-FD holds if the weak similarity on LHS implies equality on the RHS, and the equality here is defined in the same way as the equality of attribute values on RHS of p-FD's. An axiom system for functional dependencies with `NULL`'s was given by Atzeni and Morfuni [6], but the drawback was that they allowed no `NULL`'s on the LHS.

Levene and Loizou introduced weak and strong FD's, where weak (strong) functional dependency is satisfied if it holds in some (every) possible world [25]. We also used the possible world semantics to introduce our strongly possible functional dependency and it fits between weak and strong FD's. Let suppose we have a table instance with at least one non-`NULL` value is there in each attribute, then a c-FD $X_{\,w} \rightarrow Y$ satisfaction implies the satisfaction of the spFD $X \rightarrow_{sp} Y$. But the satisfaction of $X \rightarrow_{sp} Y$ does not imply the satisfaction of the p-FD $X_{\,s} \rightarrow Y$. For example, $Year \rightarrow_{sp} Credits$ is satisfied in Table2a, while $Year_{\,s} \rightarrow Credits$ is violated. Example 1 shows a brief comparison of the different functional dependencies. This example is shown in [4], we repeat it here after fixing a typo appearing in the conference version.

Example 1. Let T be the following SQL table.

employee	**d**ept	**m**anager	**s**alary
Knuth	`NULL`	Chomsky	100,000
Turing	CS	von Neumann	`NULL`
Turing	`NULL`	Gödel	`NULL`

In the following table, we compare the six types of FD's shown in the SQL table: 3-valued [27], weak and strong [25], possible [26], certain [23] and strongly possible functional dependencies (spFD's).

	3-valued	weak	strong	possible	certain	spFD
$e \to d$	unknown	T	F	F	F	T
$e \to m$	F	F	F	F	F	F
$e \to s$	unknown	T	F	T	T	T
$d \to d$	T	T	T	T	F	T
$d \to m$	unknown	T	F	T	F	F
$m \to e$	T	T	T	T	T	T
$m \to d$	T	T	T	T	T	T

Possible and certain keys were introduced by Köhler et. al. [22], such that a set K of attributes is a possible (certain) key if it is a key in some (every) possible world. The "strongly possible" concept of the present paper is in between of these two, where a strongly possible world is a possible world also. As possible worlds may use any value from an attribute domain to substitute a `NULL` it may allow an infinite set of values. On the other hand, strongly possible worlds allow only a finite set of values and created from finite attribute domains. Some of the results in [22] assume that some attribute domains are infinite. In this paper, we study the dependencies without that assumption.

Imielinski [18] introduced the construction of OR-relations which is another closely related concept to our spFD's. An OR-tuple over schema $R(A_1, A_2, \ldots, A_n)$ is a mapping $t\colon R \to \cup_{i=1}^{n} \mathcal{P}_0(dom(A_i))$ such that $t[A_i] \in \mathcal{P}_0(dom(A_i))$, where $\mathcal{P}_0(X)$ is the collection of finite subsets of set X. An OR-relation r is a bag of OR-tuples and a possible world r' for r is a collection of tuples t' such that $t'[A_i] \in t[A_i]$ for all $t \in r$. A wFD $\diamond X \to Y$ is satisfied by an OR-relation r iff there is a possible world r' of r such that $r' \models X \to Y$. Hartmann and Link already noted in [17] that Union rule does not hold for wFD's in OR-relations. Our strongly possible world concept can be modelled by OR-relations in such a way that if $t[A_i] = \perp$ for some tuple in an SQL table T, then $t[A_i]$ is replaced by the OR-set $VD(A_i)$, the visible domain of A_i. So strongly possible worlds and spFD's can be considered as special cases of possible worlds of OR-relations and wFD's in them. Our analysis shows, that most of the classical implication rules, axioms do not hold even in this special case.

Finally, Wei and Link [28] aim to define a solid generalization of functional dependencies that do not rely on the interpretation of `NULL`'s. Also, the two papers [23, 28] studied the databases normalization based on the appropriate functional dependencies. It is a future research topic on how our spFD's could be applied for normalization, as normalization is important to eliminate redundancy that may cause inconsistency at updates.

4 Algorithmic and complexity questions

In this section the complexity questions connected to strongly possible keys and strongly possible functional dependencies are studied. The main problems are NP-complete in general, but several important special cases can be solved in polynomial time.

4.1 Strongly possible keys

Following is the algorithmic question we study here. Let T be a SQL table and Σ be a collection of strongly possible key constraints, does $T \models \Sigma$ hold? We may assume, without loss of generality, that there exists at least one spWorld for every treated table. This is a reasonable assumption, as the non-existence of an spWorld happens only if a table contains only `NULL`'s in an attribute.

In [3], an algorithm is given using bipartite matchings for the case when a single spKey needs to be checked, i.e, $\Sigma = \{sp\langle K\rangle\}$. If $K = \{A_1, A_2 \dots A_b\}$, then the running time of that algorithm is $O(|R|(|T| + |T^\star|)) + O((|T| + |T^\star|)|E|)$, where $T^\star$ is the set of total tuples $T^\star = \{t^\star \in \Pi_{i=1}^{b} VD_i \colon \exists t \in T$ such that $t[K] \sim_w t^\star[K]\}$. However, the size of $T^\star$ can be an exponential function of size of T. In [4], we gave a polynomial-time refinement of that algorithm. The refined algorithm states that if a single spKey $sp\langle K\rangle$ is to be checked, then considering $T|_K$ is enough, because of K is a key if and only if the tuples are pairwise distinct on K. Next proposition was introduced in [3], it gave a sufficient condition to determine the satisfaction of a given spKey. Let us assume that $T^\star \subseteq VD_1 \times VD_2 \times \ldots \times VD_b$ is defined by $T^\star = \{t^\star \in VD_1 \times VD_2 \times \ldots \times VD_b : \exists t \in T : t[K] \sim_w t^\star\}$, and define a bipartite graph $G = (T, T^\star; E)$ with $\{t, t^\star\} \in E \iff t[K] \sim_w t^\star[K]$. Note that we use here the standard bipartite graph notation, T and $T^\star$ are the partite classes. Propositions 1, 2 and 3, Theorem 2, and Algorithm 1 were proven in [4]. We repeat them here to be self-contained.

Proposition 1. *$T \models sp\langle K\rangle$ holds if and only if there exists a matching in $G = (T, T^\star; E)$ covering T.*

We may resort to generating only part of $T^\star$ to ensure that the algorithm will run in polynomial time. Let $T = \{t_1, t_2 \dots t_m\}$ and $\ell(t_i) = |\{t^\star \in VD_1 \times VD_2 \times \ldots \times VD_b \colon t^\star \sim_w t_i[K]\}|$. Note that $\ell(t_i) = \prod_{j \colon t_i[A_j] = \perp} |VD_j|$, then these values can be calculated by scanning T once and using appropriate search tree data structures to hold values of visible domains of each attribute. Sort tuples of T in non-decreasing $\ell(t_i)$ order, i.e. assume that $\ell(t_1) \leq \ell(t_2) \leq \ldots \leq \ell(t_p)$. Let $j = \max\{i \colon \ell(t_i) < i\}$ and $T_j = \{t_1, t_2, \dots t_j\}$, and furthermore, $T_j^\star = \{t^\star : \exists t \in T_j : t^\star \sim_w t[K]\} \subseteq VD_1 \times VD_2 \times \ldots \times VD_b$. Note that $|T_j^\star| \leq \frac{1}{2} j(j-1)$. If $\forall i = 1, 2, \dots, m \colon \ell(t_i) \geq i$, then define $j = 0$ and $T_j^\star = \emptyset$.

Proposition 2. *$T \models sp\langle K\rangle$ holds if and only if $j = 0$ or there exists a matching in $G' = (T_j, T_j^\star; E|_{T_j \times T_j^\star})$ covering T_j.*

Proposition 2 was proven in [4] and it gives the basis of a polynomial-time algorithm to determine whether $T \models sp\langle K \rangle$ holds or not.

Algorithm 1 Verifying $T \models sp\langle K \rangle$

Input: Table T over schema R, $K \subseteq R$
Output: Strongly possible world $T^\star$ showing $T \models sp\langle K \rangle$ if exists

```
1: procedure spKey(item T, item R, item K)
2:     Calculate ℓ(t): t ∈ T
3:     Sort T_i in non-decreasing ℓ(t_i) order
4:     j ← max{i: ℓ(t_i) < i}
5:     Construct bipartite graph G' = (T_j, T_j^⋆; E|_{T_j × T_j^⋆})
6:     M = MaxMatching(G')
7:     if |M| < j then return T ⊭ sp⟨K⟩
8:         T^⋆ ← M ∩ T_j^⋆
9:         for k = j + 1 to |T| do
10:            Generate t_k^⋆ ∉ T^⋆ such that t_k ∼_w t_k^⋆
11:            T^⋆ ← T^⋆ ∪ {t_k^⋆}
12:        end for
13:        return T^⋆
14:    end if
15: end procedure
```

Algorithm 1 was introduced in [4] and its running time is $O(|K| \cdot |T| \log |T| + |T|^5)$. Theorem 2 shows that the spKey problem is NP-complete in general, and its proof is in [4]. We reduced the problem of deciding whether $T \models \Sigma$ for a collection Σ of spKey constraints to the problem of finding the maximal common independent set of three or more matroids in [3]. This matroid intersection problem is NP-complete, however the reduction given was not one-to-one, so it did not prove, just hinted the NP-completeness of spKey problem. Finally, by modifying an argument of [22], we could prove the following theorem in [4] by a Karp-reduction of 3SAT to our problem.

Theorem 2. *The strongly possible key satisfaction problem is NP-complete.*

In some special cases, $T \models \Sigma$ can be verified in polynomial time, where Σ is a collection of strongly possible key constraints, as the following Proposition shows.

Proposition 3. *Let us assume that T is a table over schema R, furthermore let $\Sigma = \{sp\langle K_1 \rangle, sp\langle K_2 \rangle, \ldots, sp\langle K_m \rangle\}$ be a collection of spKey constraints. If $K_i \cap K_j = \emptyset$ for $i \neq j$, then $T \models \Sigma$ can be decided in polynomial time.*

4.2 Graph theoretical characterization of strongly possible functional dependencies

In this section, we introduce a graph-theoretical characterization that determines when $T \models X \rightarrow_{sp} Y$. Recall that for a table T over Schema R $T \models X \rightarrow_{sp} Y$ if and only if there exists an spWorld T' of T such that $T' \models X \rightarrow Y$. If $T = \{t_1, t_2, \ldots, t_p\}$ and $T' = \{t'_1, t'_2, \ldots, t'_p\}$ with $t_i \sim_w t'_i$, then t'_i is called an *sp-extension* or in short an *extension* of t_i. Let $X \subseteq R$ be a set of attributes and let $t_i \sim_w t'_i$ such that for each $A \in R$: $t'_i[A] \in VD(A)$, then $t'_i[X]$ is a *strongly possible extension of* t_i *on* X. We exclude any attribute with all `NULL` values, because the visible domain for such attribute is the empty set. We recall the *weak similarity graph* [4], as a useful tool in investigations of strongly possible functional dependencies (spFD's in short).

Definition 4. *Let* $T = \{t_1, t_2, \ldots t_p\}$ *be a table (instance) over schema* R. *The* weak similarity graph G_Y with respect to $Y \subseteq R$ *is defined as* $G_Y = (T, E)$, *where* $\{t_i, t_j\} \in E \iff t_i[Y] \sim_w t_j[Y]$.

Theorem 3 characterizes using weak similarity graphs when $T \models X \rightarrow_{sp} Y$ holds. The theorem was proved in [4].

If $T \models X \rightarrow_{sp} Y$, then there exists an spWorld T' of T such that $T' \models X \rightarrow Y$. This latter one holds iff whenever $t'_1[Y] \neq t'_2[Y]$ then $t'_1[X] \neq t'_2[X]$ is also satisfied. Now, if $t_1, t_2 \in T$ are such tuples that $t_1[Y] \not\sim_w t_2[Y]$, then certainly $t'_1[Y] \neq t'_2[Y]$ holds in any spWorld T'. That is, in order to $T \models X \rightarrow_{sp} Y$ hold t'_1, t'_2 must also satisfy $t'_1[X] \neq t'_2[X]$. Recall that by Definition 4, $t_1[Y] \not\sim_w t_2[Y]$ holds exactly when $\{t_1, t_2\}$ is an edge in the *complement* of the weak similarity graph G_Y. We may think about $t'_i[X]$'s as *colors* assigned to vertices of $\overline{G_Y}$ and then we obtain that this coloring must be proper. Now colors that can be assigned to vertices come from lists special to vertices, namely from the sets *strongly possible extensions of* t *on* X. Let T be a table over schema R, $t \in T$ and $X \subseteq R$. t' is a *strongly possible extension of* t *on* X if $t[X] \sim_w t'[X]$, t' is X-total and $\forall X_i \in X$: $t'[X_i] \in VD_{X_i}$. The following theorem introduced in [4] characterizes when $T \models X \rightarrow_{sp} Y$ holds using weak similarity graphs. It tells us that the existence of proper coloring of $\overline{G_Y}$ using the lists determined by the strongly possible extensions on X is a necessary and sufficient condition for $T \models X \rightarrow_{sp} Y$ to hold.

Let $G(V, E)$ be a graph and L: $V \rightarrow 2^{\mathbb{N}}$ be a mapping that assigns each vertex a *list of colors* $L(v)$. A *list coloring of* G *using lists* $\{L(v)\colon v \in V\}$ is a mapping c: $V \rightarrow \bigcup_{v \in V} L(v)$ such that $c(v) \in L(v)$ and $c(u) \neq c(v)$ if $\{u, v\} \in E$. We use Theorem 3 in proofs of sound interference rules.

Theorem 3. *Let* $T = \{t_1, t_2 \ldots t_m\}$ *be a table over schema* R. *For* $X, Y \subseteq R$, $T \models X \rightarrow_{sp} Y$ *holds iff* $\overline{G_Y}$ *can be list colored using lists* $\{t_i^1, t_i^2 \ldots t_i^{r_i}\}$ *for* $t_i \in T$, *where* $\overline{G_Y}$ *is the complement of the weak similarity graph on* Y *and* t_i^j*'s are the strongly possible extensions of* t_i *on* X.

Table 3 illustrates that sets of tuples that are pairwise weakly similar on Y form a weak similarity clique. Now, tuples from a given clique have a unique non-`NULL`

value in each attribute of Y, unless they all contain `NULL` in that attribute. In both cases, there is a way to extend each tuple that tuples in the same clique become identical on Y. So those tuples that are in one weak similarity clique on Y can be list colored by the same color on X so that $T \models X \rightarrow_{sp} Y$. Note that weak similarity cliques of G_Y are independent vertex sets in $\overline{G_Y}$.

Table 3: Color classes and weak similarity cliques.

X	Y
Same color	Weak Similarity Clique
Same color	Weak Similarity Clique
Same color	Weak Similarity Clique

4.3 Complexity of strongly possible functional dependencies

List coloring problem is NP-complete even if all lists have length three [24]. This suggests that deciding whether a given spFD is satisfied in an SQL table is NP-complete even if all the tuples have a maximum of three extensions. However, it is not obvious the "intractable cases" of list coloring problem really correspond to spFD-satisfaction, so we give a direct proof of NP-completeness.

Definition 5. *The* spFD-Satisfaction *problem is defined as follows.*

Input: *An SQL-table T over schema R, $X, Y \subseteq R$.*

Question: *Does $T \models X \rightarrow_{sp} Y$ hold?*

Theorem 4. *The* spFD-Satisfaction *problem is NP-complete.*

Proof. spFD-Satisfaction is in NP, since an spWorld T' of T such that $T' \models X \rightarrow Y$ is a good witness. It is clearly of polynomial size of the input and whether $T' \models X \rightarrow Y$ holds can be checked in polynomial time by pairwise comparisons of tuples.
In order to prove that spFD-Satisfaction is NP-hard a Karp-reduction from 3-Color to spFD-Satisfaction is given. Let $G = (V, E)$ be an input of 3-Color with $V = \{v_1, v_2, \ldots, v_n\}$. An SQL table T is constructed over schema $R = \{A_0, A_1, \ldots, A_n\}$ of $n + 1$ tuples and $X = \{A_0\}$ and $Y = \{A_1, A_2, \ldots, A_n\}$ is set so that the complement $\overline{G_Y}$ of the weak similarity graph on Y is isomorphic

to G plus three isolated vertices. Let $t_1, t_2, \ldots, t_n$ be defined by induction on i as follows. $t_1[A_1] = 1$. If $t_1, t_2, \ldots, t_i$ are defined for $A_1, A_2, \ldots, A_i$, then let

$$t_{i+1}[A_j] = \begin{cases} \perp & \text{if } 1 \leq j \leq i \\ 1 & \text{if } j = i+1 \end{cases}$$

furthermore, for $j = 1, 2, \ldots, i$

$$t_j[A_{i+1}] = \begin{cases} 2 & \text{if } \{v_j, v_{i+1}\} \in E(G) \\ \perp & \text{if } \{v_j, v_{i+1}\} \notin E(G) \end{cases}$$

Finally, let t_{n+k} be defined for $k = 1, 2, 3$ as $t_{n+k}[A_0] = k$, $t_{n+k}[A_j] = \perp$ for $j = 1, 2, \ldots, n$. Obviously, T can be constructed from G in polynomial time. Our claim is that G is 3-colorable iff $T \models X \rightarrow_{sp} Y$. Indeed, t_{n+k} for $k = 1, 2, 3$ are isolated vertices in $\overline{G_Y}$, so $\overline{G_Y}$ is list-colorable with extensions of t_j over X iff $\overline{G_Y}$ restricted to $W = \{t_1, t_2, \ldots, t_n\}$ is list-colorable. Observe that $\overline{G_Y}|_W$ is isomorphic to $G = (V, E)$, Indeed, for $1 \leq i < j \leq n$ $t_i[Y] \not\sim_w t_j[Y]$ if $\{v_i, v_j\} \in E(G)$, because $t_i[A_j] = 2$ and $t_j[A_j] = 1$, on the other hand if $\{v_i, v_j\} \notin E(G)$, then $t_i[A_j] = \perp$ and $t_j[A_j] = 1$, $t_i[A_i] = 1$ and $t_j[A_i] = \perp$, furthermore $t_i[A_\ell], t_j[A_\ell] \in \{\perp, 2\}$ for $\ell \notin \{i, j\}$. Finally, $VD_X = \{1, 2, 3\}$, so the list of extensions of t_i on X is $\{1, 2, 3\}$ for all $1 \leq i \leq n+3$. Thus any proper list coloring of $\overline{G_Y}$ with extensions of t_i on X gives a 3-coloring of G, and vice versa, any 3-coloring of G can easily be extended to a proper list coloring of $\overline{G_Y}$ with extensions of t_i . □

However, this problem can be solved in polynomial time for some special cases. Complete graphs can be reduced to the verifying spKey problem, as the following proposition shows.

Proposition 4. *$T \models X \rightarrow_{sp} Y$ can be decided in polynomial time if $\overline{G_Y}$ is a complete graph.*

Proof. As $\overline{G_Y}$ is a complete graph, then the tuples are pairwise distinct by some non-`NULL` value on Y, then, all the tuples in T should be pairwise distinctly colored on X to satisfy the spFD $X \rightarrow_{sp} Y$. That is, $T \models X \rightarrow_{sp} Y \iff T \models sp\langle X \rangle$ that can be checked in polynomial time using Algorithm 1. □

Greedy algorithm can be applied to find a proper list coloring of a graph $G = (V, E)$ if for $\forall v \in V$ the list size of v is at least $d_G(v) + 1$. This can naturally be applied in our context, as well, that is $T \models X \rightarrow_{sp} Y$ can be decided in polynomial time if $\forall t \in T$, number of all extensions of t on X is larger than the number of weakly similar tuples to t on Y.
In order to apply list coloring, first the lists should be generated. However, the number of X-extensions of t of a tuple $t \in T$ may easily be exponential function of the size of the input as it was seen in Section4.1. This obstacle can be overcome by the following observation since it is enough to generate at most $\Delta(\overline{G_Y}) + 1$ X-extensions for each tuple t.

Proposition 5. *There exists a list coloring with the lists generated for each tuple of size less than or equal* $\Delta(\overline{G_Y})+1$ *if and only if there exists list coloring with full lists.*

Proof. $\Rightarrow$ Indeed, same list coloring can be used.
$\Leftarrow$ Assume there exists list coloring with full lists. And let $\{t_1, t_2, \ldots, t_r\}$ be set of tuples with number of extensions on X at most $\Delta(\overline{G_Y})$ and let $\{t_{r+1}, t_{r+2}, \ldots, t_s\}$ be the set of other tuples. Take $\Delta(\overline{G_Y})+1$ elements lists for $\{t_{r+1}, t_{r+2}, \ldots, t_s\}$. Then, keep the coloring of $\{t_1, t_2, \ldots, t_r\}$ and then color $\{t_{r+1}, t_{r+2}, \ldots, t_s\}$ using greedy algorithm. □

If $\overline{G_Y}$ is a tree, the list coloring problem can be solved using dynamic programming techniques in polynomial time [20]. Generally, list coloring problem restricted to graphs of maximum degree two, such as cyclic graphs, is polynomially solvable [24].

Weak similarity graphs can be used on the LHS of an spFD in a special case. Namely, if components of weak similarity graph G_X with respect to X are cycles of length at least 4, then $T \models sp\langle X\rangle$, in particular for any $Y \subseteq R$, $T \models X \rightarrow_{sp} Y$ holds.

Proposition 6. *If each connected component* C *of the weak similarity graph* G_X *with respect to* X *is a cycle of length* ≥ 4*, then* $sp\langle X\rangle$ *holds.*

Proof. Let $T = \{t_1, t_2, \ldots, t_p\}$ be a table over schema R such that each component C of the weak similarity graph G_X with respect to X is a cycle of length ≥ 4. We need NULL-free tuples from $t'_i \in VD_1 \times VD_2 \times \ldots \times VD_n$ such that $t_i[X] \sim_w t'_i[X]$ and $t'_i[X]$'s are pairwise distinct in order to $sp\langle X\rangle$ hold. Since two tuples can only have identical extensions on X if they are weakly similar on X, it is enough to construct $t'_i[X]$'s for each component of G_X separately. If this single component is a circle ($t_1[X] \sim_w t_2[X] \sim_w t_3[X] \sim_w \ldots \sim_w t_k[X] \sim_w t_1[X]$), then any extension of t_2 is distinct on X from any extensions of $t_4 \ldots t_k$. There exist $A \in X$ such that $t_1[A] \neq t_3[A]$ and both $t_1[A], t_3[A] \neq \perp$, because $t_1[X] \not\sim_w t_3[X]$. We need to make t'_2 different from t'_1 and t'_3, so that we can set $t'_2[A] = t_3[A]$ and this distinguishes it from t'_1. Applying the same idea around the cycle, $t_i[X] \sim_w t_{i+1}[X] \sim_w t_{i+2}[X]$ will make t'_i and t'_{i+1} distinct. □

5 Implications among strongly possible functional dependencies

This section treats the implication properties of spFD's. The cases when one side of the dependency is a fixed attribute set is characterized, then axioms and rules of weak, strong, possible and certain FD's are analysed with respect to spFD's. In addition to that, the interactions with different key concepts are treated. Finally, a complete characterization of the singular attribute case is given.

5.1 Strongly possible functional dependencies with one side fixed

For $T \models X \to_{sp} Y$, fixing the left-hand side of the spFD, makes the right-hand sides form a down-set, i.e, if $Y' \subset Y$, then $T \models X \to_{sp} Y'$ also holds. We introduced Proposition 7 in a proceedings paper [4], we repeat it here to fix an error in the proof of that version. It shows that for fixed left-hand side of spFD's, there is no other restriction for the right-hand sides than forming a down-set.

Proposition 7. *Let (R, R_S) be a schema and $X \not\subseteq R_S$. Let $\mathcal{Y}$ be a down-set of subsets of $R \setminus X$. Then there exists a table T over (R, R_S) such that $T \models X \to_{sp} Y$ holds iff $Y \cap (R \setminus X) \in \mathcal{Y}$.*

Proof. Let the maximal elements of $\mathcal{Y}$ be $Y_1, Y_2, \ldots, Y_s$, that is $\mathcal{Y} = \{A : \exists i\ A \subseteq Y_i\}$ and $Y_i \not\subseteq Y_j$ for $i \neq j$. Let $A_0 \in X \setminus R_S$ be a fixed attribute, and $A_1, A_2, \ldots A_n$ be the other attributes of R. Table T contains tuples $t_0, t_1, \ldots t_s$ such that

$$t_0[A_i] = \begin{cases} \perp \text{ if } i = 0 \\ 0 \text{ if } i > 0 \end{cases}$$

and

$$t_i[A_j] = \begin{cases} 0 \text{ if } A_j \in Y_i \cup X \\ i \text{ if } A_j \notin Y_i \cup X \end{cases} \quad \text{for } i = 1, 2, \ldots, s \text{ and } j = 1, 2, \ldots, n$$

finally $t_i[A_0] = i$ for $i = 1, 2, \ldots, s$. So, the table is constructed as follows.

A_0	A_1			$\ldots$			A_n
$\perp$	0			$\ldots$			0
i	0	$\ldots$		0	i	$\ldots$	i

$$\underbrace{\qquad\qquad\qquad\qquad}_{Y_i \cup X}$$

$Y \in \mathcal{Y} \iff \exists 1 \leq i_Y \leq s \colon Y \subseteq Y_{i_Y}$, so FD $X \to Y$ holds in the spWorld obtained by replacing $\perp$ in $t_0[A_0]$ by i_Y, because only tuples t_0 and t_{i_Y} can agree in X. On the other hand, if $Y \notin \mathcal{Y}$, then for every $1 \leq i \leq s$ there exists an attribute $A_{j_i} \in Y \setminus Y_i$, so whichever element $i \in VD_{A_0}$ is put in place of $\perp$ in $t_0[A_0]$, we get that $t_0[X] = t_i[X]$, but $t_0[Y] \neq t_i[Y]$, hence $T \not\models X \to_{sp} Y$. □

We can characterize the case of fixed right-hand side of an spFD as well, as it is clear that for a fixed set $Y \subset R$, the collection of attribute sets $\mathcal{X} = \{X \colon T \models X \to_{sp} Y\}$ forms an up-set, where T is a table over a schema R. However, this condition is the only one we have. This is shown in Proposition 8 and Theorem 5 proved in [4]. We repeat them here for the sake of completeness.

Proposition 8. *Let (R, R_S) be a schema, $Y \subset R$ be a fixed set of attributes, furthermore let $\mathcal{X}$ be an upset of subsets of $R \setminus Y$. Then there exists a table T over (R, R_S) such that $T \models X \rightarrow_{sp} Y \iff X \in \mathcal{X}$.*

The proof uses the Armstrong instance construction for strongly possible keys from [3].

Theorem 5. *Suppose that $\Sigma = \{sp\langle K\rangle : K \in \mathcal{K}\}$ is a collection of spKey constraints such that if $|K| = 1$, then $K \subseteq R_S$. Then, there exists an Armstrong table for (R, R_S, Σ).*

5.2 Strongly possible functional dependencies axiomatisation

The first steps towards a possible axiomatisation of spFD's were given in [5]. We studied and analyzed the axioms of weak/strong FD's given by Levene and Loizou [25] and also the axioms of certain FD's given by Köhler and Link [23] in context of spFD's. The investigation showed that some of these axioms are not sound for spFD's. Table 4 shows the axioms that are still sound for spFD's and it also shows several possible weakenings and restrictions that keep soundness for those that are not. All proofs and counterexamples are detailed in [4, 5]. We repeat them in this paper to be self-contained.

We may conclude from Table 4 that more than one spFD in the premise of a rule usually cause problems in soundness. This is caused by the fact that a single spWorld may not satisfy the different spFD's due to the limitations of visible domains. This problem must be handled for a complete axiomatization. Particularly, the fact that composition rule is not sound in general makes usual proof methods of completeness virtually unusable.

Table 4: spFD Axioms Soundness

Axiom	**spFD Soundness**
Reflexivity	**Sound:** If $Y \subseteq X \subseteq R$ then $T \models X \rightarrow_{sp} Y$
Augmentation	**Sound:** If $T \models X \rightarrow_{sp} Y$ and $W \subseteq R$, then $T \models XW \rightarrow_{sp} YW$
Union	**Not Sound:** If $T \models X \rightarrow_{sp} Y$ and $T \models X \rightarrow_{sp} Z$, then $T \models X \rightarrow_{sp} YZ$ Possible weakenings are: • **Strong FD Mixed-union:** If $T \models \Box X \rightarrow Y$ and $T \models X \rightarrow_{sp} Z$, then $T \models X \rightarrow_{sp} YZ$ or if $T \models X \rightarrow_{sp} Y$ and $T \models \Box X \rightarrow Z$, then $T \models X \rightarrow_{sp} YZ$ • **Certain FD Mixed-union:** If $T \models X \rightarrow_{sp} Y$ and $T \models X \rightarrow_{w} Z$, then $T \models X \rightarrow_{sp} YZ$ • **`NULL`-free union:** If $T \models X \rightarrow_{sp} Y$ and $T \models X \rightarrow_{sp} Z$ and $X \subseteq R_S$, then $T \models X \rightarrow_{sp} YZ$

Transitivity	**Not Sound:** If $T \models X \rightarrow_{sp} Y$ and $T \models Y \rightarrow_{sp} Z$, then $T \models X \rightarrow_{sp} Z$ Possible weakenings are: • **Strong FD Mixed-transitivity:** If $T \models X\Box \rightarrow Y$ and $T \models Y \rightarrow_{sp} Z$ or if $T \models X \rightarrow_{sp} Y$ and $T \models \Box Y \rightarrow Z$, then $T \models X \rightarrow_{sp} Z$. • **Certain FD Mixed-transitivity:** If $T \models X \rightarrow_{sp} Y$ and $T \models Y_{\,w} \rightarrow Z$, then $T \models X \rightarrow_{sp} Z$ • **Sp-transitivity:** If $T \models X \rightarrow_{sp} Y$ and $T \models Y \rightarrow_{sp} Z$, and $Y \subseteq R_s$, then $T \models X \rightarrow_{sp} Z$
Pseudo-transitivity	**Not Sound:** If $T \models X \rightarrow_{sp} Y$ and $T \models YZ \rightarrow_{sp} V$, then $T \models XZ \rightarrow_{sp} V$ or if $T \models X \rightarrow_{sp} YZ$ and $T \models Y \rightarrow_{sp} V$, then $T \models X \rightarrow_{sp} ZV$ Possible weakenings are: • **Strong FD Mixed-pseudo-transitivity:** If any of the given spFD's is strong FD, it is sound. • **Certain FD Mixed-pseudo-transitivity I:** If $T \models X \rightarrow_{sp} Y$ and $T \models XY_{\,w} \rightarrow Z$, then $T \models X \rightarrow_{sp} Z$ • **Certain FD Mixed-pseudo-transitivity II:** If $T \models X \rightarrow_{sp} Y$ and $T \models YZ_{\,w} \rightarrow V$, then $T \models XZ \rightarrow_{sp} V$ • `NULL`**-free pseudo-transitivity:** If $T \models X \rightarrow_{sp} YZ$ and $T \models Y \rightarrow_{sp} V$ and $Y \subseteq R_S$, then $T \models X \rightarrow_{sp} ZV$.
Composition	**Not Sound:** If $T \models X \rightarrow_{sp} Y$ and $T \models A \rightarrow_{sp} B$, then $T \models XA \rightarrow_{sp} YB$ Possible weakenings are: • **Mixed composition:** If $T \models \Box X \rightarrow Y$ and $T \models A \rightarrow_{sp} B$, or if $T \models X \rightarrow_{sp} Y$ and $T \models \Box A \rightarrow B$, then $T \models XA \rightarrow_{sp} YB$. • `NULL`**-free composition:** If $T \models X \rightarrow_{sp} Y$ and $T \models A \rightarrow_{sp} B$ and $YA \subseteq R_S$, then $T \models XA \rightarrow_{sp} YB$. • **Disjoint composition:** If $T \models X \rightarrow_{sp} Y$ and $T \models A \rightarrow_{sp} B$ and $X \cap A = \emptyset$, then $T \models XA \rightarrow_{sp} YB$.
Decomposition	**Sound:** If $T \models X \rightarrow_{sp} YZ$, then $T \models X \rightarrow_{sp} Y$ and $T \models X \rightarrow_{sp} Z$.

5.2.1 sp-keys and sp-FD/c-FDs interaction

For a relation R, the ordinary functional dependency $X \rightarrow R$ implies that X is a key in R, because duplicate rows are prohibited in the relational model [9]. On the other hand, for an SQL table T, the spFD $X \rightarrow_{sp} T$ does not imply the spKey $sp\langle X\rangle$, because duplicate rows are permitted in the bag semantics of data in SQL tables.

We study interaction between sp-keys, certain keys, spFD's and c-FDs in the following.

- **sp-key/spFD Weakening:** $\frac{sp\langle X\rangle}{(X\rightarrow_{sp}Y)}\ \forall Y \subseteq R$
  Indeed, if there exists an spWorld such that all the tuples are pairwise distinct on X, then $X \rightarrow R$ holds in that spWorld.

- **certain key/spFD Transitivity:** $\frac{(X\rightarrow_{sp}Y)\ c\langle XY\rangle}{sp\langle X\rangle}$
  The certain key $c\langle XY\rangle$ implies that all the tuples are not pairwise weakly similar on XY. Then, for every two tuples, they either have distinct and non-`NULL` values on X or on Y. But we have $X \rightarrow_{sp} Y$, then $\overline{G_Y}$ can be list colored by X extensions, and we can extend the extensions of this coloring in any way to R. So, by using this coloring, $sp\langle X\rangle$ is satisfied.
  However, the sp-key/FD Transitivity true in following form is not sound: $\frac{(X\rightarrow_{sp}Y)\ sp\langle XY\rangle}{sp\langle X\rangle}$. The reason is the fact that there will be a possibility of encountering a weak similarity on X and distinctness on Y, as illustrated in Table 5.

Table 5: sp-key/spFD Transitivity dissatisfaction

	X	**Y**
1	1	1
1	$\perp$	1
1	2	2

5.3 spFDs for singular attributes

Substituting a value from the visible domain in place of a $\perp$ produces a duplication in that attribute, since for a singular attribute, the visible domain represents all the possible replacements for any $\perp$ occurrence. For example, in Table 6, all possible substitutions of the `NULL` in the second tuple on X are $\{1, 2, 3\}$, as they form VD_A. A singular attribute can only be an sp-key if there are no $\perp$'s in that attribute. However, spFD's are possible between singular attributes with `NULL`'s. Any occurrence of a `NULL` in the LHS of an spFD for a singular attribute causes a duplication, and this requires a corresponding duplication possibility on the RHS to satisfy the spFD. In the example in Table 6, $X \rightarrow_{sp} Y$ holds because, in the first and the third tuples, there is a duplication possibility on Y for the `NULL` replacement on X in the second tuple. On the other hand, $Y \not\rightarrow_{sp} X$ holds, because any replacement for the `NULL` in the third tuple on Y will not get a corresponding duplication possibility on X.
In the present subsection, we study the case of singular attributes as a special case of spFD implication. The following proposition shows a bidirectional property for singular attributes spFD.

Table 6: spFD for Singular Attributes

X	**Y**
1	1
$\perp$	1
2	$\perp$
3	2

Proposition 9. *For singular attributes X and Y, if $T \models X \to_{sp} Y$ and $|VD_X| = |VD_Y|$, then $T \models Y \to_{sp} X$*

Proof. We may assume without loss of generality that $VD_X = \{1, 2, \dots \ell\}$. Since $T \models X \to_{sp} Y$, for every $i \in \{1, 2, \dots \ell\}$ if $t \in T$ is a tuple, then $t[X] = i$ implies that $t[Y] = a_i$ or $t[Y] = \perp$. Also, let $\{b_1, b_2, \dots, b_r\} \subset VD_Y$ be those pairwise distinct visible domain values that satisfy $t[Y] = b_j \Rightarrow t[X] = \perp$. Assume that for $1 \leq i \leq k$ we have non-`NULL` a_i such that there exists a tuple t with $t[X] = i$ and $t[Y] = a_i$, that is, for $k+1 \leq j \leq \ell$ we have $t[X] = j \Rightarrow t[Y] = \perp$. Table 7 shows the possible types of tuples on $\{X, Y\}$ If $a_i \neq a_j$ for $i \neq j$, then by $|VD_X| = |VD_Y|$ we have that $r = \ell - k$ and the `NULL`'s can easily be substituted in these two columns so that $1 \leq i \leq k$ is matched with a_i and $k+1 \leq j \leq \ell$ is matched with b_{j-k} so the two columns are basically identical, that is $Y \to_{sp} X$ holds. On the other hand, if $a_i = a_j$ for some $i \neq j$, then we have that $r > \ell - k$ since the sizes of divisible domains of X and Y are the same. Consider a spWorld T' of T that satisfies $T' \models X \to Y$ functional dependency. Such a T' exists, since $T \models X \to_{sp} Y$. Now, the `NULL`'s in the tuples that have value b_j in Y can only be substituted by values from $\{k+1, \dots \ell\}$, because for $1 \leq i \leq k$ there are tuples with non-`NULL` values $t[X] = i$ and $t[Y] = a_i$. However, as $r > \ell - k$, we must have $1 \leq u < v \leq r$ and $k+1 \leq j \leq \ell$ so that we have two tuples t, t' with $t[X] = t'[X] = j$ and $t[Y] = b_u$ and $t'[Y] = b_v$ contradicting to $T' \models X \to Y$. □

Example 2. The table below shows an instance T with $T \models X \to_{sp} Y$ and $|VD_X| = |VD_Y| = 2$. Value 2 in attribute X is shown only in t_4 where $t_4[Y] = \perp$. Then, to have $|VD_X| = |VD_Y|$, we have value a_2 in attribute Y is shown only in tuples having $\perp$ on X. Then, X and Y are two singular attributes satisfy a bidirectional implication property.

Proposition 10. *For singular X and Y, if $T \models X \to_{sp} Y$, then$|VD_X| \geq |VD_Y|$.*

Proof. Let T' be a spWorld of T that satisfies functional dependency $X \to Y$ that exists by $T \models X \to_{sp} Y$. Observe that the set of values appearing in column X of T' is VD_X, while that of values appearing in column Y is VD_Y. Since if $t'_1[X] = t'_2[X]$ implies $t'_1[Y] = t'_2[Y]$ for any $t'_1, t'_2 \in T'$, the mapping $f \colon VD_Y \to VD_X$ given by $f(v') = v$ if there is a tuple $t' \in T'$ such that $t'[X] = v$, $t'[Y] = v'$ is well defined and is an injection. □

In the next proposition, we show the bidirectional implication of the spFD in the singular attributes directed graph of spFD's.

Proposition 11. *Let T be an instance over the relation R. If the spFD's between singular attributes $X_i \in R$, for $i = 1, 2 \ldots, w$, form a directed circle in the spFD graph, that is $T \models X_i \rightarrow_{sp} X_{i+1}$ and $T \models X_w \rightarrow_{sp} X_1$. Then the reverse direction of the spFD circle also holds in T, i.e. $T \models X_{i+1} \rightarrow_{sp} X_i$ and $T \models X_1 \rightarrow_{sp} X_w$.*

Proof. As the spFD's form a circle in the spFD graph, then all the attributes have the same number of values in their visible domains. Indeed, by $T \models X_i \rightarrow_{sp} X_{i+1}$, we have $|VD_{X_i}| \geq |VD_{X_{i+1}}|$ for any i, and by $T \models X_w \rightarrow_{sp} X_1$, we have $|VD_{X_w}| \geq |VD_{X_1}|$. So, $|VD_{X_i}| = |VD_{X_{i+1}}|$ for any i. Hence, by Proposition 9, the other direction for each spFD is also satisfied by T. □

The bidirectional implication of the spFD between the singular attributes does not show that the complements of their weak similarity graphs are the same, For example, $Y \rightarrow_{sp} Z$ and $Z \rightarrow_{sp} Y$ hold in the table below, but $\overline{G_Y} \neq \overline{G_Z}$. On the

Table 7: Possible types of tuples

X	Y
1	a_1
1	$\perp$
2	a_2
2	$\perp$
$\vdots$	$\vdots$
k	a_k
k	$\perp$
$k+1$	$\perp$
$\vdots$	$\vdots$
ℓ	$\perp$
$\perp$	a_1
$\vdots$	$\vdots$
$\perp$	a_k
$\perp$	b_1
$\vdots$	$\vdots$
$\perp$	b_r

	X	Y
t_1	1	$\perp$
t_2	$\perp$	a_2
t_3	1	a_1
t_4	2	$\perp$
t_5	$\perp$	a_2
t_6	$\perp$	$\perp$

other hand, $X \rightarrow_{sp} Y$ and $X \rightarrow_{sp} Z$ hold, but $X \not\rightarrow_{sp} YZ$.

X	Y	Z
1	1	$\perp$
$\perp$	2	1
2	$\perp$	2

Propositions 9 and 11 allow us to introduce the following rule.

Digraph rule Let $G = (R, E)$ be a directed graph If $(A_i, A_j) \in E$ and A_i and A_j are in the same strongly connected component of G, then $(A_j, A_i) \in E$ holds, as well.

Theorem 6. *Let T be an SQL table over scheme $R = \{A_1, A_2, \ldots, A_n\}$. If $G = (R, E)$ is a directed graph defined by $(A_i, A_j) \in E \iff T \models \{A_i\} \rightarrow_{sp} \{A_j\}$ for attributes $A_i, A_j \in R$, then $G = (R, E)$ satisfies the Digraph rule. Furthermore, for every directed graph $G = (R, E)$ that satisfies the Digraph rule there exists an SQL table T such that $(A_i, A_j) \in E \iff T \models \{A_i\} \rightarrow_{sp} \{A_j\}$.*

The proof of Theorem 6 is based on a series of propositions.

Proposition 12. *Let T be an SQL table over scheme $R = \{A_1, A_2, \ldots, A_n\}$ and let $G = (R, E)$ be the directed graph defined by $(A_i, A_j) \in E \iff T \models \{A_i\} \rightarrow_{sp} \{A_j\}$ for attributes $A_i, A_j \in R$. Then $G = (R, E)$ satisfies the Digraph rule.*

Proof. Let $(A_i, A_j) \in E$ be an edge of the directed graph defined by spFD's between singular attributes so that A_i and A_j are in the same strongly connected component of G. This means that there exists a directed path $A_j = X_1, X_2, \ldots X_w = A_i$ in G, thus together with edge (A_i, A_j) a directed cycle is obtained. So by Proposition 11, the other direction of the spFD's hold, i.e $T \models A_j \rightarrow_{sp} A_i$. □

Proposition 13. *Let $G = (R, E)$ be a strongly connected graph that satisfies the Digraph rule. Then there exists an SQL table T over schema $R = \{A_1, A_2, \ldots, A_n\}$ such that $(A_i, A_j) \in E \iff T \models \{A_i\} \rightarrow_{sp} \{A_j\}$. Furthermore, we may assume that $VD(A_i) = \{1, 2\}$ for all $i = 1, 2, \ldots, n$ in T and that if $(A_i, A_j) \notin E$, then there exists two rows of T that agree and non-null in A_i and differ and non-null in A_j.*

Proof. Note that a strongly connected graph $G = (R, E)$ satisfies the Digraph rule iff $(A_i, A_j) \in E \iff (A_j, A_i) \in E$. In particular, for any *induced* subgraph $G' = (R', E')$ of $G = (R, E)$ also satisfies the Digraph rule. We may assume without loss of generality that $(A_1, A_2) \in E$. Let $G_k = (R_k, E_k)$ be the subgraph of $G = (R, E)$ induced by $R_k = \{A_1, A_2, \ldots, A_k\}$. We use induction on k to prove that there exists an SQL table T_k over R_k such that $(A_i, A_j) \in E_k \iff T_k \models \{A_i\} \rightarrow_{sp} \{A_j\}$ and if $(A_i, A_j) \notin E$, then there exists two rows of T_k that agree and non-null in A_i and differ and non-null in A_j. The base case $k = 2$ is trivial

$$T_2 = \begin{array}{|c|c|}\hline A_1 & A_2 \\ \hline 1 & 1 \\ 2 & \bot \\ \bot & 2 \\ \hline \end{array}$$

so $T_2 = \{t_0, t_1, t_2\}$ with $t_0[A_j] = 1$ for $j \in \{1, 2\}$ and

$$t_i[A_j] = \begin{cases} 2 & \text{if } i = j \\ \bot & \text{if } i \neq j \end{cases}.$$

Note that T_2 has one entry 2 in each column, and these entries are in different rows, in column A_i the 2 is in row t_i. Now assume that $T_k = \{t_0, t_1, t_2, \ldots, t_k\}$ exists for $G_k = (R_k, E_k)$ such that column A_i has its only entry 2 in row t_i and consider G_{k+1}. We add new column A_{k+1} and a new row (tuple) t_{k+1} to T_k as follows.

$$t_i[A_{k+1}] = \begin{cases} 1 & \text{if } (A_{k+1}, A_i) \notin E_{k+1} \\ \bot & \text{if } (A_{k+1}, A_i) \in E_{k+1} \end{cases} \quad \text{for } i = 1, 2, \ldots, k \text{ and } t_{k+1}[A_{k+1}] = 2.$$

Furthermore

$$t_{k+1}[A_i] = \begin{cases} 1 & \text{if } (A_i, A_{k+1}) \notin E_{k+1} \\ \bot & \text{if } (A_i, A_{k+1}) \in E_{k+1} \end{cases} \quad \text{for } i = 1, 2, \ldots, k$$

$T_{k+1} =$

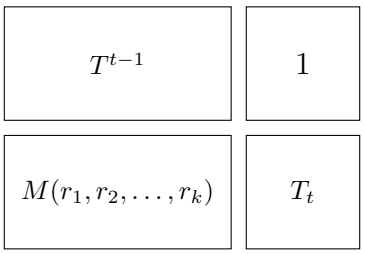
A_1	A_2	$\ldots$	A_k	A_{k+1}
1	1	$1 \ldots 1$	1	1
2	$\perp$	$1 \ldots 1$	1	1
$\perp$	2	$1 \ldots \perp$	1	$\perp$
1	$\perp$	$2 \ldots 1$	$\perp$	$\perp$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\perp$	1	$1 \ldots \perp$	2	1
1	$\perp$	$1 \ldots \perp$	1	2

This table satisfies the requirements, since no $A_i \rightarrow_{sp} A_j$ for $1 \leq i, j \leq k$ is destroyed by row t_{k+1} by imputing 1's in place of $\perp$'s in the last row. Similarly, if we impute a 2 in place of $\perp$ in the last column in row t_i and everywhere else 1's in that column, then we get a strongly possible world showing $A_{k+1} \rightarrow_{sp} A_i$. Furthermore imputing 2 in place of $\perp$'s in t_{k+1} we get $A_i \rightarrow_{sp} A_{k+1}$ exactly if $(A_i, A_{k+1}) \in E_{k+1}$. □

Proof. (of Theorem 6). We use induction on the number t of strongly connected components of $G = (R, E)$. The base case $t = 1$ is the statement of Proposition 13. Let the strongly connected components of $G = (R, E)$ be $C_1, C_2, \ldots, C_t$, we may assume without loss of generality, that they are in a topological sort order, that is if $(A_i, A_j) \in E$ and $A_i \in C_i$ and $A_j \in C_j$, then $i \leq j$. There exists an SQL table T_i for each C_i such that $\{A, B\} \in E(C_i) \iff T_i \models A \rightarrow_{sp} B$. Furthermore, by Proposition 13 we have that entries in T_i are $1, 2, \perp$, if $A \rightarrow_{sp} B$, 1's in column A match 1's or $\perp$ in B, and 2's are not in the same row with a 1 in these 2 columns. If $A \not\rightarrow_{sp} B$, then there exist two rows as following:

A	B
1	1
1	2

Assume by induction, we have a table T^{t-1} for the subgraph induced by $C_1 \cup C_2 \cup ... \cup C_{t-1}$. Construct T^t as follows. First position T^{t-1} and T_t on disjoint row and column sets so that T^{t-1} is extended by 1's in the columns of C_t. In the rows of T_t put matrix $M(r_1, r_2, \ldots, r_k)$, which has entry r_i in its i^{th} row for each column of $C_1 \cup C_2 \cup ... \cup C_{t-1}$, where $r_1, r_2, \ldots r_k$ are pairwise distinct numbers, different from anything appearing in T^{t-1}:



Then we add two rows, $\mathbf{t_1^A}$ and $\mathbf{t_2^A}$ for each $A \in C_1 \cup C_2 \cup ... \cup C_{t-1}$ as follows, $\forall B \in C_1 \cup C_2 \cup ... \cup C_{t-1}$ and for all $X, Y \in C_t$ if $A \rightarrow_{sp} X$ and $A \not\rightarrow_{sp} Y$.

	A	B	X	Y
$\mathbf{t_1^A}$	r_A	$\perp$	$\perp$	1
$\mathbf{t_2^A}$	r_A	$\perp$	$\perp$	2

where r_A is a new value not appearing in T^{t-1}, also $r_A \neq r_i$ $\forall i = 1, 2, \ldots, k$ and if $A \neq B$ then $r_A \neq r_B$.

Let T^t be the table obtained. We check pairs of attributes A, X that $T^t \models A \rightarrow_{sp} X \iff (A, X) \in E$.

Case 1: Both $A, X \in C_1 \cup C_2 \cup ... \cup C_{t-1}$. If $(A, X) \notin E$, then by the induction hypothesis there exist two rows of T^{t-1} that are both not null in both A and X and agree in A and differ in X, so $T^t \not\models A \rightarrow_{sp} X$. On the other hand, if $(A, X) \in E$, then the induction hypothesis provides an spWorld T^* of T^{t-1} that $T^* \models A \rightarrow X$. Columns A and X can be extended identically below T^{t-1} so that values on those rows in A and X differ from values in T^*, thus imputing any values in place of nulls in the other columns we get an spWord T' of T^t such that $T' \models A \rightarrow X$.

Case 2: Both $A, X \in C_t$. If $(A, X) \notin E$, then by Proposition 13 there are two rows of T_t that they are both 1 in A and take different values (1 and 2) in X, so $T^t \not\models A \rightarrow_{sp} X$. On the other hand, if $(A, X) \in E$, then there is an spWord of T_t that has columns A and X identical. The rows above T_t are also identical in A and X and these two columns can be extended identically below T_t, as well so that 1's in A match 1's in X and the same holds for entries 2. Then other null values of T^t can be arbitrarily substituted from visible domains of the given attributes and an spWord is obtained where $A \rightarrow X$ holds, that is $T^t \models A \rightarrow_{sp} X$.

Case 3: $A \in C_1 \cup C_2 \cup ... \cup C_{t-1}$ and $X \in C_t$. In this case $(X, A) \notin E$ holds, and there are two rows in T^{t-1} that have different not null values in A, but these two rows take value 1 in X, so $T^t \not\models X \rightarrow_{sp} A$. If If $(A, X) \notin E$, then rows $\mathbf{t_1^A}$ and $\mathbf{t_2^A}$ have identical values in A but different values in X, so $T^t \not\models A \rightarrow_{sp} X$. On the other hand, if $(A, X) \in E$, then the nulls of X in rows $\mathbf{t_1^A}$ and $\mathbf{t_2^A}$ can be substituted by 1's, similarly for any rows $\mathbf{t_1^B}$ and $\mathbf{t_2^B}$ that contain nulls in X. If for some $B \in C_1 \cup C_2 \cup ... \cup C_{t-1}$ the corresponding rows $\mathbf{t_1^B}$ and $\mathbf{t_2^B}$ contain 1 and 2 in column X (i.e. $(B, X) \notin E$), then in $\mathbf{t_1^B}$ we substitute 1 in column A,

and in $\mathbf{t_2^B}$ substitute r_i in place of the null in column A, where r_i is the value in $M(r_1, r_2, \ldots, r_k)$ that stands in the unique row of T_t which has a 2 in column X. The rest of the nulls of T^t can be imputed arbitrarily from the appropriate visible domains, the spWord of T_t obtained satisfies functional dependency $A \to X$. □

6 Conclusions

Entering incomplete tuples are allowed into many recent systems' databases. Several research work studied the keys and functional dependencies constraints over incomplete data, such as p/c-keys [22], spKeys [3], c-FDs [23], p-FDs[26], and s/w-FDs [25].

This paper continued the research work started in [3] of strongly possible worlds of SQL tables with `NULL`'s. We introduced a polynomial-time algorithm that checks whether a given set K of attributes is a spKey or not. We also proved that it is NP-complete to do the same for an arbitrary system of $\Sigma = \{sp\langle K_i\rangle \colon i = 1, 2, \ldots n\}$ of spKey constraints. On the other hand, we showed that Σ can be verified in polynomial time if the sets K_i are pairwise disjoint. Further, we studied strongly possible functional dependencies that were introduced in [4] earlier, as a functional dependency constraint in an SQL table containing NULL's, using the visible domain concept. A graph-theoretical characterization using list coloring is given that can be employed to check when a given spFD holds. This characterization allowed us to prove that verifying whether a single spFD $X \to_{sp} Y$ holds is NP-complete. This is in sharp contrast with spKey problem, where a single key can be checked in polynomial time.

In another paper [5] spFD properties and axioms analogous to those of weak and strong FD's given by Levene et.al. are introduced and suitable weakenings were given for those axioms that are non sound for spFD's. Here we summarize those in a table for completeness and extend the investigation on sp-Keys and sp/c-FDs together to obtain weakening and transitivity interaction rules.

Our study shows that the spFD's between singular attributes have special properties, because the visible domain represents all the possible extensions for any `NULL` occurrence. There is a natural correspondence between directed graphs and single attribute spFD's, a characterization of those directed graphs that may occur in this context was given.

The properties of spFD's listed form a step toward a possible axiomatization of spFD's. The main challenge of applying the visible domain is that different spFD's in the premises of rules may hold in different and incompatible strongly possible worlds. More investigation is needed to find how to incorporate this into the axiom system. A first step would be to resolve the following open problem.

Question 1. *Let us assume that $R_S = \emptyset$. Do Reflexivity, Augmentation, Decomposition, Disjoint composition, and Digraph rule form a complete system of inference rules for strongly possible functional dependencies in this case?*

A further simplification that could be attacked using Theorem 3 is when we

assume that there exist no spFD's between singleton attributes. Our experience shows that in that case spFD's are mostly independent of each other, so the first four rules of the previous question could be a complete set for inferences. Another future research direction is defining the closures concerning spFD's. Where the usual definition of $X^+ = \{A\colon T \models X \rightarrow_{sp} A\}$ may result in $T \not\models X \rightarrow_{sp} X^+$ for spFD's, as the union rule is not sound.
Finally, for database tables, lossless decomposition is an important application of FD's to eliminate redundancy and the possibilities of inconsistent updates. Köhler and Link[23] and Wei and Link [28] show how to use c-FD's or embedded FD's for that. Our future work will include a similar investigation for spFD's.

References

[1] Achs, Ágnes and Kiss, Attila. Fuzzy extension of datalog. *Acta Cybernetica*, 12(2):153–166, 1995. URL: `https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3454`.

[2] Acuna, Edgar and Rodriguez, Caroline. The treatment of missing values and its effect on classifier accuracy. In *Classification, clustering, and data mining applications*, pages 639–647. Springer, 2004. DOI: `10.1007/978-3-642-17103-1_60`.

[3] Alattar, Munqath and Sali, Attila. Keys in relational databases with nulls and bounded domains. In *European Conference on Advances in Databases and Information Systems*, pages 33–50. Springer, 2019. DOI: `10.1007/978-3-030-28730-6_3`.

[4] Alattar, Munqath and Sali, Attila. Functional dependencies in incomplete databases with limited domains. In *International Symposium on Foundations of Information and Knowledge Systems*, pages 1–21. Springer, 2020. DOI: `10.1007/978-3-030-39951-1_1`.

[5] Alattar, Munqath and Sali, Attila. Towards an axiomatization of strongly possiblefunctional dependencies. *The Vietnam Journal of Computer Science*, 8(1):133–151, 2021. DOI: `10.1142/S2196888821500056`.

[6] Atzeni, Paolo and Morfuni, Nicola M. Functional dependencies and constraints on null values in database relations. *Information and Control*, 70(1):1–31, 1986. DOI: `10.1016/S0019-9958(86)80022-5`.

[7] Chang, Gang and Ge, Tongmin. Comparison of missing data imputation methods for traffic flow. In *Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE)*, pages 639–642. IEEE, 2011. DOI: `10.1109/TMEE.2011.6199284`.

[8] Cheng, Ching-Hsue, Wei, Liang-Ying, and Lin, Tzu-Cheng. Improving relational database quality based on adaptive learning method for estimating

null value. In *Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)*, pages 81–81. IEEE, 2007. DOI: 10.1109/ICICIC.2007.350.

[9] Codd, Edgar F. *The relational model for database management: version 2*. Addison-Wesley Longman Publishing Co., Inc., 1990.

[10] Date, CJ. Not is not "not"! (notes on three-valued logic and related matters), chapter 8. In *Relational Database Writings*. Addison-Wesley, Reading, MA, 1989.

[11] Dhevi, AT Sree. Imputing missing values using inverse distance weighted interpolation for time series data. In *2014 Sixth International Conference on Advanced Computing (ICoAC)*, pages 255–259. IEEE, 2014. DOI: 10.1109/ICoAC.2014.7229721.

[12] Farhangfar, Alireza, Kurgan, Lukasz, and Dy, Jennifer. Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition*, 41(12):3692–3705, 2008. DOI: 10.1016/j.patcog.2008.05.019.

[13] Farhangfar, Alireza, Kurgan, Lukasz A, and Pedrycz, Witold. Experimental analysis of methods for imputation of missing values in databases. In *Intelligent Computing: Theory and Applications II*, volume 5421, pages 172–182. International Society for Optics and Photonics, 2004. DOI: 10.1117/12.542509.

[14] Farhangfar, Alireza, Kurgan, Lukasz A, and Pedrycz, Witold. A novel framework for imputation of missing values in databases. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(5):692–709, 2007. DOI: 10.1109/TSMCA.2007.902631.

[15] Grahne, Gösta. Dependency satisfaction in databases with incomplete information. In *Proceedings of the 10th International Conference on Very Large Data Bases*, pages 37–45, 1984.

[16] Grzymala-Busse, Jerzy W. and Hu, Ming. A comparison of several approaches to missing attribute values in data mining. In *International Conference on Rough Sets and Current Trends in Computing*, pages 378–385. Springer, 2000.

[17] Hartmann, Sven and Link, Sebastian. The implication problem of data dependencies over SQL table definitions: Axiomatic, algorithmic and logical characterizations. *ACM Trans. Database Syst.*, 37(2), 2012. DOI: 10.1145/2188349.2188355.

[18] Imielinski, Tomasz. Incomplete information in logical databases. *IEEE Data Engineering Bulletin*, 12(2):29–40, 1989.

[19] Imielinski, Tomasz and Lipski, Witold. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–791, 1984. DOI: 10.1145/1634.1886.

[20] Jansen, Klaus and Scheffler, Petra. Generalized coloring for tree-like graphs. *Discrete Applied Mathematics*, 75(2):135–155, 1997. DOI: 10.1016/S0166-218X(96)00085-6.

[21] Kiss, A. and Márkus, T. Functional and inclusion dependencies and their implication problems. In *Proceedings of the 10th International Seminar on Database Management Systems*, pages 31–38, 1987.

[22] Köhler, Henning, Leck, Uwe, Link, Sebastian, and Zhou, Xiaofang. Possible and certain keys for SQL. *The VLDB Journal*, 25(4):571–596, 2016. DOI: 10.1007/s00778-016-0430-9.

[23] Köhler, Henning and Link, Sebastian. SQL schema design: Foundations, normal forms, and normalization. *Information Systems*, 76:88–113, 2018. DOI: 10.1016/j.is.2018.04.001.

[24] Kratochvil, Jan and Tuza, Zsolt. Algorithmic complexity of list colorings. *Discrete Applied Mathematics*, 50(3):297–302, 1994. DOI: 10.1016/0166-218X(94)90150-3.

[25] Levene, Mark and Loizou, George. Axiomatisation of functional dependencies in incomplete relations. *Theoretical Computer Science*, 206(1):283–300, 1998. DOI: 10.1016/S0304-3975(98)80029-7.

[26] Lien, Y Edmund. On the equivalence of database models. *Journal of the ACM*, 29(2):333–362, 1982. DOI: 10.1145/322307.322311.

[27] Vassiliou, Yannis. Functional dependencies and incomplete information. In *Proceedings of the 6th VLDB Conference*. Morgan Kaufmann Publishers, 1980.

[28] Wei, Ziheng and Link, Sebastian. Embedded functional dependencies and data-completeness tailored database design. *Proceedings of the VLDB Endowment*, 12(11):1458–1470, 2019. DOI: 10.14778/3342263.3342626.

[29] Zhang, Shichao, Qin, Zhenxing, Ling, Charles X, and Sheng, Shengli. "Missing is useful": Missing values in cost-sensitive decision trees. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1689–1693, 2005. DOI: 10.1109/TKDE.2005.188.

Received 16th August 2020