Strongly Possible Keys for SQL

Munqath Alattar^{1,3} · Attila Sali^{1,2}

Received: 8 February 2019 / Revised: 14 October 2019 / Accepted: 8 August 2020 / Published online: 25 August 2020
© The Author(s) 2020

Abstract

Missing data value is an extensive problem in both research and industrial developers. Two general approaches are there to deal with the problem of missing values in databases; they could be either ignored (removed) or imputed (filled in) with new values (Farhangfar et al. in *IEEE Trans Syst Man Cybern-Part A: Syst Hum* 37(5):692–709, 2007). For some SQL tables, it is possible that some candidate key of the table is not null-free and this needs to be handled. Possible keys and certain keys to deal with this situation were introduced in Köhler et al. (*VLDB J* 25(4):571–596, 2016). In the present paper, we introduce an intermediate concept called strongly possible keys that is based on a data mining approach using only information already contained in the SQL table. A strongly possible key is a key that holds for some possible world which is obtained by replacing any occurrences of nulls with some values already appearing in the corresponding attributes. Implication among strongly possible keys is characterized, and Armstrong tables are constructed. An algorithm to verify a strongly possible key is given applying bipartite matching. Connection between matroid intersection problem and system of strongly possible keys is established. For the cases when no strongly possible keys hold, an approximation notion is introduced to calculate the closeness of any given set of attributes to be considered as a strongly possible key using the g_3 measure, and we derive its component version g_4 . Analytical comparisons are given between the two measures.

Keywords Possible and certain keys · Strongly possible keys · Database · Null values · Data imputation · Armstrong tables · Implication problem

Research was partially supported by the National Research, Development and Innovation Office (NKFIH) (Grants K-116769 and K-132696). 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: TAMOP-4.2.1/B-09/1/KMR-2010-0002).

✉ Attila Sali
sali.attila@renyi.hu
Munqath Alattar
m.attar@cs.bme.hu

¹ Department of Computer Science and Information Theory, Budapest University of Technology and Economics, Budapest, Hungary

² Alfréd Rényi Institute of Mathematics, Budapest, Hungary

³ ITRDC, University of Kufa, Kufa, Iraq

1 Introduction

Keys have always been fundamental for database management, in particular for understanding the structure and semantics of data. For a given collection of entities, a key is a set of attributes whose values enable us to uniquely identify each entity. A standard example is a relational table, where a key is a set of columns such that there are no two distinct rows that are the same restricted to the given columns. Of course, keys are significant to other data models, e.g., XML, RDF, object models and higher order data models, as well. As Köhler et al. formulate it in [20]

Knowledge about keys enables us to (i) uniquely reference entities across data repositories, (ii) minimize data redundancy at schema design time to process updates efficiently at run time, (iii) provide better selectivity estimates in cost-based query optimization, (iv) provide a query optimizer with new access paths that can lead to substantial speedups in query processing, (v) allow the database administrator (DBA) to improve the efficiency of data access via physical design techniques

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

(a) Incomplete Dataset

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

(b) Complete Dataset

Fig. 1 Complete and incomplete datasets

such as data partitioning or the creation of indexes and materialized views, and (vi) provide new insight into application data.

Many systems today allow entering incomplete tuples into a database. For example, in the case of data warehousing if different sources of raw data are merged, some attributes may exist in some of the sources while not available in some of the others. This makes it necessary to treat keys over incomplete tables. It is common to encounter databases having up to half of the entries missing, making it very difficult to mine them using data analysis methods that can work only with complete data [11].

There are different reasons why incompleteness occurs in database tables. Date [3] determined more than one kind of missing data and identified seven distinct types of null as follows: value not applicable, value unknown, value not exist, value undefined, value not valid, value not supplied, and value being the empty set. The present paper deals with data consumption with missing values in a database table, and we take the second, third, and seventh types. For the other types of missing data, we assume that symbol N/A belongs to each domain, and we treat it as regular domain element in comparisons.

Missing values issue complicates data analysis for the analysts. Other problems are usually associated with missing values such as loss of data efficiency and effectiveness [10]. Although some methods of data analysis may overcome the missing value problem, many others require complete databases. Two general approaches are there to deal with the problem of missing values in databases, incomplete tuples either could be ignored (removed) or imputed (filled in) with new values [10].

In relational databases, a key over a relation is satisfied, if no two distinct tuples have the same values on all the attributes of the key. Codd formulated the principle of a key uniqueness and totality that for a key K of any relation schema R , any relation with nulls over R , K must be null-free [7,16,17]. However, it is possible that for a table there exists no null-free set of attributes that is a key and this violates Codd's condition for the keys. So the occurrences of nulls in candidate key sets of attributes need to be handled. For example, in Fig. 1a, the candidate key (*CourseName Year*) has a null in the last tuple.

The first approach of handling the missing values in the key attributes involves ignoring any tuple that has a null in any of its values in the key. This may lead to the loss of a large amount of data and may change the original data pattern and integrity if a large number of tuples need to be ignored compared to the total number of tuples. The other approach includes an imputation operation for each occurrence of a null with a value from the attribute domain as explained by Köhler et al. [20]. We investigate the situation when the attributes' domains are not known. For that, we only consider what we have in the given data and extract the values to be imputed from the data itself for each attribute so that the resulting complete dataset after the imputation would not contain two tuples having the same value in the designated key set. Köhler et al. [20] used possible worlds by replacing each occurrence of a null with a value from the corresponding attribute's (possibly infinite) domain. Each possible world is considered as a table of total tuples. They defined a possible key as a key that is satisfied by some possible world of a nontotal database table and a certain key as a key that is satisfied by every possible world of the table. For example, Fig. 1a (a) has some possible world that satisfies the possible key {*Course Name*}, while there is no possible world of the table that satisfies the key {*Lecturer*} and, furthermore, every possible world of the table satisfies the certain key {*Course Name, Year, Semester*}.

In many cases, we have no proper reason to assume the existence of any other attribute values than the ones already existing in the table. Such examples could be types of cars, diagnoses of patients, applied medications, dates of exams, course descriptions, etc. In an SQL table, most of these attributes would have type VARCHAR(n) or DATE, so in dealing with possible worlds, one should consider any character string of length up to n , or all possible dates from "1000-01-01" to "9999-12-31." This is certainly undesired in the cases mentioned above.

We define a strongly possible key as a key that is satisfied by some possible world that is obtained by replacing each occurrence of a null by a value from the corresponding attribute existing values. We call this kind of a possible world a strongly possible world. This is a data mining-type approach, our idea is that we are given a raw table with nulls, and we would like to identify possible key sets based on the data only. As an example, {*Course Name, Year*}

is a strongly possible key of table in Fig. 1a, as the strongly possible world Fig. 1b shows. However, clearly $\{Course\ Name, Year\}$ is not a certain key, since we may replace the null in the first attribute of the third tuple by *Mathematics*. This is like if we defined each attribute type as

- $ENUM(val1, val2, val3, \dots)$ or
- $SET(val1, val2, val3, \dots)$,

where $val1, val2, val3, \dots$ are the non-NULL values that appear in the table in the given attribute's column.

There are incomplete SQL tables that do not have certain keys; for example, see Fig. 3b. In such cases, strongly possible keys make the least possible assumption about attribute domains, as they only take values that are already present, in contrast to possible keys that could take any value from a (possibly infinite) predefined domain. In this way, the results obtained by normalization, decomposition, etc. using strongly possible keys approximate the (theoretical) complete database lying behind the incomplete table. The importance of keys and the earlier investigation of the possible and certain keys motivate studying strongly possible keys and their semantic definition and properties.

We treat the implication problem for strongly possible keys and find that it behaves similarly to keys in complete datasets, which makes a difference from possible keys. Furthermore, we show that strongly possible key constraints enjoy Armstrong tables if they satisfy a natural necessary condition. We also point out a connection of matroid intersection problem and the satisfaction of a system of strongly possible keys. Furthermore, we introduce an algorithm to verify a single strongly possible key using matchings in bipartite graphs. Finally, we treat approximation measures of strongly possible keys.

The organization of this paper is as follows. In Sect. 2, the related work is reviewed, and Sect. 3 contains preliminaries and definitions. Strongly possible keys over relational data with null occurrences in the key attributes are studied in Sect. 4. In Sect. 5, the existence of a system of strongly possible keys and the use of matchings to discover strongly possible keys are studied. Algorithmic aspects of strongly possible keys are discussed in Sect. 6, and some tests results on real-world datasets are also shown. In Sect. 7, we apply an approximation measure keys to strongly possible keys and derive a new measure and compare the two measures. Results and future research directions are concluded in Sect. 8.

2 Related Work

Keys are important constraints that enforce the semantics of relational database systems. A key K satisfied by a total rela-

tion over a relation schema R if there are no two tuples in the relation that agree on all the attributes of K . Database relations that occur in real database systems usually contain occurrences of null values, and for some cases, this includes the key columns. Various studies have been done for the purpose of handling missing values.

Sree [4] shows that it is necessary to impute the missing values based on other information in the dataset to overcome the biased results that affect the accuracy of classification generated by missing values. Similarly, we use the attribute's existing values for each null in that attribute. Cheng et al. [6] utilize clustering algorithms to cluster data and calculate coefficient values between different attributes by generating the minimum average error.

Alireza et al. introduced a framework of imputation methods in [10] and evaluates how the choice of different imputation methods affects the performance in [11]. Experimental analyses of several algorithms for imputation of missing values were performed by [1, 5, 9, 18]. Our imputation method adopts the concept of graph matching by assigning for each incomplete record a complete one from the complete set of records constructed by combination of all attribute values of visible domains. An approach introduced by Zhang et al. [28] discusses and compares several strategies that utilize only known values and that “missing is useful” for cost reduction in cost-sensitive environments.

Köhler et al. [20] introduced possible and certain keys. A set K of attributes is a possible key if there is a possible world where K is a key. On the other hand, K is a certain key if it is a key in every possible world. The main concept of the present paper is between these two, since a strongly possible world is a possible world, as well. Possible worlds may use any value from an attribute domain to replace a null. This effectively allows an infinite pool of values. Strongly possible worlds are created from finite attribute domains. Some of the results in [20] essentially use that some attribute domains are infinite. In particular, the characterization of possible keys uses symbols not occurring before, to be imputed in place of nulls. This cannot be done for strongly possible keys. In the present paper, we investigate what can be stated without assuming arbitrarily large domains.

3 Preliminaries

We start with summarizing some basic definitions and terminologies. Let $R = \{A_1, A_2, \dots, A_n\}$ 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 as $D_i = dom(A_i)$ for $i = 1, 2, \dots, n$. For $X \subseteq R$, $D_X = \prod_{A_i \in X} D_i$.

An instance $T = (t_1, t_2, \dots, t_s)$ over R is a list of tuples that each tuple is a function $t : R \rightarrow \bigcup_{A_i \in R} dom(A_i)$ and $t[A_i] \in dom(A_i)$ for all A_i in R . For a tuple $t_r \in T$ and

$X \subset R$, let $t_r[X]$ be the restriction of the r th tuple of T to X . By taking list of tuples, we use the *bag semantics* that allows several occurrences of the same tuple.

In practice, database tables may have missing information about the value of some $t_j[A_i]$. Codd's null marker is included in each domain to represent the missing information [12]. Morrissett classified imperfect data into four main classes [25]. These classes are: (i) imprecise and vague, which are defined as not exact and indefinite in nature, respectively; (ii) ambiguous and subjective, which represent the uncategorizable data; (iii) unclear and uncertain, which represent the not explicitly defined data; and (iv) inconsistent and incomplete, which define the values that show contradictions or not being finished. The first two classes do not define missing values, but show undetermined and unmeasurable values, while the last two classes define the missing values and can be represented using Codd's null marker $\perp$.

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 does not contain any occurrence of $\perp$ within all the attributes in the relation, i.e., if it is R -total. Two tuples 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. [20] 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 $X \subseteq T$ 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 the same for strong similarity. For a null-free table (a table with R -total tuples), a set of attributes $K \subset R$ is a *key* if there are no two distinct tuples in the table that share the same values in all the attributes of K :

$$t_a[K] \neq t_b[K] \quad \forall 0 \leq a, b \leq s \text{ such that } a \neq b$$

Possible and certain keys were defined by Köhler et al. [20]. Let $T' = (t'_1, t'_2, \dots, t'_s)$ be a table that represents a total version of T , which is obtained by replacing the occurrences of $\perp$ in all attributes $t[A_i]$ with a value from the domain D_i , different from $\perp$ for each i . T' is called a *possible world* of T . T' is a possible world if t'_i is weakly similar to t_i and T' is completely null-free table. A *possible key* K denoted as $p\langle K \rangle$ is a key for some possible world T' of T . Similarly, a subset K of attributes is a *certain key* denoted as $c\langle K \rangle$, if it is a key for every possible world T' of T .

4 Strongly Possible Keys

A database attribute's domain is a predefined set of values that are allowed to be used for all the tuples in that attribute. For example, in Fig. 1a, the attribute *CourseName* has a predefined domain of all the computer science-related topics, but it only uses two values of Mathematics and Datamining along with $\perp$ in the last tuple.

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 the tuples in T :

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

$VD_{CourseName}$ in Fig. 1a is {Mathematics, Datamining}. The term visible domain refers to the data that already exist in a given dataset. For example, if we have a dataset with no information about the attributes' domains definitions, then we use the data themselves to define their own structure and domains. This may provide more realistic results when extracting relationships between data.

While a possible world is obtained by using the domain values instead of the occurrence of null as defined in Sect. 3, a strongly possible world is obtained by using the visible-domain values.

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

That allows us to construct a possible world of a set of data with some missed values by using only the available information. We define a *strongly possible key* as a key for some strongly possible world of T .

Definition 3 A subset $K \subseteq R$ is a *strongly possible key* (in notation $sp\langle K \rangle$) in T if there exists a strongly possible world T' such that K is a key in T' .

Recall the same instance in Fig. 1a implies

$sp\langle CourseName \ Year \rangle$ as a strongly possible key, because there is a strongly possible world in Fig. 1b where *CourseName Year* is a key. On the other hand, the table implies neither $sp\langle CourseName \ Lecturer \rangle$ nor $sp\langle Year \ Lecturer \rangle$, because there are no strongly possible worlds T' that have (*CourseName Lecturer*) or (*Year Lecturer*) as keys.

Proposition 1 Let $T = \{t_1, t_2, \dots, t_p\}$ be an instance over R . $K \subseteq R$ is a $sp\langle K \rangle \iff \exists T' \subseteq VD_1 \times VD_2 \times \dots \times VD_n$ s.t. $T' = \{t'_1, t'_2, \dots, t'_p\}$ where $t_i[K] \sim_w t'_i[K]$ and $t'_i \neq t'_j$ if $i \neq j$ and K is a key in T' .

Note that if $t_i[K] \sim_s t_j[K]$ for $i \neq j$, then K is not a strongly possible key, but the reverse is not necessarily true. For example, take the instance T of two attributes (A_1, A_2)

Fig. 2 Demonstrating no implication

	K	$R \setminus K$
t_1	0 0 0 0	00000000
t_2	$\perp \perp \perp \perp$	11111111

with the two tuples $t_1 = (1, 1)$ and $t_2 = (\perp, 2)$. Then, the only strongly possible world is $T' = \{(1, 1), (1, 2)\}$ and A_1 is not a key in T' ; nevertheless, $t_1[A_1] \sim_s t_2[A_1]$. This makes a significant difference between strongly possible keys and possible keys as it is proven by Köhler et.al.:

Theorem 1 [20] $X \subseteq R$ is a possible key for table T iff no two tuples in T are strongly similar on X .

In the relational model, any subset of attributes that are not keys are called anti keys [26]. The analogous concept can be defined in the present context.

Definition 4 We say that K is *strongly possible anti key* $\neg sp \langle K \rangle$ if $\nexists T'$ is strongly possible world such that K is a key in T' .

Definition 5 An attribute $A \in K$ is called *redundant* if $K \setminus A$ is a strongly possible key in T . And the key $sp \langle K \rangle$ is called *minimal* if $\neg sp \langle Y \rangle$ holds $\forall Y \subset K$.

4.1 Implication Problem

Integrity constraints determine the way the elements are associated with each other in a database. The implication problem specifies if a given set of constraints entails further constraints. In other words, given an arbitrary set of constraints, the implication problem is to determine whether a single constraint is satisfied by all instances satisfying given set of constraints. In our context, to define the implication, let us consider Σ as a set of strongly possible key constraints and θ as a single strongly possible key over a relation schema R . Σ logically implies θ , denoted as $\Sigma \models \theta$, if for every instance T over R satisfying every strongly possible key in Σ , we have that T satisfies θ . The next theorem describes the implication problem for the strongly possible keys.

Theorem 2 $\Sigma \models sp \langle K \rangle \iff \exists Y \subseteq K$ s.t. $sp \langle Y \rangle \in \Sigma$.

Proof $\Leftarrow$: $\exists T'$ s.t. $t'_i[Y] \neq t'_j[Y], \forall i \neq j$, so $t'_i[K] \neq t'_j[K], \forall i \neq j$ holds, as well.

$\Rightarrow$: Suppose indirectly that $sp \langle Y \rangle \notin \Sigma \forall Y \subseteq K$. Consider the following instance consisting of two tuples $t_1 = (0, 0, \dots, 0), t_2[K] = (\perp, \perp, \dots, \perp)$, and $t_2[R \setminus K] = (1, 1, \dots, 1)$ as in Table 2. Then, the only possible t'_2 in T' is $t'_2(0, 0, \dots, 0, 1, 1, \dots, 1)$. Furthermore, for each Z , where $sp \langle Z \rangle \in \Sigma$, there must exist $z \in Z \setminus K$; thus, $t'_1[Z] \neq t'_2[Z]$ but $t'_1[K] = t'_2[K]$ showing that (t_1, t_2) satisfies every strongly possible key constraint from Σ , but does not satisfy $sp \langle K \rangle$.

Note 1 If $\Sigma \models \neg sp \langle K \rangle$ and $Y \subseteq K$, then $\Sigma \models \neg sp \langle Y \rangle$.

Note 2 If $\Sigma \models sp \langle K \rangle$, then $\Sigma \models p \langle K \rangle$, but the reverse is not necessarily true, since $D_K \supseteq VD_K$ could be proper containment so K could be made a key by imputing values from $D_K \setminus VD_K$. For example, in Fig. 2, it is shown that $\neg sp \langle K \rangle$ holds, but $p \langle K \rangle$ may hold in some T' if there is at least one other value in the domain of K than zero that can be placed instead of the nulls in the second tuple so that $t'_1[K] \neq t'_2[K]$ results.

Note 3 If $\Sigma \models c \langle K \rangle$, then $\Sigma \models sp \langle K \rangle$. As certain keys hold in any possible world, they hold also if this possible world is created using visible domain.

Note 4 For a single attribute A , $sp \langle A \rangle \iff t[A] \sim_s t'[A] \forall t, t'$ s.t. $t \neq t'$, i.e., if there are no nulls in A .

In other words, single attribute with a null value cannot be a strongly possible key. That is because replacing an occurrence of null with a visible-domain value results in duplicated values for that attribute.

4.2 Armstrong Tables

Armstrong tables are useful tools to represent constraint sets in a user-friendly way [2,8,15,24]. For a class $\mathcal{C}$ of constraints and a set Σ of constraints in $\mathcal{C}$, aC-Armstrong table T for Σ satisfies Σ and violates all the constraints in $\mathcal{C}$ not implied by Σ . Therefore, given an Armstrong table T for Σ , the problem of deciding for an arbitrary constraint σ in $\mathcal{C}$ whether Σ implies σ reduces to the problem of verifying whether σ holds on T . The ability to compute an Armstrong table for Σ provides us with a data sample that is a perfect semantic summary of Σ . For further details how Armstrong tables help the communication between database engineers and domain experts, the reader is referred to Sect. 4.2 of [20].

Following [20], we introduce the concept of null-free subschema (NFS). Let R be a schema; an NFS R_S over R is a set such that $R_S \subseteq R$. An instance T satisfies NFS R_S if it is R_S -total; that is, each tuple $t \in T$ is R_S -total. This corresponds to NOT NULL constraint of SQL.

Definition 6 Let Σ be a collection of strongly possible key constraints. An instance T over (R, R_S) is an *Armstrong table* for (R, R_S, Σ) if for every strongly possible key θ over R , θ holds in T iff $\Sigma \models \theta$, and for every attribute $A \in R \setminus R_S$ there exists a tuple $t \in T$ with $t[A] = \perp$.

Let us suppose that $\Sigma = \{sp \langle K \rangle : K \in \mathcal{K}\}$ is given. By Note 4 if $|K| = 1$, then $K \subseteq R_S$ must hold. If this restriction is satisfied, then Σ enjoys an Armstrong table. We would like to point out a significant difference between strongly possible keys and possible keys. In [20], an example of a collection of possible and certain key constraints is given that does not have an Armstrong table, while the next theorem states this does not happen for strongly possible keys.

Theorem 3 Suppose that $\Sigma = \{sp \langle K \rangle : K \in \mathcal{K}\}$ is a collection of strongly possible key constraints such that if $|K| = 1$, then $K \subseteq R_S$. Then, there exists an Armstrong table for (R, R_S, Σ) .

Proof Let $\mathcal{A}$ be the collection of strongly possible anti keys; that is, $\mathcal{A} = \{A \subseteq R : \Sigma \not\models sp \langle A \rangle\}$. According to Theorem 2 and Note 1, $\mathcal{A}$ is a downset and $\Sigma \models sp \langle K \rangle \iff K \setminus A \neq \emptyset$ for all $A \in \mathcal{A}$. Let $H = \{K_1, K_2, \dots, K_u\}$ be the set of singleton attribute keys, note that $K_i \notin \mathcal{A}$ for all $i = 1, 2, \dots, u$ and $A \in \mathcal{A}$. Let the maximal (under containment) elements of $\mathcal{A}$ be $\{A_1, A_2, \dots, A_p\}$, and assume that $R = \{K_1, K_2, \dots, K_u, X_1, X_2, \dots, X_n\}$ with $R \setminus R_S = \{X_1, X_2, \dots, X_m\}$. Construct table $T = \{t_0, t_1, \dots, t_{p(m+1)}\}$ as follows. $t_0[X] = 0 \forall X \in R$. For $i = 1, 2, \dots, p$ let $t_i[X] = \perp \forall X \in R \setminus R_S$, $t_i[X] = 0$ for $X \in R_S \cap A_i$ and $t_i[X] = i$ for $X \in R_S \setminus A_i$. Note that $H \subseteq R_S \setminus A_i$ for all i . Let $t_{ip+j}[X_j] = i$ and $t_{ip+j}[X_\ell] = ip+j$ for $i = 1, 2, \dots, p$ $j = 1, 2, \dots, m$ and $\ell \neq j$. Furthermore, let $t_{ip+j}[K_g] = ip+j$ for $g = 1, 2, \dots, u$. Observe that t_z is R -total for $z > p$ and that $t_z[X] \neq t_u[X]$ for $u, z > p$ and for all $X \in R$.

Create a strongly possible world T' from T by replacing the null of $t_i[X]$ by 0 if $X \in A_i$ and by i otherwise. We claim that no two tuples of T' agree on all attributes of K if $\Sigma \models sp \langle K \rangle$. Indeed, this latter property happens iff $K \setminus A_i \neq \emptyset$ for all $i = 1, 2, \dots, p$; hence, for all $0 \leq j < i \leq p$, there exists an attribute $X \in K \setminus A_i$ such that $t_j[X] \neq i$ but $t_i[X] = i$. Furthermore, if $0 \leq i \leq p$ and $j > p$, then t_i and t_j can agree in at most one attribute, but that attribute is not a singleton attribute key. On the other hand, if $\Sigma \not\models sp \langle L \rangle$, then there exists i such that $L \subseteq A_i$, which implies that $t_0[L] = t_i[L]$; that is, L is not a key in table T' .

5 Matchings, Matroids, and Strongly Possible Keys

In this section, we study the verification problem of strongly possible keys. That is given table instance $T = \{t_1, t_2, \dots, t_s\}$ over schema $R = \{A_1, A_2, \dots, A_n\}$ and a collection $\mathcal{K} = \{K_1, K_2, \dots, K_p\}$, attribute sets determine whether $\Sigma = \{sp \langle K_1 \rangle, sp \langle K_2 \rangle, \dots, sp \langle K_p \rangle\}$ holds in T . As we will see, there is a significant difference between the cases of single strongly possible key and system of multiple strongly possible keys.

5.1 Checking a Single Strongly Possible Key

If we want to decide whether $sp \langle K \rangle$ holds or not, we can forget about the attributes that are not in K , since we need distinct values on K as a matching from $VD_{A_1} \times VD_{A_2} \times \dots \times VD_{A_b}$ to $T = \{t_1, t_2, \dots, t_r\}|_K$ where $K = \{A_1, A_2, \dots, A_b\}$.

Thus, we may construct a table T' that is formed by finding all the possible combinations of the visible domains of $T|_K$ that are weakly similar to some tuple in $T|_K$.

$$T' = \{t' : \exists t \in T : t'[K] \sim_w t[K]\} \subseteq VD_1 \times VD_2 \times \dots \times VD_b$$

Let bipartite graph $G = (T, T'; E)$ be defined by $\{t, t'\} \in E \iff t[K] \sim_w t'[K]$. Finding a matching between T and T' that covers all the tuples in T (if exist) will result in the set of tuples in T' that can be used to replace incomplete tuples in T to obtain a strongly possible world for T , which verifies that K is a strongly possible key. The algorithmic details are discussed in Sect. 6.

Example 1 Figure 3b shows an incomplete set of tuples where $K = \{Lecturer, Course\}$. A visible domain can be identified for each attribute to construct tuples of T' by finding the combinations of all the visible-domain values as shown in Fig. 3c (where we included all tuples for tables a and b together). Bipartite graphs between tuples with null(s) in T and tuples in T' excluding those tuples that agree on K to any total tuple in T are constructed. Figure 4b illustrates the graph for Fig. 3b which contains a complete matching to assign a total tuple to each nontotal tuple in T and K is a key, while Figs. 3a, 4a show there is no matching that covers all the tuples in T .

5.2 System of Multiple Strongly Possible Keys

Here, we show how the existence of a system of strongly possible keys is equivalent to the existence of a given sized common independent set of several matroids. This is a natural extension of the bipartite matching idea of the previous subsection, since transversal matroids are generalizations of bipartite graphs. For basic definitions and properties of matroids, the reader is referred to Welsh's book [27]. Let us be given schema $R = \{A_1, A_2, \dots, A_n\}$, and let $\mathcal{K} = \{K_1, K_2, \dots, K_p\}$ be a collection of attribute sets and $T = \{t_1, t_2, \dots, t_s\}$ be an instance with possible null occurrences. Our main question here is whether $\Sigma = \{sp \langle K_1 \rangle, sp \langle K_2 \rangle, \dots, sp \langle K_p \rangle\}$ holds in T ? Let $E_i = \{t' \in VD_1 \times VD_2 \times \dots \times VD_n : t' \sim_w t_i\}$. A strongly possible world that satisfies Σ is given by an injective mapping

$$f : T \rightarrow VD_1 \times VD_2 \times \dots \times VD_n \text{ such that } f(t_i) \in E_i \forall i$$

and for each j , K_j is a key in $T' = f(T)$. Let $S \subseteq VD_1 \times VD_2 \times \dots \times VD_n$ be the union $S = E_1 \cup E_2 \cup \dots \cup E_s$ and define bipartite graph $G = (T, S; E)$ by $\{t, t'\} \in E \iff t \sim_w t'$ for $t \in T$ and $t' \in S$. Let $(S, \mathcal{M}_0)$ be the transversal matroid defined by G on S ; that is, a subset $X \subseteq S$ satisfies

	Lecturer	Course
t_1	Sarah	Mathematics
t_2	Sarah	$\perp$
t_3	$\perp$	Data Mining
t_4	James	Data Mining
t_5	David	Data Mining

(a)

	Lecturer	Course
t_1	Sarah	Mathematics
t_2	Sarah	$\perp$
t_3	$\perp$	Mathematics
t_4	James	Data Mining
t_5	David	Data Mining

(b)

	Lecturer	Course
t'_1	Sarah	Mathematics
t'_2	Sarah	Data Mining
t'_3	James	Mathematics
t'_4	James	Data Mining
t'_5	David	Mathematics
t'_6	David	Data Mining

(c)

	A_1	A_2
t_1	0	0
t_2	0	$\perp$
t_3	$\perp$	1
t_4	1	1
t_5	2	0
t_6	2	1
t_7	3	0
t_8	3	1
	$\vdots$	$\vdots$
t_{2n+1}	n	0
t_{2n+2}	n	1

(d)

Fig. 3 Data samples

Fig. 4 Data graphs

$X \in \mathcal{M}_0$ if X can be matched into T . Furthermore, consider the partitions

$$S = S_1^j \cup S_2^j \cup \dots \cup S_{p_j}^j \quad (1)$$

induced by K_j for $j = 1, 2, \dots, p$ such that S_i^j 's are maximal sets of tuples from S that agree on K_j . Let $(S, \mathcal{M}_j)$ be the partition matroid given by (1). We can formulate the following theorem.

Theorem 4 Let T be an instance over the schema $R = \{A_1, A_2, \dots, A_n\}$, and let $\mathcal{K} = \{K_1, K_2, \dots, K_p\}$ be a collection of attribute sets. $\Sigma = \{sp\langle K_1 \rangle, sp\langle K_2 \rangle, \dots, sp\langle K_p \rangle\}$ holds in T if and only if the matroids $(S, \mathcal{M}_j)$ have a common independent set of size $|T|$ for $j = 0, 1, \dots, p$.

Proof An independent set T' of size $|T|$ in matroid $(S, \mathcal{M}_0)$ means that the tuples in T' form a strongly possible world for T . That they are independent in $(S, \mathcal{M}_j)$ means that K_j is a key in T' ; that is, $sp\langle K_j \rangle$ holds.

Conversely, if $\Sigma = \{sp \langle K_1 \rangle, sp \langle K_2 \rangle, \dots, sp \langle K_p \rangle\}$ holds in T , then there exists a strongly possible world $T' = \{t'_1, t'_2, \dots, t'_s\} \subseteq VD_1 \times VD_2 \times \dots \times VD_n$ such that $t_i \sim_w t'_i$. This means that $T' \subseteq S$ and that T' is independent in transversal matroid $(S, \mathcal{M}_0)$. That $sp \langle K_j \rangle$ holds implies that tuples t'_i are pairwise distinct on K_j ; that is, T' is independent in partition matroid $(S, \mathcal{M}_j)$.

Unfortunately, Theorem 4 does not give a good algorithm to decide the satisfaction of a system Σ of strongly possible keys, because as soon as Σ contains at least two constraints, then we would have to calculate the size of the largest common independent set of at least three matroids, known to be an NP-complete problem in general [13].

In the case of a single strongly possible key $sp \langle K \rangle$ constraint, Theorem 4 requires computing the largest common independent set of two matroids, which can be solved in polynomial time [21]. However, we can solve that case by reducing to the somewhat simpler problem of matchings in bipartite graphs.

To conclude this section, we give some necessary conditions for strongly possible key constraints to be satisfied.

5.3 Necessary Conditions

Let $c_v(A)$ denote the number of tuples that have value v in attribute A ; that is, $c_v(A) = |\{t \in T : t[A] = v\}|$. The next are some necessary conditions to $T \models sp \langle K \rangle$ to hold.

Proposition 2 *Let $K \subseteq R$ be a set of attributes. If $sp \langle K \rangle$ holds, then*

1. No two tuples t_i, t_j are strongly similar in K .
2. $|T| \leq \prod_{A \in K} |VD_A|$.
3. $\forall B \in K$, the number of nulls in B is at most $\sum_{v \in VD_B} \left(\frac{\prod_{A \in K} |VD_A|}{|VD_B|} - c_v(B) \right)$.
4. For all $v \in VD_B$ we have $c_v(B) \leq \frac{\prod_{A \in K} |VD_A|}{|VD_B|}$.

Proof (1) follows from the fact that any strongly possible key is a possible key, as well, and from the characterization of possible keys given in [20] that $p \langle K \rangle$ holds $\iff t_i[K] \sim_s t_j[K]$ for any $i \neq j$. In addition to that, for any set of attributes, the maximum number of distinct combinations of their values is the size of the multiplication of their visible domain, and this proves (2). Moreover, to prove conditions (3) and (4), when $sp \langle K \rangle$ is satisfied in T , then there should exist a T' with no two tuples having the same values in all attributes of K after filling all their nulls. So for each set of tuples S that has the same value v in the attribute B , the number of distinct combinations of the other attributes is the multiplication of their VD s, which means the number of tuples in S should not be more than $\prod_{A \in (K \setminus B)} |VD_A|$. Thus,

the number of times that the value v can be used to replace a null in attribute B is at most $\frac{\prod_{A \in K} |VD_A|}{|VD_B|} - c_v(B)$.

Note that $sp \langle K \rangle$ holds if a matching covering T exists in the bipartite graph $G = (T, T'; E)$ defined as above, $\{t, t'\} \in E \iff t[K] \sim_w t'[K]$. We can apply Hall's theorem to obtain $\forall X \subseteq T$, and we have $|N(X)| \geq |X|$ for

$$N(X) = \{t' : \exists t \in X \text{ such that } t' \sim_w t[K]\}$$

The conditions in Proposition 2 are implied by Hall's condition, as well. Let us assume that Hall's condition is true for a set of tuples T and attribute set K . If t_i, t_j are strongly similar on K , then the set $X = \{t_i, t_j\}$ has $1 = |N(X)| < |X| = 2$ that proves (1). For condition (2), the graph here is $G = (T, T'; E)$ such that $T' \leq \prod_{A \in K} |VD_A|$ and T' is the collection of neighboring tuples to the tuples in T so that it is always true that $|T| \leq |T'|$. Condition (3) is implied as follows. Let X be the set of tuples that have nulls in B . Then, the number of tuples $t' \in N(X)$ such that $t'[B] = v$ is at most $\frac{\prod_{A \in K} |VD_A|}{|VD_B|}$. Finally, for an attribute $B \in K$, let X_v be the set of all tuples that have value v in the attribute B . Then, $|N(X_v)| \leq \frac{|VD_1 \times VD_2 \times \dots \times VD_B|}{|VD_B|}$ proving (4).

As defined in Sect. 3, certain key is a key for any possible world, i.e., all the tuples are distinct after filling the nulls regardless of what values are used. We prove that one does not need to check all possible worlds in order to verify that some certain key constraint is satisfied, and it is enough to consider strongly possible worlds only.

Theorem 5 *Let T be a table instance over schema R such that a strongly possible world of T exists. $K \subseteq R$ is a certain key if and only if K is a key in any strongly possible world of T .*

Proof $\Rightarrow$: If $c \langle K \rangle$ holds, then K is a key in any possible world by definition, so in particular in any strongly possible world, as well.

$\Leftarrow$: Let us assume that K is a key in any strongly possible world, but there exists a possible world T' , and two distinct tuples $t'_1 \neq t'_2$ of T' such that $t'_1[K] = t'_2[K]$. Let $A \in K$ be an attribute. There are three possibilities. If neither t_1 nor t_2 has $\perp$ in A , then let $t_1[A] = t_2[A] = t''_1[A] = t''_2[A]$. If one of them has $\perp$ in A , say $t_1[A] = \perp$, then we can set $t''_1[A] = t''_2[A] = t_2[A] \neq \perp$ such that $t_2[A] \in VD_A$. Finally, if $t_1[A] = t_2[A] = \perp$, then pick any $x \in VD_A$ and set $t''_1[A] = t''_2[A] = x$. Such an x exists, since T has a strongly possible world. For attributes not in K , extend t''_1 and t''_2 arbitrarily from the visible domains of the attributes. Also, fill up the nulls of other tuples of T from the visible domains to obtain a strongly possible world, where distinct tuples t''_1 and t''_2 agree on K , contradicting to the assumption that K is a key in any strongly possible world.

Note that a table instance T over schema R fails to have a strongly possible world only if there exists an attribute $A \in R$

Fig. 5 A certain key graph

that for all $t \in T$ $t[A] = \perp$ holds, which is a rather degenerate case.

Thus, certain keys can also be recognized from the bipartite graph $= (T, T'; E)$ defined above, since all the tuples are distinct after filling the nulls regardless of what visible-domain values are used. Then, every incomplete tuple in T has a distinct set of weakly similar tuples in T' and any one of these tuples can be assigned to that incomplete tuple. The bipartite graph of a certain key would contain a connected component for each tuple in T with a set of tuples in T' as illustrated in Fig. 5.

The concept of strongly possible keys lies in between the two concepts of possible and certain keys. Every certain key is a strongly possible key, and every strongly possible key is a possible key. Figure 6 shows that

$$\{K : c\langle K \rangle \text{ holds}\} \subseteq \{K : sp\langle K \rangle \text{ holds}\} \subseteq \{K : p\langle K \rangle \text{ holds}\}.$$

For example, Fig. 1a satisfies the certain key $c\langle \text{CourseName}, \text{Year}, \text{Semester} \rangle$. Also, there are some strongly possible worlds that satisfy strongly possible keys such as $sp\langle \text{CourseName}, \text{Year} \rangle$,

$sp\langle \text{CourseName}, \text{Credits} \rangle$, and $sp\langle \text{Year}, \text{Credits} \rangle$, but they violate the certain keys for these attribute sets. Furthermore, $p\langle \text{Year}, \text{Lecturer} \rangle$ is satisfied but neither $c\langle \text{Year}, \text{Lecturer} \rangle$ nor $sp\langle \text{Year}, \text{Lecturer} \rangle$ holds.

6 Strongly Possible Key Discovery

$sp\langle K \rangle$ holds if and only if there exists a matching between T and T' that covers all tuples of T as explained earlier. We

Fig. 6 Possible, strongly possible, and certain key scopes

introduce Algorithm 1 to find a strongly possible world (if it exists) for a given T which verifies that $sp\langle K \rangle$ holds. We start by generating T' from T . T' contains all total tuples that are weakly similar to some incomplete tuples in T . Let us define the usable visible domain of an attribute B as

$$UVDB = \left\{ v \in VD_A : \frac{\prod_{A \in K} |VD_A|}{VD_B} - c_v(B) > 0 \right\}.$$

We use the usable visible-domain values for each null in the attributes of K to reduce complexity based on Proposition 2 4.

Generating T' is done by taking the nontotal tuples of T one by one and finding the weakly similar total tuples using the $UVDB$ s for each attribute instead of the nulls. This process may result in some duplicates in T' because it is possible that more than one nontotal tuple in T can be weakly similar to same total tuple. For example, in Fig. 1a, the first and the last tuples are both nontotal and are both weakly similar to the total tuple (Mathematics, 2019, Sarah, 5) generated by using $UVDB$ for each null. For that, in the algorithm, we need to remove these duplicated tuples in T' after generating

the weakly similar tuples for all the tuples $t \in T$. Removing duplicates can be done by sorting the tuples using Radix sort.

After calculating T' , we find the maximum matching between $T \setminus T_{\text{total}}$ and $T' \setminus T_{\text{total}}$, where T_{total} is the set of all the total tuples in T . Every total tuple in T is matched to the corresponding total tuple in T' , so that we find the maximum matching between $T' \setminus T_{\text{total}}$ and $T \setminus T_{\text{total}}$ to reduce the complexity. Function BipartiteMatching uses the standard augmenting path method to find a largest matching [23]. Let us recall that an augmenting path with respect to a matching M is a path from an unmatched element in T to an unmatched element in T' that uses nonmatching and matching edges alternately. We define $M \oplus P = (M \setminus P) \cup (P \setminus M)$ for an augmenting path P with respect to a matching M . $M \oplus P$ is again a matching, and since P started and ended with non-matching edges, $M \oplus P$ is of size one larger than M . If the resulting largest matching covers all the tuples in $T \setminus T_{\text{total}}$, then T has a strongly possible world which is $T' \cup T_{\text{total}}$ that verifies $sp \langle K \rangle$, otherwise $\neg sp \langle K \rangle$ holds.

The running time depends on the size of T' , which could be exponential in the size of the input. Sorting using Radix Sort takes $O(|R|(|T| + |T'|))$ time, while finding the largest matching in $G = (T \setminus T_{\text{total}}, T' \setminus T_{\text{total}}; E)$ takes $O((|T \setminus T_{\text{total}}| + |T' \setminus T_{\text{total}}|)|E|)$ time by the augmenting path method.

Algorithm 1 Discovering a Strongly Possible Keys

Input: Dataset T on relation schema R with a key K of b attributes

Output: A strongly possible world if one exists

```

1: procedure spKeyDiscovery(item  $T$ , item  $K$ )
2:    $T_{\text{total}} = \{t : t[i] \neq \perp \forall i = (1, 2, \dots, b)\}$ 
3:    $T' = \emptyset$ 
4:   for all  $t \in T \setminus T_{\text{total}}$  do
5:      $T'_{\text{temp}} = \{t'_{\text{temp}} : t'_{\text{temp}} \sim_w t \text{ Using } \begin{cases} UVD_A, & \text{if } t[A] = \perp \\ t[A], & \text{if } t[A] \neq \perp \end{cases}\}$ 
6:      $T' = T' \cup T'_{\text{temp}}$ 
7:      $T'_{\text{temp}} = \emptyset$ 
8:   end for
9:   Sort  $T'$  using RadixSort
10:  Remove the duplicated tuples in  $T'$ 
11:  Graph  $G = \{V = (T \setminus T_{\text{total}}, T' \setminus T_{\text{total}}); E\}$ 
12:  BipartiteMatching( $G$ )
13:  if  $\text{size}(\text{Match}) = |T \setminus T_{\text{total}}|$  then
14:     $F = \text{Match} \cap T$ 
15:     $F = F \cup T_{\text{total}}$ 
16:    return  $F$ 
17:  else
18:    return "No strongly Possible World Exists for  $T'$ "
19:  end if
20: end procedure
21: procedure BipartiteMatching(item  $G$ )
22:   $\text{Match} = \emptyset$ 
23:  repeat
24:     $P = (\text{AugmentingPath}(G))$ 
25:     $\text{Match} = \text{Match} \triangle P$ 
26:  until  $P = \emptyset$ 
27:  return  $\text{Match}$ 
28: end procedure
29: procedure AugmentingPath(item  $G, M$ )
30:  Direct edges not in  $M$  from  $T$  to  $T'$ , edges in  $M$  from  $T'$  to  $T$ 
31:  Add  $s, t$  and connect them to nonmatched vertices in  $T$  and  $T'$ ,
    respectively
32:  Run BFS on  $G$  from source  $s$ .
33:  if  $t$  is reached then
34:    return  $P \setminus \{s, t\}$  for a shortest path  $P$  from  $s$  to  $t$ 
35:  else
36:    return  $\emptyset$ 
37:  end if
38: end procedure

```

6.1 Application to Real-Life Datasets

The experiments work was done on an Asus X541UJ, Intel core i7, CPU 2.7 GHz, 8GB RAM, and 64-bit operating system. The datasets were stored and processed in MySQL version 5.7.21.

We used the following two publicly available datasets, and we removed some attribute values and replaced them by nulls randomly in 10%, and 50% of its rows simulate incomplete tables.

1. <http://archive.ics.uci.edu/ml/datasets/QSAR+fish+toxicity>
2. <https://www.kaggle.com/mirichoi0218/insurance>

We looked for the smallest possible strongly possible key sets, that is, two-element sets. We found that the tables satisfy

no certain key of two elements after introducing the nulls, while we could find some attributes combinations that form strongly possible keys. For the other pairs, we found that they are not strongly possible keys because either there is no matching that covers all the rows or of the existence of some duplicate rows.

QSAR fish toxicity Dataset This dataset was used to develop quantitative regression QSAR models to predict acute aquatic toxicity toward the fish *Pimephales promelas* (fathead minnow) on a set of 908 chemicals.

Table name is `qsar_fish_toxicity` which has 908 rows and seven attributes, six molecular descriptors and one quantitative experimental response, where $|VD_{CIC0}| = 502$, $|VD_{SM1Dz(Z)}| = 186$, $|VD_{GATS1i}| = 557$, $|VD_{NdsCH}| = 5$, $|VD_{NdsC}| = 7$, $|VD_{MLOGP}| = 559$, and $|VD_{quantitative_response}| = 827$.

The table satisfies $sp\langle MLOGP_quantitative_response \rangle$ but not $c\langle MLOGP_quantitative_response \rangle$ for 10% null occurrence.

$sp\langle SM1Dz(Z) NdsCH \rangle$ and $sp\langle NdsCH NdsC \rangle$ are not satisfied because their visible-domains multiplication is less than the total number of rows in the table. Furthermore, all other two-column combinations do not form strongly possible keys because either there is no matching that covers all the tuples or of the existence of some duplicate rows.

Similar to the 10% null occurrence, the table satisfies $sp\langle MLOGP_quantitative_response \rangle$ but not $c\langle MLOGP_quantitative_response \rangle$ for 50% null occurrence.

Medical Cost Personal Datasets This dataset contains some health services history to accurately predict insurance costs. It has 1338 rows and seven columns ($|VD_{age}| = 47$, $|VD_{sex}| = 2$, $|VD_{bmi}| = 548$, $|VD_{children}| = 6$, $|VD_{smoker}| = 2$, $|VD_{region}| = 4$, and $|VD_{charges}| = 1337$).

The table satisfies $sp\langle bmi\ age \rangle$ and $sp\langle bmi\ charges \rangle$ with no certain key of size 2, after randomly adding nulls in 10% and 50%, respectively, of the rows.

7 Strongly Possible Keys Approximation

The main motivation for the study of strongly possible keys lies in the unique identification of tuples of an incomplete dataset by filling up occurrences of nulls using the already-present data only, if it is possible. In some cases, there may not exist any strongly possible keys, so how well a table T approximates $sp\langle K \rangle$ is a natural question. Let T be the instance in Fig. 3a, then there is no possibility of having $sp\langle K \rangle$ because of the violation of Hall's condition as in Fig. 4. In fact, $sp\langle K \rangle$ holds restricted to the last four tuples; that is, if we remove the first tuple, $sp\langle K \rangle$ is satisfied. This

is what we call the *Approximate Strongly Possible Key* (ASP Key) property.

Definition 7 Attribute set K is approximate strongly possible key of ratio a in table T , in notation $asp_a\langle K \rangle$, if there exists a subset S of the tuples T such that $T \setminus S$ satisfies $sp\langle K \rangle$, and $|S|/|T| \leq a$. The variable a represents the approximation which is the ratio of the number of tuples needed to be removed over the total number of tuples so that $sp\langle K \rangle$ holds.

Variable a has a value ranging from 0 to 1, it is equal to 0 exactly when $sp\langle K \rangle$ holds in T , i.e. no tuples are required to be removed. So for our example table, the strongly possible key approximation is $asp_{0.2}\langle K \rangle$ since maximum of four tuples out of five may have a unique value in the key set in any strongly possible world.

Let us consider now Fig. 3d that has $2n + 2$ tuples, and like in Fig. 3a, there are two tuples with the same values after filling the nulls using visible-domain elements. But the ratio of having an $sp_a\langle K \rangle$ for Fig. 3d is smaller, since there are $2n + 1$ tuples with distinct values after filling the nulls. Hence, table d has a smaller degree to have a strongly possible key. To measure this degree in a given dataset, we use the g_3 measure introduced in [19]. g_3 based on the idea that the degree to which ASP key is approximated is determined by the minimum number of tuples needed to be removed from T so that K becomes an strongly possible key.

$$g_3(K) = \min\{|S|/|T| : (T \setminus S) \models sp\langle K \rangle\}$$

Finding a minimum set of tuples that we need to remove can be done by constructing the maximum matching in graph $G = (T, T'; E)$, namely the unmatched tuples need to be removed.

Proposition 3 If $g_3(K) = 0$, then the conditions in Proposition 2 are true.

So by using the g_3 measure, the approximation measure of the strongly possible key is:

$$g_3(K) = \frac{|T| - v(G)}{|T|}$$

where $v(G)$ denotes the maximum size of a matching in graph G . Note that the smaller $g_3(K)$ is, the closer is K being a strongly possible key.

Let $\mathcal{M}$ be the collection of connected components in graph G that satisfy the strongly possible key condition; i.e. there is a matching covering all tuples in that set ($\forall M \in \mathcal{M} \nexists X \subseteq M \cap T$ such that $|X| > N(X)$). Let $C \subseteq G$ be defined as $C = G \setminus \bigcup_{M \in \mathcal{M}} M$, and let $\mathcal{M}'$ be the set of connected components of C . Let V_M denote the set of vertices of T in a component M . So, the maximum matching can be written as $\sum_{M \in \mathcal{M}} (|V_M|) + \sum_{M' \in \mathcal{M}'} v(M')$. Therefore, we can

Fig. 7 Sample tables for comparison results

Fig. 8 Visible-domain combinations of tables of Fig. 7

(A)	$t'_i = (i-1, 0) \ i = 1, 2, \dots, n+1$
(B)	$t'_i = (i-1, 0) \ i = 1, 2, \dots, n+1$
(C)	$t'_i = (i-1, 0) \ i = 1, 2, \dots, n+1$
(D)	$t'_i = (i, 0, 0) \text{ for } i = 1, 2, \dots, n-\beta, \text{ and } t'_{n-\beta+j} = (0, 0, j-1) \text{ for } j = 1, 2, \dots, l+1$
(E)	$t'_i = (0, 0)$
(F)	$t'_i = (i, 0) \ i = 1, 2, \dots, n-1$
(G)	$t'_\ell = (\ell, 0) \text{ for } \ell = 1, 2, \dots, k(n-1)$

rewrite measure g_3 as:

$$g_3(K) = \frac{|T| - (\sum_{M \in \mathcal{M}} |V_M|) + \sum_{M' \in \mathcal{M}'} v(M')}{|T|}$$

Measurement of strongly possible keys approximation can be more appropriate by taking into consideration the effect of each connected component in the graph on the matching. More specifically, $\mathcal{M}$ represents the sets of tuples that do not require any tuple to be removed to get a strongly possible key, while the components of $\mathcal{M}'$ represent the sets of tuples that contain some tuples needed to be removed to have a strongly possible key. We consider the components of $\mathcal{M}$ to get their effect doubled in the approximation measure because they represent a part of the data that is not affected by any tuples removal. So we propose a derived version of g_3 measure named g_4 that consider the effects of these components:

$$g_4(K) = \frac{|T| - (\sum_{M \in \mathcal{M}} |V_M|) + \sum_{M' \in \mathcal{M}'} v(M')}{|T| + \sum_{M \in \mathcal{M}} |V_M|}$$

7.1 Analytical Comparison

We compare approximation measures g_3 and g_4 similarly to the investigations in [14]. Figure 7 shows seven tables that represent the key part only of the data where each table has more than one attribute. Tables A, B, and C have $2n$ tuples, tables E and F have n tuples, and table D has $n+l$ tuples, while table G has kn tuples. Table D includes a variable $0 \leq \beta \leq \frac{n}{2}$. We intend to use these cases to illustrate the differences between the two measures and give a bound of g_3/g_4 where it is always true that $g_3 - g_4 \geq 0$. The graphs show the weak similarity relationship between the data tuples and the visible domain combinations. The visible-domain combinations are shown in Fig. 8. For example, in table A, the first two tuples of T in the left side of the graph can have a unique weakly similar tuple in T' for each, while for the rest, every two tuples in T form a connected component that have only one weakly similar tuple in T' . On the other hand, all the tuples of table E form a connected component of size n that are weakly similar to a single tuple in T' .

We intend to distinguish between having an *sp* key that holds without any tuples removal and that which requires removing at least one tuple. A connected component that has a maximum matching that covers all the tuples in T of that component is stronger than the ones that require removing at

Table 1 Main Comparison Results

	A	B	C	D	E	F	G
g_3	$\frac{n-1}{2n}$	$\frac{n-1}{2n}$	$\frac{1}{4}$	$\frac{\beta-1}{n+l}$	$\frac{n-1}{n}$	$\frac{1}{n}$	$\frac{1}{n}$
g_4	$\frac{n-1}{2n+2}$	$\frac{n-1}{3n}$	$\frac{1}{6}$	$\frac{\beta-1}{n+2l}$	$\frac{n-1}{n}$	$\frac{1}{2n-2}$	$\frac{1}{2n-2}$

least one tuple. For example, in Fig. 7, tables A and B show the case that requires removing almost half of the tuples to get the strongly possible key hold. But in table B, half of the data need no changes, while table A has only two tuples out of $2n$ tuples which can be left as they are. Table C has a quarter of the total number of the tuples needed to be removed to get the maximum complete matching for the remaining tuples. l tuples in table D have complete matching in the graph, and the rest needs $\beta-1$ tuples to get removed. For table E, one tuple should be kept after removing the rest, and table F requires one tuple of the first two tuples to be removed. And there are kn tuples in table G such that for each of n tuples, there is only one that needs to be removed, meaning there are k tuples to be removed of the total.

Table 1 shows the measures of the seven tables using g_3 and g_4 . The results show some differences in g_4 measures between two tables although they may have the same g_3 value. The g_3 measure of table A, $g_3(A)$, is $\frac{n-1}{2n}$ and similarly $g_4(A) = \frac{n-1}{2n+2}$, while $g_3(B) = \frac{n-1}{2n}$ and $g_4(B) = \frac{n-1}{3n}$. Tables A and B got a lower approximation using g_4 than using g_3 , while table E has an equal measure for both g_3 and g_4 , because it has no $\mathcal{M}$ components unlike tables A and B. Tables F and G have a common tuples distribution or pattern that both of them have one tuple which should be removed for each n tuple, so that both of the tables' measures showed the same values and ratio. Finally, table F shows the maximum difference between g_3 and g_4 , while table E shows the minimum difference.

Theorem 6 For any table T and set of attributes K , we have either $g_3(K) = g_4(K)$ or $1 < g_3(K)/g_4(K) < 2$. Furthermore, for any rational number $1 \leq \frac{p}{q} < 2$, there exist tables of arbitrarily large number of tuples with $g_3(K)/g_4(K) = \frac{p}{q}$.

Proof $g_3(K)$ and $g_4(K)$ differ only in the denominator part. The number of tuples of the components in $\mathcal{M}$ cannot be more than the total number of tuples in the table, so $0 \leq \sum_{M \in \mathcal{M}} |V_M| \leq |T|$ and $\sum_{M \in \mathcal{M}} |V_M| = |T|$ iff every tuple is a member of a connected component in $\mathcal{M}$. In the latter

case, $g_3(K) = g_4(K)$, otherwise the denominator of $g_4(K)$ is less than twice the denominator of $g_3(K)$ that proves the inequalities of the ratio.

Table *E* proves that $g_3(K) = g_4(K)$ can hold for arbitrarily large tables. Now let $1 < \frac{p}{q} < 2$ be given with $\frac{p}{q} = 1 + \frac{p'}{q'}$. Consider Table *D* where

$$g_3(K)/g_4(K) = \left(\frac{\beta - 1}{n + l}\right) / \left(\frac{\beta - 1}{n + 2l}\right)$$

which can simply be written as $1 + \frac{l}{n+l}$. Now, taking $n = \alpha(q' - p')$, $l = \alpha p'$ and any β between 2 and $\lfloor \frac{n}{2} \rfloor$, we obtain that

$$g_3(K)/g_4(K) = 1 + \frac{p'}{q'}.$$

Note that $g_3(K)$ ranges between $1/n$ and $1/2$ depending on the choice of β .

8 Conclusion and Future Directions

The main contributions of this paper are as follows:

- We introduced and defined strongly possible keys over database tables that contain some occurrences of nulls.
- We provided some properties and some necessary conditions so that a strongly possible key holds in a given dataset. We show that deciding whether a given set of attributes is a strongly possible key can be done by application of matchings in bipartite graph, so Hall's condition is naturally applied.
- We provided an algorithm to validate a strongly possible key by finding a proper strongly possible world for that key if there is any such a world. The algorithm was applied to real-world datasets to determine all two-element strongly possible key sets.
- We showed that deciding whether a given system of sets of attributes is a system of strongly possible keys for a given table can be done using matroid intersection. However, we need at least three matroids, and matroid intersection of three or more matroids is NP-complete, which suggests that our problem is also NP-complete.
- We studied systems of strongly possible keys, and we gave characterization of the implication problem.
- We showed that systems of strongly possible key constraints enjoy Armstrong instances provided they satisfy a natural necessary condition.
- We showed that certain keys are already characterized by strongly possible worlds, and one does not need to check all possible worlds.

- An approximation concept of the strongly possible key was introduced to measure how close approximation of a strongly possible key holds in a data relation, using g_3 measure. We derived the measure g_4 from g_3 and gave bounds of the two measures.

Strongly possible keys are special case of possible keys of relational schemata with each attribute having finite domain. So future research is needed to decide what properties of implication, axiomatization of inference remain valid in this setting. Note that the main results in [20] use that at least one attribute has infinite domain.

We plan to extend research from keys to functional dependencies. Weak and strong functional dependencies were introduced in [22]. A wFD $X \rightarrow_w Y$ holds if there is a possible world T' that satisfies FD $X \rightarrow Y$, while sFD $X \rightarrow_s Y$ holds if every possible world satisfies FD $X \rightarrow Y$. Our strongly possible world concept naturally induces an intermediate concept of functional dependency. Future research on possible keys of finite domains might extend our results on strongly possible keys.

Funding Open access funding provided by ELKH Alfréd Rényi Institute of Mathematics.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Acuna E, Rodriguez C (2007) The treatment of missing values and its effect in the classifier accuracy classification, clustering, and data mining applications, pp 639–648
2. Beeri C, Dowd M, Fagin R, Statman R (1984) On the structure of Armstrong relations for functional dependencies. *J ACM* 31(1):30–46
3. Date CJ (1990) NOT is not “not”! (notes on three-valued logic and related matters) in relational database writings 1985–1989. Addison-Wesley, Reading, MA
4. Dhevi ATS (2014) Imputing missing values using inverse distance weighted interpolation for time series data. In: Sixth international conference on advanced computing (ICoAC), pp 255–259
5. Chang G, Ge T (2011) Comparison of missing data imputation methods for traffic flow. In: International conference on transportation, mechanical, and electrical engineering (TMEE), pp 639–642
6. Cheng CH, Wei LY, Lin TC (2007) Improving relational database quality based on adaptive learning method for estimating null

- value. In: Second international conference on innovative computing, information and control (ICICIC), pp 81–81
7. Codd EF (1990) The relational model for database management, version 2. Addison-Wesley Publishing Company, Reading, MA
 8. Fagin R (1982) Horn clauses and database dependencies. *J ACM* 29(4):952–985
 9. Farhangfar A, Kurgan L, Pedrycz W (2004) Experimental analysis of methods for imputation of missing values in databases. *Intell Comput: Theory Appl II* 5421:172–183
 10. Farhangfar A, Kurgan LA, Pedrycz W (2007) A novel framework for imputation of missing values in databases. *IEEE Trans Syst Man Cybern-Part A: Syst Hum* 37(5):692–709
 11. Farhangfara A, Kurganb L, Dy J (2008) Impact of imputation of missing values on classification error for discrete data. *Pattern Recogn* 41(12):3692–3705
 12. Ferrarotti F, Hartmann S, Link S (2011) Codd table representations under weak possible world semantics. In: Database and expert systems applications. DEXA, LNCS, vol 6860, pp 125–139
 13. Garey MR, Johnson DS (1979) Computers and intractability. A guide to the theory of NP-completeness. Freeman, New York
 14. Giannella C, Robertson E (2004) On approximation measures for functional dependencies. *Inf Syst* 29(6):483–507
 15. Hartmann S, Kirchberg M, Link S (2012) Design by example for SQL table definitions with functional dependencies. *VLDB J* 21(1):121–144
 16. Hartmann S, Leck U, Link S (2010) Strong keys and functional dependencies in partial database relations. In: The fifth international conference on digital telecommunications ICDT
 17. Hartmann S, Leck U, Link S (2010) On Codd families of keys over incomplete relations. *Comput J* 54(7):1166–1180
 18. Grzymala-Busse JW, Hu M (2000) A comparison of several approaches to missing attribute values in data mining. In: Rough sets and current trends in computing RSCTC, pp 378–385
 19. Kivinen J, Mannila H (1995) Approximate inference of functional dependencies from relations. *Theor Comput Sci* 149(1):129–149
 20. Köhler H, Leck U, Link S, Zhou X (2016) Possible and certain keys for SQL. *VLDB J* 25(4):571–596
 21. Lawler EL (1975) Matroid intersection algorithms. *Math Program* 9:31–56
 22. Levene M, Loizou G (1998) Axiomatisation of functional dependencies in incomplete relations *Journal. Theor Comput Sci* 206(1–2):283–300
 23. Lovász L, Plummer MD (2009) Matching theory, vol 367. American Mathematical Society, Providence
 24. Mannila H, Rähä K-J (1992) Design of relational databases. Addison-Wesley, Boston
 25. Morrisett M (2013) Missing data in the relational model (Doctoral dissertation), Virginia Commonwealth University, 2013
 26. Sali A, Schewe K-D (2008) Keys and armstrong databases in trees with restructuring. *Acta Cybern* 18(3):529–556
 27. Welsh DJA (1976) Matroid theory. Academic Press, New York
 28. Zhang S, Qin Z, Ling CX, Sheng S (2005) “Missing is useful”: missing values in cost-sensitive decision trees. *IEEE Trans Knowl Data Eng* 17(12):1689–1693

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.