Automated Reasoning about Key Sets⋆

Miika Hannula and Sebastian Link

University of Auckland, Department of Computer Science, New Zealand
[m.hannula|s.link]@auckland.ac.nz

Abstract. Codd's rule of entity integrity stipulates that every table in a database has a primary key. Hence, the attributes that form the primary key carry no missing information and have unique value combinations. In practice, data records cannot always meet such requirements. Previous work has proposed the notion of a key set, which can identify more data records uniquely when information is missing. Apart from the proposal, key sets have not been investigated much further. We outline important database applications, and investigate computational limits and techniques to reason automatically about key sets. We establish a binary axiomatization for the implication problem of key sets, and prove its coNP-completeness. We show that perfect models do not always exist for key sets. Finally, we show that the implication problem for unary key sets by arbitrary key sets has better computational properties. The fragment enjoys a unary axiomatization, is decidable in time quadratic in the input, and perfect models can always be generated.

1 Introduction

Keys provide efficient access to data in database systems. They are required to understand the structure and semantics of data. For a given collection of entities, a key refers to a set of column names whose values uniquely identify an entity in the collection. For example, a key for a relational table is a set of columns such that no two different rows have matching values in each of the key columns. Keys are fundamental for most data models, including semantic models, object models, XML, RDF, and graphs. They advance many classical areas of data management such as data modeling, database design, and query optimization. Knowledge about keys empowers us to 1) uniquely reference entities across data repositories, 2) reduce data redundancy at schema design time to process updates efficiently at run time, 3) improve selectivity estimates in query processing, 4) feed new access paths to query optimizers that can speed up the evaluation of queries, 5) access data more efficiently via physical optimization such as data partitioning or the creation of indexes and views, and 6) gain new insight into application data. Modern applications create even more demand for keys. Here, keys facilitate data integration, help detect duplicates and anomalies, guide the repair of data, and return consistent answers to queries over dirty data. The discovery of keys from data sets is a core task of data profiling.

⋆ Research is supported by Marsden funding from the Royal Society of New Zealand.

Due to the demand in real-life applications, data models have been extended to accommodate missing information. The industry standard for data management, SQL, allows occurrences of a null marker to model any kind of missing value. Occurrences of the null marker mean that no information is available about an actual value of that row on that attribute, not even whether the value exists and is unknown nor whether the value does not exist. Codd's principle of entity integrity suggests that every entity should be uniquely identifiable. In SQL, this has led to the notion of a primary key. A primary key is a collection of attributes which stipulates uniqueness and completeness. That is, no row of a relation must have an occurrence of the null marker on any columns of the primary key and the combination of values on the columns of the primary key must be unique. The requirement to have a primary key over every table in the database is often inconvenient in practice. Indeed, it can happen easily that a given relation does not exhibit any primary key. This is illustrated by the following example.

Example 1. Consider the following snapshot of data from an accident ward at a hospital [15]. Here, we collect information about the *name* and *address* of a patient, who was treated for an *injury* in some *room* at some *time*.

room	*name*	*address*	*injury*	*time*
1	Miller	$\perp$	cardiac infarct	Sunday, 19
$\perp$	$\perp$	$\perp$	skull fracture	Monday, 19
2	Maier	Dresden	leg fracture	Sunday, 16
1	Miller	Pirna	leg fracture	Sunday, 16

Evidently, the snapshot does not satisfy any primary key since each column features some null marker occurrence, or a duplication of some value.

In response, several researchers proposed the notion of a key set. As the term suggests, a key set is a set of attribute subsets. Naturally, we call the elements of a key set a key. A relation satisfies a given key set if for every pair of distinct rows in the relation there is some key in the key set on which both rows have no null marker occurrences and non-matching values on some attribute of the key. The formal definition of a key set will be given in Definition 1 in Section 3. The flexibility of a key set over a primary key can easily be recognized, as a primary key would be equivalent to a singleton key set, with the only element being the primary key. Indeed, with a key set different pairs of rows in a relation may be distinguishable by different keys of the key set, while all pairs of rows in a relation can only be distinguishable by the same primary key. We illustrate the notion of a key set on our running example.

Example 2. The relation in Example 1 satisfies no primary key. Nevertheless, the relation satisfies several key sets. For example, the key set $\{\{room\}, \{time\}\}$ is satisfied, but not the key set $\{\{room, time\}\}$. The relation also satisfies the key sets $\mathcal{X}_1 = \{\{room, time\}, \{injury, time\}\}$ and $\mathcal{X}_2 = \{\{name, time\}, \{injury, time\}\}$, as well as the key set $\mathcal{X} = \{\{room, name, time\}, \{injury, time\}\}$.

It is important to point out a desirable feature that primary keys and key sets share. Both are independent of the interpretation of null marker occurrences. That is, any given primary key and any given key set is either satisfied or not, independently of what information any of the null marker occurrences represent. Primary keys and key sets are only dependent on actual values that occur in the relevant columns. This is achieved by stipulating the completeness criterion. The importance of this independence is particularly appealing in modern applications where data is integrated from various sources, and different interpretations may be associated with different occurrences of null markers.

Given the flexibility of key sets over primary keys, and given their independence of null marker interpretations, it seems natural to further investigate the notion of a key set. Somewhat surprisingly, however, neither the research community nor any system implementations have analyzed key sets since their original proposal in 1989. The main goal of this article is to take first steps into the investigation of computational problems associated with key sets. In database practice, one of the most fundamental problems is the implication problem. The problem is to decide whether for a given set $\Sigma \cup \{\varphi\}$ of key sets, every relation that satisfies all key sets in Σ also satisfies φ. Reasoning about the implication of any form of database constraints is important because efficient solutions to the problem enable us to facilitate the processing of database queries and updates.

Example 3. Recall the key sets $\mathcal{X}_1$, $\mathcal{X}_2$, and $\mathcal{X}$ from Example 2. An instance of the implication problem is whether $\Sigma = \{\mathcal{X}_1, \mathcal{X}_2\}$ implies the key set $\varphi = \mathcal{X}$, and another instance is whether Σ implies $\varphi' = \{\{room\}, \{name\}, \{address\}, \{time\}\}$.

Contributions. Our contributions can be summarized as follows.

- We compare the notion of a key set with other notions of keys. In particular, primary keys are key sets with just one element, and certain keys are unary key sets, for which every key is a singleton.
- We illustrate how automated reasoning tools for key sets can facilitate efficient updates and queries in database systems.
- We establish a binary axiomatization for the implication problem of key sets. Here, binary refers to the maximum number of premises that any inference rule in our axiomatization can have. This is interesting as all previous notions of keys enjoy unary axiomatizations, in particular primary keys. What that means semantically is that every given key set that is implied by a set of key sets is actually implied by at most two of the key sets.
- We establish that the implication problem for key sets is *coNP*-complete. Again, this complexity is quite surprising in comparison with the linear time decidability of other notions of keys.
- An interesting notion in database theory is that of Armstrong databases. A given class of constraints, such as keys, key sets, or other data dependencies [13], is said to enjoy Armstrong databases whenever for every given set of constraints in this class there is a single database with the property that for every constraint in the class, the database satisfies this constraint if and only if the constraint is implied by the given set of constraints. This is a powerful

property as multiple instances over the implication problem reduce to validating satisfaction over the same Armstrong database. Consequently, the generation of Armstrong databases would create 'perfect models' of a given constraint set, which has applications in the acquisition of requirements in database practice. We show that key sets do not enjoy Armstrong relations, as opposed to other classes of keys known from the literature.
- We then identify an expressive fragment of key sets for which the associated implication problem can be characterized by a unary axiomatization and a quadratic-time algorithm. The fragment also enjoys Armstrong relations and we show how to generate them with conservative use of time and space.

Organization. We discuss related work in Section 2. Basic notions and notation are fixed in Section 3. Section 4 discusses applications of key sets in the processing of queries and updates. An axiomatization for key sets is established in Section 5. The *coNP*-completeness of the implication problem is settled in Section 6. The general existence of Armstrong relations is dis-proven in Section 7. A computationally friendly fragment of key sets is identified in Section 8. We conclude and briefly discuss future work in Section 9.

2 Related Work

We provide a concise discussion on the relationship of key sets with other notions of keys over relations with missing information.

Codd is the inventor of the relational model of data [4]. He proposed the rule of entity integrity, which stipulates that every entity in every table should be uniquely identifiable. In SQL that led to the introduction of primary keys, which stipulate uniqueness and completeness on the attributes that form the primary key. The primary key is a distinguished candidate key. We call an attribute set a *candidate key* for a given relation if and only if every pair of distinct tuples in the relation has no null marker occurrences on any of the attributes of the candidate key and there is some attribute of the candidate key on which the two tuples have non-matching values. The notions of primary and candidate keys have been introduced very early in the history of database research [12]. Candidate keys are singleton key sets, that is, key sets with just one element (namely the candidate key). Hence, instead of having to be complete and unique on the same combination of columns in a candidate key, key sets offer different alternatives of being complete and unique for different pairs of tuples in a relation. Candidate keys were studied in [7]. In that work, the associated implication problem was characterized axiomatically and algorithmically, the automatic generation of Armstrong relations was established, and extremal problems associated with families of candidate keys were investigated. As Example 1 shows, there are relations on which no candidate key holds, but which satisfy key sets.

Lucchesi and Osborn studied computational problems associated with candidate keys [12]. However, their focus was an algorithm that finds all candidate keys implied by a given set of functional dependencies. They also proved that deciding whether a given relation satisfies some key of cardinality not greater than some given positive integer is NP-complete. Recently, this problem was shown to be W[2]-complete in the size of the key [2]. The discovery which key sets hold on a given relation is beyond the scope of this paper and left as an open problem for future work.

Key sets were introduced by Thalheim [14] as a generalization of Codd's rule for entity integrity. He studied combinatorial problems associated with unary key sets, such as the maximum cardinality that non-redundant families of unary key sets can have, and which families attain them [13, 15]. Key sets were further discussed by Levene/Loizou [11] where they also generalized Codd's rule for referential integrity. Somewhat surprisingly, the study of the implication problem for key sets has not been addressed by previous work. This is also true for other automated tasks which require reasoning about key sets.

More recently, the notions of possible and certain keys were proposed [8]. These notions are defined for relations in which null marker occurrences are interpreted as 'no information', and possible worlds of an incomplete relation are obtained by independently replacing null marker occurrences by actual domain values (or the N/A marker indicating that the value does not exist). A key is said to be *possible* for an incomplete relation if and only if there is some possible world of the incomplete relation on which the key holds. A key is said to be *certain* for an incomplete relation if and only if the key holds on every possible world of the incomplete relation. For example, the relation in Example 1 satisfies the possible key $p\langle room, name, address\rangle$, since the key $\{room{,}name{,}address\}$ holds on the possible world:

room	*name*	*address*	*injury*	*time*
1	Miller	Dresden	cardiac infarct	Sunday, 19
2	Maier	Pirna	skull fracture	Monday, 19
2	Maier	Dresden	leg fracture	Sunday, 16
1	Miller	Pirna	leg fracture	Sunday, 16

of the relation. In contrast, the key $\{room{,}name\}$ is not possible for the relation because the first and last tuple will have matching values on room and name in every possible world of the relation. The key $\{address\}$ is possible, but not certain, and the key $\{room{,}time\}$ is certain for the given relation. Now, it is not difficult to see that an incomplete relation satisfies the certain key $c\langle A_1, \ldots, A_n\rangle$ if and only if the relation satisfies the key set $\{\{A_1\}, \ldots, \{A_n\}\}$. In this sense, certain keys correspond to key sets which have only singleton keys as elements. The papers [8] investigate computational problems for possible and certain keys with `NOT NULL` constraints. In the current paper we investigate a different class of key constraints, namely key sets. In particular, the computationally-friendly fragment of key sets we identify in Section 8 subsumes the class of certain keys as the special case of unary key sets.

Recently, *contextual keys* were introduced as a means to separate completeness from uniqueness requirements [16]. A contextual key is an expression (C, X) where $X \subseteq C$. These are different from key sets since $X \subseteq C$ is a key for only those tuples that are complete on C. In particular, the special case where $C = X$

only requires uniqueness on X for those tuples that are complete on X. This captures the `UNIQUE` constraint of SQL. We leave it as future work to investigate contextual key sets.

3 Preliminary Definitions

In this section, we give some basic definitions and fix notation.

A *relation schema* is a finite non-empty set of attributes, usually denoted by R. A *relation* r over R consists of tuples t that map each $A \in R$ to $\mathrm{Dom}(A) \cup \{\bot\}$ where $\mathrm{Dom}(A)$ is the domain associated with attribute A and $\bot$ is the unique null marker. Given a subset X of R, we say that a tuple t is X*-total* if $t(A) \neq \bot$ for all $A \in X$. Informally, a relation schema represents the column names of database tables, while each tuple represents a row of the table, so a relation forms a database instance. Moreover, $\mathrm{Dom}(A)$ represents the possible values that can occur in column A of a table, and $\bot$ represents missing information. That is, if $t(A) = \bot$, then there is no information about the value $t(A)$ of tuple t on attribute A.

In our running example, we have the relation schema

$$\textsc{Ward}=\{\mathit{room},\mathit{name},\mathit{address},\mathit{injury},\mathit{time}\}.$$

Each of these attributes comes with a domain, which we do not specify any further here. Each row of the table in Example 1 represents a tuple. The second row, for example, is $\{\mathit{injury}, \mathit{time}\}$-total, but not total on any proper superset of $\{\mathit{injury}, \mathit{time}\}$. The four tuples together constitute a relation over Ward.

The following definition introduces the central object of our studies. It was first defined by Thalheim in [14].

Definition 1. *A* key set *is a finite, non-empty collection $\mathcal{X}$ of subsets of a given relation schema R. We say that a relation r over R* satisfies *the key set $\mathcal{X}$ if and only if for all distinct $t, t' \in r$ there is some $X \in \mathcal{X}$ such that t and t' are X-total and $t(X) \neq t'(X)$. Each element of a key set is called a* key*. If all keys of a key set are singletons, we speak of a* unary key set*.*

In the sequel we write $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \ldots$ for key sets and $X, Y, Z, \ldots$ for attribute sets, and $A, B, C, \ldots$ for attributes. We sometimes write A instead of $\{A\}$ to denote the singleton set consisting of only A. If $\boldsymbol{X}$ is a sequence, then we may sometimes write simply $\boldsymbol{X}$ for the set that consists of all members of $\boldsymbol{X}$.

As already mentioned in Example 2, the relation in Example 1 satisfies the key sets $\mathcal{X}_1$, $\mathcal{X}_2$, and $\mathcal{X}$. It also satisfies the unary key set $\{\{\mathit{room}\}, \{\mathit{time}\}\}$, but not the singleton key set $\{\{\mathit{room}, \mathit{time}\}\}$.

A fundamental problem in automated reasoning about any class of constraints is the *implication problem*. For key sets, the problem is to decide whether for an arbitrary relation schema R, and an arbitrary set $\Sigma \cup \{\varphi\}$ of key sets over R, Σ implies φ. Indeed, Σ *implies* φ if and only if every relation over R that satisfies all key sets in Σ also satisfies the key set φ. The following section illustrates how solutions to the implication problem of key sets can facilitate the efficient processing of queries and updates.

4 Applications for Automated Reasoning

The most important applications of data processing are updates and queries. We briefly describe in this section how automated reasoning about key sets can facilitate each of these application areas.

4.1 Efficient Updates

When databases are updated it must be ensured that the resulting database satisfies all the constraints that model the business rules of the underlying application domain. Violations of the constraints indicate sources of inconsistency, and an alert of such inconsistencies should at least be issued to the database administrator. This is to ensure that appropriate actions can be taken, for example, to disallow the update. This quality assurance process incurs an overhead in terms of the time it takes to validate the constraints. As such, users of the database expect that such overheads are minimized. In particular, the time on validating constraints increases with the volume of the database. As a principal, the set of constraints that are specified on the database and therefore subject to validation upon updates, should be non-redundant. That is, no constraints should be specified that are already implied by other specified constraints. The simple reason is that the validation of any implied constraints is a waste of time because the validity of the other constraints already ensures that any implied constraint is valid as well. This is a strong real-life motivation for developing tools that can decide implication. In our running example, the set $\Sigma = \{\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}\}$ of key sets is redundant because the subset $\Sigma' = \{\mathcal{X}_1, \mathcal{X}_2\}$ implies the key set $\mathcal{X}$. Automated solutions to the implication problem can thus automatize the minimization of overheads in validating constraints under database updates.

4.2 Efficient Queries

We are interested in the names of patients that can be identified uniquely based on information about their name and the room and time at the accident ward, or based on information about their injury and the time at the accident ward. In SQL, this may be expressed as follows.

```
   SELECT name
     FROM WARD
    WHERE room IS NOT NULL AND name IS NOT NULL AND
          time IS NOT NULL
 GROUP BY room, name, time
   HAVING count(room, name, time) ≤ 1
    UNION
   SELECT name
     FROM WARD
    WHERE injury IS NOT NULL AND time IS NOT NULL
 GROUP BY injury, time
   HAVING count(injury, time) ≤ 1 ;
```

Upward closure	Refinement	Composition
$\dfrac{\mathcal{X}}{\mathcal{X}\cup\mathcal{Y}}$	$\dfrac{\mathcal{X}\cup\{XY\}}{\mathcal{X}\cup\{X,Y\}}$	$\dfrac{\mathcal{X}_1 \quad \mathcal{X}_2}{\{Z_{(X_1,X_2)} \mid (X_1,X_2)\in\mathcal{X}_1\times\mathcal{X}_2\}}$ $Z_{(X_1,X_2)}\subseteq X_1\cup X_2$, and $X_1\subseteq Z_{(X_1,X_2)}$ or $X_2\subseteq Z_{(X_1,X_2)}$

Table 1: An axiomatization $\mathfrak{A}$ for key sets

Knowing that the underlying relation over WARD satisfies the two key sets $\mathcal{X}_1$ and $\mathcal{X}_2$ and that the key set $\mathcal{X}=\{\{room, name, time\},\{injury, time\}\}$ is implied by $\mathcal{X}_1$ and $\mathcal{X}_2$, one can deduce that every tuple of WARD must be in at least one of the sub-query results of the `UNION` query. That is, the query above can be simplified to

`SELECT DISTINCT` *name*
`FROM` WARD ;

Note that the `DISTINCT` word is necessary since the `UNION` operator eliminates duplicates. When evaluated on the example from the introduction, each query will return the result $\{$(*name*: Miller),(*name*: $\bot$), (*name*: Maier)$\}$.

Motivated by the applications of key sets for data processing and the lack of knowledge on automated reasoning tasks associated with key sets, the following sections will investigate the implication problem for key sets.

5 Axiomatizing Key Sets

In this section we establish axiomatizations for arbitrary key sets as well as unary ones. This will enable us to effectively enumerate all implied key sets, that is, to determine the semantic closure $\Sigma^*=\{\sigma \mid \Sigma\models\sigma\}$ of any given set Σ of key sets. A finite axiomatization facilitates human understanding of the interaction of the given constraints, and ensures all opportunities for the use of these constraints in applications can be exploited.

In using an axiomatization we determine the semantic closure by applying *inference rules* of the form $\frac{\text{premise}}{\text{conclusion}}$. For a set $\mathfrak{R}$ of inference rules let $\Sigma\vdash_{\mathfrak{R}}\varphi$ denote the *inference* of φ from Σ by $\mathfrak{R}$. That is, there is some sequence $\sigma_1,\ldots,\sigma_n$ such that $\sigma_n=\varphi$ and every σ_i is an element of Σ or is the conclusion that results from an application of an inference rule in $\mathfrak{R}$ to some premises in $\{\sigma_1,\ldots,\sigma_{i-1}\}$. Let $\Sigma^+_{\mathfrak{R}}=\{\varphi \mid \Sigma\vdash_{\mathfrak{R}}\varphi\}$ be the *syntactic closure* of Σ under inferences by $\mathfrak{R}$. $\mathfrak{R}$ is *sound* (*complete*) if for every set Σ over every R we have $\Sigma^+_{\mathfrak{R}}\subseteq\Sigma^*$ ($\Sigma^*\subseteq\Sigma^+_{\mathfrak{R}}$). The (finite) set $\mathfrak{R}$ is a (finite) *axiomatization* if $\mathfrak{R}$ is both sound and complete.

Table 1 shows a finite axiomatization $\mathfrak{A}$ for key sets. A non-trivial rule is **Composition** which is illustrated by our running example.

Example 4. Recall Example 1 from the introduction, in particular $\Sigma=\{\mathcal{X}_1,\mathcal{X}_2\}$ and $\varphi=\mathcal{X}$. It turns out that φ is indeed implied by Σ, since φ can be inferred from Σ by an application of the Composition rule, and the rule is sound for the implication of key sets. Indeed, $\mathcal{X}_1 \times \mathcal{X}_2$ consists of:

$$(\{\mathit{room},\mathit{time}\}, \{\mathit{name},\mathit{time}\}),$$
$$(\{\mathit{room},\mathit{time}\}, \{\mathit{injury},\mathit{time}\}),$$
$$(\{\mathit{injury},\mathit{time}\}, \{\mathit{name},\mathit{time}\}), \text{ and}$$
$$(\{\mathit{injury},\mathit{time}\}, \{\mathit{injury},\mathit{time}\}) .$$

and for each element $X = (X_1, X_2)$ we need to pick one attribute set Z_X that is contained in the union $X_1 \cup X_2$ and contains either X_1 or X_2. For the first element we pick $\{\mathit{room}, \mathit{time}, \mathit{name}\}$, and for the remaining three elements we pick $\{\mathit{injury}, \mathit{time}\}$. That results in the key set $\mathcal{X}$.

We now proceed with the completeness proof for the axiom system $\mathfrak{A}$ of Table 1. The proof proceeds in three stages. First in Lemma 1, we show a characterization of the implication problem. This is applied in Lemma 2 to show that $\mathfrak{A}$ extended with n-ary Composition for all $n \in \mathbb{N}$ is complete (see Table 2). At last, we show in Lemma 3 that n-ary Composition can be simulated with the binary Composition of $\mathfrak{A}$.

$$\frac{\mathcal{X}_1 \quad \dots \quad \mathcal{X}_n}{\{Z_{\boldsymbol{X}} \mid \boldsymbol{X} \in \mathcal{X}_1 \times \dots \times \mathcal{X}_n\}} \qquad Z_{\boldsymbol{X}} \subseteq \bigcup \boldsymbol{X} \text{ and } \bigvee_i X_i \subseteq Z_{\boldsymbol{X}}$$

Table 2: The n-ary Composition rule

Lemma 1. *$\{\mathcal{X}_1, \dots, \mathcal{X}_n\} \models \mathcal{Y}$ iff for all $(X_1, \dots, X_n) \in \mathcal{X}_1 \times \dots \times \mathcal{X}_n$ there is $\mathcal{Z} \subseteq \mathcal{Y}$ such that $\bigcup \mathcal{Z} \subseteq \bigcup_i X_i$, and $X_i \subseteq \bigcup \mathcal{Z}$ for some i.*

Proof. Assume first that one finds such an $\mathcal{Z}$. We show that any relation r that satisfies each $\mathcal{X}_i$ satisfies also $\mathcal{Y}$. Let t, t' be two tuples from r. Then for some $(X_1, \dots, X_n) \in \mathcal{X}_1 \times \dots \times \mathcal{X}_n$, t and t' are both $\bigcup_i X_i$-total and disagreeing on each X_i. Assume that i is such that $X_i \subseteq \bigcup \mathcal{Z}$, and let $A \in X_i$ be such that $t(A) \neq t'(A)$. Then selecting some $Z \in \mathcal{Z}$ such that it also contains A, we have that t and t' are Z-total and deviate on Z. Thus Z is witness for $r \models \mathcal{Y}$.

For the other direction we assume that no such $\mathcal{Z}$ exists. Then there is $(X_1, \dots, X_n) \in \mathcal{X}_1 \times \dots \times \mathcal{X}_n$ such that for $\mathcal{Z} := \{Z \in \mathcal{Y} \mid Z \subseteq \bigcup_i X_i\}$, $X_i \not\subseteq \bigcup \mathcal{Z}$ for all i. Then, selecting an attribute A_i from $X_i \setminus \bigcup \mathcal{Z}$ for all i, we may construct a relation r satisfying $\{\mathcal{X}_1, \dots, \mathcal{X}_n, \neg\mathcal{Y}\}$. This relation r consists of two tuples t, t' where t is a constant function mapping all of R to 0, and t' maps $\bigcup_i A_i$ to 1, $\bigcup_i X_i \setminus \bigcup_i A_i$ to 0, and all the remaining attributes to $\perp$. Now, obviously r satisfies all $\mathcal{X}_i$. Furthermore, for $Y \in \mathcal{Y} \setminus \mathcal{Z}$, t' is not Y-total, and for $Y \in \mathcal{Y} \cap \mathcal{Z}$ both t and t' are Y-total but with constant values 0. Therefore, r is a witness of $\{\mathcal{X}_1, \dots, \mathcal{X}_n\} \not\models \mathcal{Y}$ which concludes the proof. $\square$

Notice that the latter condition of Lemma 1 can be equivalently stated as $X_i \subseteq \bigcup\{Y \in \mathcal{Y} \mid Y \subseteq \bigcup_i X_i\}$ for some i.

Lemma 2. *The axiomatization* $\mathfrak{A}$ *extended with* n*-ary Composition is complete for key sets.*

Proof. Assume $\{\mathcal{X}_1, \ldots, \mathcal{X}_n\} \models \mathcal{Y}$. Then we obtain by Lemma 1 for all $\boldsymbol{X} = (X_1, \ldots, X_n) \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ a subset $\mathcal{Z}_{\boldsymbol{X}} \subseteq \mathcal{Y}$ such that $\bigcup \mathcal{Z}_{\boldsymbol{X}} \subseteq \bigcup \boldsymbol{X}$, and $X_i \subseteq \bigcup \mathcal{Z}_{\boldsymbol{X}}$ for some i. Then by Composition we may derive $\{\bigcup \mathcal{Z}_{\boldsymbol{X}} \mid \boldsymbol{X} \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n\}$. With repeated applications of Refinement we then derive $\bigcup\{\mathcal{Z}_{\boldsymbol{X}} \mid \boldsymbol{X} \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n\}$. Since this set is a subset of $\mathcal{Y}$, we finally obtain $\mathcal{Y}$ with a single application of Upward closure. □

Lemma 3. n*-ary Composition is derivable in* $\mathfrak{A}$.

Proof. Assume that $\mathcal{K} = \{Z_{\boldsymbol{X}} \mid \boldsymbol{X} \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n\}$ is obtained from $\mathcal{X}_1, \ldots, \mathcal{X}_n$ by an application of n-ary Composition. We will perform consecutive applications of (binary) Composition until we have obtained $\mathcal{K}$. Composition is applied incrementally so that the first application of this rule combines $\mathcal{X}_1$ and $\mathcal{X}_2$ to obtain a new key set $\mathcal{X}$, the second combines $\mathcal{X}$ and $\mathcal{X}_3$ to obtain the next key set $\mathcal{X}'$, the third $\mathcal{X}'$ and $\mathcal{X}_4$ to obtain $\mathcal{X}''$, and so forth. Once $\mathcal{X}_n$ is reached the cycle is started again from $\mathcal{X}_1$.

At each step of the aforementioned procedure we have deduced a key set $\mathcal{X}$ such that each $X \in \mathcal{X}$ either is a union $\bigcup \mathcal{Y}_1 \cup \ldots \cup \bigcup \mathcal{Y}_n$ for $\mathcal{Y}_i \subseteq \mathcal{X}_i$, or belongs to the required key set $\mathcal{K}$. In the previous case, provided that each $\mathcal{Y}_i$ is the maximal subset of $\mathcal{X}_i$ such that $\bigcup \mathcal{Y}_i \subseteq X$, we refer to $\mathcal{Y}_1 \cup \ldots \cup \mathcal{Y}_n$ as the *maximal decomposition* of X and $|\mathcal{Y}_1 \cup \ldots \cup \mathcal{Y}_n|$ as the *decomposition size* of X. Furthermore, given a set $Z_{\boldsymbol{X}} \in \mathcal{K}$ where $\boldsymbol{X} \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ we say that a set $X_i \in \boldsymbol{X}$ is *full* in $Z_{\boldsymbol{X}}$ if $X_i \subseteq Z_{\boldsymbol{X}}$. By the prerequisite of the n-ary Composition some member of $\boldsymbol{X}$ is always guaranteed to be full in $Z_{\boldsymbol{X}}$.

Initialization. Consider an instance of n-ary Composition. We initialize the procedure by applying Composition $n-1$ many times so that we obtain the key set $\{\bigcup \boldsymbol{X} \mid \boldsymbol{X} \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n\}$. This is done by letting $\mathcal{U}_1 := \mathcal{X}_1$ and taking the key set $\mathcal{U}_{i+1} = \{X_1 \cup X_2 \mid (X_1, X_2) \in \mathcal{U}_i \times \mathcal{X}_{i+1}\}$ for $i = 1, \ldots, n-1$.

Inductive step. After the initial step we have reached a key set $\mathcal{V}_1 := \mathcal{U}_n$ such that all $X \in \mathcal{V}_1 \setminus \mathcal{K}$ have decomposition size at least 1. Assume now that we have reached a key set $\mathcal{V}_m$ such that all $X \in \mathcal{V}_m \setminus \mathcal{K}$ have decomposition size at least m. As the induction step we show how to obtain a key set $\mathcal{V}_{m+1}$ such that every member of $\mathcal{V}_{m+1} \setminus \mathcal{K}$ has decomposition size at least $m+1$. This is done by taking a single round of applications of Composition to $\mathcal{V}_m$ and $\mathcal{X}_1, \ldots, \mathcal{X}_n$. That is, $\mathcal{V}_m$ and $\mathcal{X}_1$ are first combined using Composition, then the outcome is combined with $\mathcal{X}_2$, and its outcome with $\mathcal{X}_3$, and so forth until we have applied this procedure to $\mathcal{X}_n$. All these applications keep the members of $V_m \cap \mathcal{K}$ fixed. For instance, at the first step $Z_{(X,Y)}$ for $X \in \mathcal{V}_m \cap \mathcal{K}$ and any $Y \in \mathcal{X}_1$ is defined as X. We show how this deduction handles an arbitrary $X \in \mathcal{V}_m \setminus \mathcal{K}$.

By induction assumption each $X \in \mathcal{V}_m \setminus \mathcal{K}$ has decomposition size at least m. Let $\bigcup \mathcal{Y}_1 \cup \ldots \cup \bigcup \mathcal{Y}_n$ be the maximal decomposition of X. Now, assume towards a contradiction that for each i there is $Y_i \in \mathcal{Y}_i$ such that Y_i is not full in any $Z_{\boldsymbol{Y}} \in \mathcal{K}$ where $\boldsymbol{Y} \in \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_n$ and Y_i is the ith member of $\boldsymbol{Y}$. Then, however, the diagonal $\boldsymbol{Y}' = (Y_1, \ldots, Y_n)$ must have a member that is full in $Z_{\boldsymbol{Y}'}$.

This is a contradiction and hence there is i such that all $Y_i \in \mathcal{Y}_i$ are full in some $Z_{\boldsymbol{Y}} \in \mathcal{K}$ where $\boldsymbol{Y} \in \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_n$ and Y_i is the ith member of $\boldsymbol{Y}$. With regards to X, Composition is then applied as follows. For the first $i-1$ applications X is kept fixed. For the ith application that considers $\mathcal{X}_i$, each pair of X and $Y \in \mathcal{Y}_i$ is transformed to that $Z_{\boldsymbol{Y}} \in \mathcal{K}$ in which Y is full. Furthermore, each pair of X and $Y \in \mathcal{X}_i \setminus \mathcal{Y}_i$ is transformed to XY. Take note that the decomposition size of XY is at least $n+1$. At last, the remaining applications of Composition keep the obtained sets fixed. Since this procedure is applied to all $X \in \mathcal{V}_m \setminus \mathcal{K}$, we obtain that $\mathcal{V}_{m+1} \setminus \mathcal{K}$ has only sets with decomposition size at least $m+1$. This concludes the induction step.

Now, $\mathcal{V}_{M+1}$ where $M = |\mathcal{X}_1 \cup \ldots \cup \mathcal{X}_n|$ is a subset of $\mathcal{K}$. Hence, we conclude that $\mathcal{V}_{M+1}$ yields $\mathcal{K}$ with one application of Upward closure. □

Note that a simulation of one application of n-ary of Composition to $\{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ takes at most $(n+1) \cdot |\bigcup_{i=1}^{n} \mathcal{X}_i|$ applications of binary Composition plus one application of Upward Closure.

The previous three lemmata now generate the following axiomatic characterization of key set implication. We omit the soundness proof which is straightforward to check.

Theorem 1. *The axiomatization $\mathfrak{A}$ is sound and complete for key sets.*

Another important application. A direct application of an axiomatization is the efficient representation of collections of key sets. Similar to the computation of non-redundant covers during update operations, removing any redundant constraints makes the result easier to understand by humans. This is, for example, important for the discovery problem of key sets in which one attempts to efficiently represent all those key sets that a given relation satisfies. Even more directly, one can understand any sound inference rule as an opportunity to apply pruning techniques as part of a discovery algorithm. A complete axiomatization ensures all opportunities for the pruning of a search space can be exploited.

6 Complexity of Key Set Implication

In this section we settle the exact computational complexity of the implication problem for key sets. While the implication problem for most notions of keys over incomplete relations is decidable in linear time, the implication problem for key sets is likely to be intractable. This should also be seen as evidence for the expressivity of key sets.

Theorem 2. *The implication problem for key sets is coNP-complete.*

Proof. Consider first the membership in $\mathsf{co}-\mathsf{NP}$. By Lemma 1, for determining whether $\{\mathcal{X}_1, \ldots, \mathcal{X}_n\} \not\models \mathcal{Y}$, it suffices to choose $X_1, \ldots, X_n$ respectively from $\mathcal{X}_1, \ldots, \mathcal{X}_n$, and then deterministically check that $X_i \not\subseteq \bigcup \mathcal{Z}$ for all i, where $\mathcal{Z}$ is selected deterministically as $\mathcal{Z} := \{Z \in \mathcal{Y} \mid Z \subseteq \bigcup_i X_i\}$.

For the hardness, we reduce from the complement of 3-SAT. Let $C_1, \dots, C_n$ be a collection of clauses, each consisting of three literals, i.e., propositions of the form p or negated propositions of the form $\neg p$. Let P be the set of all proposition symbols that appear in some C_i, and let $\overline{P}$ consist of their negations. Letting $P \cup \overline{P}$ be our relation schema, we show that $\bigwedge_i \bigvee C_i$ has a solution iff $\{\{p, \neg p\} \mid p \in P\} \not\models \{C_1, \dots, C_n\}$. Notice that the antecedent is a set of singleton key sets, each of size two.

Assume first that there is a solution. Let $S \subseteq \mathcal{P}(P)$ encode the complement of that solution, i.e., S is such that each C_i contains some $p \notin S$ or some $\neg p$ for $p \in S$. Let $\overline{S} = \{\neg p \mid p \notin S\}$, and define singleton sets $X_p = \{p, \neg p\} \cap (S \cup \overline{S})$, encoding those literals that are set false by the solution. Then $C_i \not\subseteq \bigcup_p X_p$ for all i, implying be Lemma 1 that $\{\{p, \neg p\} \mid p \in P\} \not\models \{C_1, \dots, C_n\}$.

Assume then that $\{\{p, \neg p\} \mid p \in P\} \not\models \{C_1, \dots, C_n\}$. By Lemma 1 we find $X_p \in \{p, \neg p\}$ such that for no $\mathcal{Z} \subseteq \{C_1, \dots, C_n\}$ we have that $\bigcup \mathcal{Z} \subseteq \bigcup_p X_p$ and $\bigvee_p X_p \subseteq \bigcup \mathcal{Z}$. Now, $C_i \subseteq \bigcup_p X_p$ implies $X_p \subseteq C_i$ for three distinct p, and therefore we must have $C_i \not\subseteq \bigcup_p X_p$ for all i. It is now easy to see that the sets X_p give rise to a solution to the satisfiability problem. □

7 Armstrong Relations

In this section we ask the basic question whether key sets enjoy Armstrong relations. These are special models which are perfect for a given collection of key sets. More formally, a given relation r is said to be *Armstrong* for a given set Σ of key sets if and only if for all key sets φ it is true that r satisfies φ if and only if Σ implies φ. Indeed, an Armstrong relation is a perfect model for Σ since it satisfies all keys sets implied by Σ and does not satisfy any key set that is not implied by Σ. Armstrong relations have important applications in data profiling [1] and the requirements acquisition phase of database design [10].

Unfortunately, arbitrary sets of key sets do not enjoy Armstrong relations as the following result manifests.

Theorem 3. *There are sets of key sets for which no Armstrong relations exist.*

Proof. An example is $\Sigma = \{\{\{A\}, \{B\}\}, \{\{C\}, \{D\}\}\}$ with attributes A, B, C, D. Then $\sigma_1 = \{\{A, C\}, \{A, D\}, \{B, C\}\}$ and $\sigma_2 = \{\{A, D\}, \{B, C\}, \{B, D\}\}$ are two non-consequences of Σ, respectively exemplified by the two 2-tuple relations on the left of Figure 1, where "d" refers to any distinct total value.

These are the only possible types of tuple pairs that satisfy $\Sigma \cup \{\neg\sigma_1\}$ and $\Sigma \cup \{\neg\sigma_2\}$, respectively. Therefore, we observe that any relation r satisfying Σ and refuting both σ_1 and σ_2 has a homomorphism from a relation of the form on the right of Figure 1 to a subset of r with the condition that this homomorphism preserves nulls and maps domain values to domain values. However, then neither $\{\{A\}, \{B\}\}$ nor $\{\{C\}, \{D\}\}$ is a key set anymore. □

A	B	C	D
d	d	$\bot$	d
$\bot$	d	d	d

A	B	C	D
d	d	d	$\bot$
d	$\bot$	d	d

A	B	C	D
d	d	$\bot$	d
$\bot$	d	d	d
d	d	d	$\bot$
d	$\bot$	d	d

Fig. 1

8 Implication for Unary by Arbitrary Key Sets

In this section we identify a fragment of key sets for which automated reasoning is efficient. This is strongly motivated by the results of the previous sections in which the coNP-completeness of the implication problem, and the lack of general Armstrong relations has been established. Indeed, the fragment is the implication of unary key sets by arbitrary key sets. We show that this fragment is captured axiomatically by the Refinement and Upward Closure rules, can be decided in time quadratic in the input, and Armstrong relations always exist and can be computed with conservative use of time and space.

8.1 An algorithmic characterization

Our first result establishes that unary key sets must be implied by a single key set from the given collection of key sets.

Theorem 4. *Let $\Sigma = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ be a collection of arbitrary key sets, and let $\varphi = \{\{A_1\}, \ldots, \{A_k\}\}$ be a unary key set over relation schema R. Then Σ implies φ if and only if there is some $i \in \{1, \ldots, n\}$ such that $\bigcup \mathcal{X}_i \subseteq \{A_1, \ldots, A_k\}$.*

Proof. If $\bigcup \mathcal{X}_i \subseteq X$ for some $i \in \{1, \ldots, n\}$, Refinement and Upward Closure infer φ from Σ. Due to the rules' soundness, φ is implied by Σ.

Vice versa, assume that $\bigcup \mathcal{X}_i \not\subseteq X$ holds for all $i = 1, \ldots, n$. Let r be defined as $r = \{t, t'\}$ where t and t' are two total tuples that agree on $X = \{A_1, \ldots, A_k\}$ and disagree elsewhere. It follows that r violates φ. Since $\bigcup \mathcal{X}_i \not\subseteq X$ for all $i = 1, \ldots, n$, t_1 and t_2 must differ on some attribute in $\bigcup \mathcal{X}_i$ for $i = 1, \ldots, n$. This means, r satisfies all key sets in Σ. Consequently, Σ does not imply φ. □

A direct consequence of Theorem 4 is the quadratic time complexity of the implication problem for unary by arbitrary key sets. For a collection Σ of key sets let $|\Sigma|$ denote the total number of attribute occurrences in elements of Σ.

Corollary 1. *The implication problem of unary key sets by arbitrary key sets is decidable in time $\mathcal{O}(|\Sigma| \times |\varphi|)$ in the input $\Sigma \cup \{\varphi\}$.*

8.2 A finite axiomatization

Our next result establishes a finite axiomatization for the implication of unary by arbitrary key sets that consists of the Refinement and Upward Closure rules. As this fragment is decidable in time quadratic in the input, and the general case is *coNP*-complete, the Composition rule is the source of likely intractability.

Corollary 2. *The implication problem of unary key sets by arbitrary key sets has a sound and complete axiomatization in Refinement and Upward Closure.*

Proof. Let $\Sigma = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ be a set of key sets, and let $\varphi = \{\{A_1\}, \ldots, \{A_k\}\}$ be a unary key set over relation schema R. If φ can be inferred from Σ by a sequence of applications of the Refinement and Upward Closure rules, the soundness of these rules ensures that φ is also implied by Σ.

For completeness we assume that φ cannot be inferred from Σ by means of applications using the Refinement and Upward Closure rules. Hence, $\bigcup \mathcal{X}_i \not\subseteq X$ holds for all $i = 1, \ldots, n$. Theorem 4 shows that Σ does not imply φ. □

8.3 Existence and computation of Armstrong relations

Armstrong models relative to unary consequences are also easy to obtain. It merely suffices to take a disjoint union of all of the two tuple relations mentioned in the proof of Theorem 4.

Corollary 3. *The implication problem of unary key sets by arbitrary key sets has Armstrong relations.* □

While the existence of perfect models is easy to come by the disjoint union construction, an actual generation of Armstrong relations by this construction is not efficient. Smaller Armstrong relations can be constructed as follows. Theorem 4 shows that the implication problem of unary key sets $\mathcal{X}$ by a collection $\Sigma = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ of arbitrary key sets only depends on the attributes contained in each given key set of Σ, and not on how they are grouped as sets in a key set. We thus identify, without loss of generality, $\mathcal{X}$ with $\bigcup \mathcal{X}$ and each $\mathcal{X}_i$ with $\bigcup \mathcal{X}_i$.

The idea is then to compute so-called anti-keys, which are the maximal subsets of the underlying relation schema which are key sets not implied by Σ. Given the anti-keys, an Armstrong relation for Σ can be generated by starting with a single complete tuple, and introducing for each anti-key a new tuple that has matching total values on the attributes of the anti-key and unique values on attributes outside the anti-key. This construction ensures that all non-implied (unary) key sets are violated and all given key sets are satisfied. The computation of the anti-keys from Σ can be done by taking the complements of the minimum transversals of the hypergraph formed by the elements of Σ. A transversal for a given set of attribute subsets $\mathcal{X}_i$ is an attribute subset $\mathcal{T}$ such that $\mathcal{T} \cap \mathcal{X}_i \neq \emptyset$ holds for all i. While many efficient algorithms exist for the computation of all hypergraph transversals, it is still an open problem whether there is an algorithm that is polynomial in the output [5]. We can show that this construction always generates an Armstrong relation whose number of tuples is at most quadratic in that of an Armstrong relation that requires a minimum number of tuples.

Corollary 4. *Armstrong relations that are at most quadratic in that of a minimum Armstrong relation can be generated for unary by arbitrary key sets.*

Proof (Sketch). One can show first that a given relation is Armstrong for a given set of key sets if and only if for every anti-key the relation has two tuples which have matching values on exactly those attributes that form the anti-key and for no union over the elements of a key set there is a pair of tuples with matching values on all attributes in the union. Subsequently, one can show that the number of tuples in a minimum-sized Armstrong relation is bounded from below by one half of the square root of 1 plus 8 times the number of anti-keys, and bounded upwards by the increment of the number of anti-keys. Consequently, our construction generates an Armstrong relation that is at most quadratic in a minimum-sized Armstrong relation. □

Our construction can also be viewed as a construction of Armstrong relations for certain keys by key sets. Note that [8] constructed Armstrong relations for sets of possible and certain keys under `NOT NULL` constraints, whenever they exist. Our construction here does not require null markers.

Example 5. Consider the set $\Sigma = \{\mathcal{X}_1, \mathcal{X}_2\}$ with $\mathcal{X}_1$ and $\mathcal{X}_2$ from Example 2 over the relation schema WARD. Then $\bigcup \mathcal{X}_1 = \{room, time, injury\}$ and $\bigcup \mathcal{X}_2 = \{name, time, injury\}$. The minimum transversals would be $\mathcal{T}_1 = \{time\}$, $\mathcal{T}_2 = \{injury\}$, and $\mathcal{T}_3 = \{room, name\}$, and their complements on WARD are the anti-keys $\mathcal{A}_1 = \{room, name, address, injury\}$, $\mathcal{A}_2 = \{room, name, address, time\}$, and $\mathcal{A}_3 = \{address, injury, time\}$. The following relation is Armstrong for Σ.

room	*name*	*address*	*injury*	*time*
1	Miller	24 Queen St	leg fracture	Sunday, 16
1	Miller	24 Queen St	leg fracture	Monday, 19
1	Miller	24 Queen St	arm fracture	Monday, 19
2	Maier	24 Queen St	arm fracture	Monday, 19

The relation satisfies $\mathcal{X}_1$ and $\mathcal{X}_2$, but the relation violates the unary key set $\varphi' = \{\{room\}, \{name\}, \{address\}, \{time\}\}$, so φ' is not implied by Σ.

9 Conclusion and Future Work

We took first steps in investigating limits and opportunities for automated reasoning about key sets in databases. Key sets provide a more general and flexible implementation of entity integrity than Codd's notion of a primary key. We showed that the implication problem for general key sets enjoys a binary axiomatization, is *coNP*-complete, and lacks Armstrong relations. The implication problem of unary key sets by arbitrary key sets enjoys a unary axiomatization, is decidable in quadratic input time, and Armstrong relations can always be generated using hypergraph transversals such that the number of tuples is guaranteed to be at most quadratic in the minimum number of tuples required.

Interesting questions arise in theory and practice. Our *coNP*-completeness result calls for fixed-parameter solutions. A characterization for the existence of Armstrong relations in the general case would be interesting, and their efficient construction whenever possible. The validation of key sets in databases is an important practical issue, for which effective index structures need to be found. The problem of computing all key sets that hold in a given relation is important for data profiling [1]. Automated reasoning about foreign key sets is interesting as they generalize referential integrity [11]. Similar to how functional and inclusion dependencies and independence atoms interact [3, 9], automated reasoning for functional, multivalued, and inclusion dependency sets is interesting [6].

References

1. Abedjan, Z., Golab, L., Naumann, F.: Profiling relational data: a survey. VLDB J. 24(4), 557–581 (2015)
2. Bläsius, T., Friedrich, T., Schirneck, M.: The parameterized complexity of dependency detection in relational databases. In: IPEC 2016, August 24-26, 2016. pp. 6:1–6:13 (2016)
3. Casanova, M.A., Fagin, R., Papadimitriou, C.H.: Inclusion dependencies and their interaction with functional dependencies. J. Comput. Syst. Sci. 28(1), 29–59 (1984)
4. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM 13(6), 377–387 (Jun 1970)
5. Eiter, T., Gottlob, G., Makino, K.: New results on monotone dualization and generating hypergraph transversals. SIAM J. Comput. 32(2), 514–537 (2003)
6. Hannula, M., Kontinen, J., Link, S.: On the finite and general implication problems of independence atoms and keys. J. Comput. Syst. Sci. 82(5), 856–877 (2016)
7. Hartmann, S., Leck, U., Link, S.: On Codd families of keys over incomplete relations. Comput. J. 54(7), 1166–1180 (2011)
8. Köhler, H., Leck, U., Link, S., Zhou, X.: Possible and certain keys for SQL. VLDB J. 25(4), 571–596 (2016)
9. Köhler, H., Link, S.: Inclusion dependencies and their interaction with functional dependencies in SQL. J. Comput. Syst. Sci. 85, 104–131 (2017)
10. Langeveldt, W., Link, S.: Empirical evidence for the usefulness of Armstrong relations in the acquisition of meaningful functional dependencies. Inf. Syst. 35(3), 352–374 (2010)
11. Levene, M., Loizou, G.: A generalisation of entity and referential integrity in relational databases. ITA 35(2), 113–127 (2001)
12. Lucchesi, C.L., Osborn, S.L.: Candidate keys for relations. J. Comput. Syst. Sci. 17(2), 270–279 (1978)
13. Thalheim, B.: Dependencies in relational databases. Teubner (1991)
14. Thalheim, B.: On semantic issues connected with keys in relational databases permitting null values. Elektronische Informationsverarbeitung und Kybernetik 25(1/2), 11–20 (1989)
15. Thalheim, B.: The number of keys in relational and nested relational databases. Discrete Applied Mathematics 40(2), 265–282 (1992)
16. Wei, Z., Link, S., Liu, J.: Contextual keys. In: Mayr, H.C., Guizzardi, G., Ma, H., Pastor, O. (eds.) Conceptual Modeling - 36th International Conference, ER 2017, Valencia, Spain, November 6-9, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10650, pp. 266–279. Springer (2017)