The VLDB Journal manuscript No.
(will be inserted by the editor)

Possible and Certain Keys for SQL

Henning Köhler · Uwe Leck · Sebastian Link · Xiaofang Zhou

Received: date / Accepted: date

Abstract Driven by the dominance of the relational model and the requirements of modern applications, we revisit the fundamental notion of a key in relational databases with NULL. In SQL, primary key columns are NOT NULL, and UNIQUE constraints guarantee uniqueness only for tuples without NULL. We investigate the notions of possible and certain keys, which are keys that hold in some or all possible worlds that originate from an SQL table, respectively. Possible keys coincide with UNIQUE, thus providing a semantics for their syntactic definition in the SQL standard. Certain keys extend primary keys to include NULL columns, and can uniquely identify entities whenever feasible, while primary keys may not. In addition to basic characterization, axiomatization, discovery, and extremal combinatorics problems, we investigate the existence and construction of Armstrong tables, and describe an indexing scheme for enforcing certain keys. Our experiments show that certain keys with NULLs occur in real-world data, and related computational problems can be solved efficiently. Certain keys are therefore semantically well-founded and able to meet Codd's entity integrity rule while handling high volumes of incomplete data from different formats.

Keywords Armstrong database · Axiomatization · Data profiling · Discovery · Extremal combinatorics · Implication problem · Index · Key · Null marker · SQL

H. Köhler
School of Engineering & Advanced Technology, Massey University, Palmerston North, New Zealand E-mail: h.koehler@massey.ac.nz

U. Leck
Department of Mathematics, The University of Flensburg, Germany
E-mail: uwe.leck@uni-flensburg.de

S. Link
Department of Computer Science, The University of Auckland, Auckland, New Zealand
E-mail: s.link@auckland.ac.nz

X. Zhou
The University of Queensland, Brisbane, Australia; and Soochow University, Suzhou, China
E-mail: zfx@itee.uq.edu.au

1 Introduction

Keys have always been a core enabler for data management. They are fundamental for understanding the structure and semantics of data. Given a collection of entities, a key is a set of attributes 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 essential for many other data models, including semantic models, object models, XML and RDF. They are fundamental in many classical areas of data management, including data modeling, database design, indexing, and query optimization. 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 such as data partitioning or the creation of indexes and materialized views, and vi) provide new insights into application data. Modern applications raise the importance of keys further. They can facilitate the data integration process, help with the detection of duplicates

and anomalies, provide guidance in repairing and cleaning data, and return consistent answers to queries over dirty data. The discovery of keys from data is one of the core activities in data profiling.

According to Gartner, the NoSQL market will be worth 3.5 billion US dollars annually by 2018, but that for relational database technology will be worth 40 billion US dollars annually [12]. This underlines that "relational databases are here to stay, and that there is no substitute for important data" [12]. For these reasons relational databases must meet basic requirements of modern applications, inclusive of big data. On the one hand, relational technology must handle increases in the volume, variety and velocity of data. On the other hand, veracity and quality of the data must still be enforced, to support data-driven decision making. As a consequence, it is imperative that relational databases can acquire as much data as possible without sacrificing reasonable levels of data quality. Nulls constitute a convenient relational tool to accommodate high volumes of incomplete data from different formats. Unfortunately, nulls oppose the current golden standard for data quality in terms of keys, that is, Codd's rule of entity integrity. Entity integrity states that every table must have a primary key and that the columns which form the primary key must be unique and not null [10]. The goal of entity integrity is to ensure that real-world entities can be identified efficiently in database tables. In SQL, entity integrity is enforced by adding a primary key clause to a schema definition. `UNIQUE` constraints cannot always uniquely identify entities in which null markers occur in the columns involved. Nevertheless, even `UNIQUE` can provide a great deal of conditional logic to maintain patterns. These patterns are portable across relational databases: The indexes they create can be used by the optimizer to improve performance. But the real advantage is that the `UNIQUE` constraint can simply be evoked to enforce uniqueness for tuples without NULL. This eliminates the need for procedural code in the database and the application layers, and therefore a potential source of error. SQL systems enforce entity integrity by not allowing operations, that is inserts and updates, that produce an invalid primary key. Operations that create a duplicate primary key or one containing nulls are rejected. Hence, relational databases exhibit a trade-off between their main mechanism to accommodate the modern application requirements and their main mechanism to guarantee entity integrity.

We illustrate this trade-off by the following example. Consider the snapshot I of the `literature_references` table in the RFAM (RNA families) data set in Table 1. The table violates entity integrity as every potential primary key over the schema is violated by I. In particular, column *journal* carries the null marker $\perp$. Nevertheless, every tuple in I can be uniquely identified by a combination of column pairs, such as (title, journal) or (author, journal)[1]. Hence, the syntactic requirements of primary keys prohibit the entry of some data without good reason. The inability to insert important data into the database may force organizations to abandon key validation, exposing their future data to less quality, inefficiencies in data processing, waste of resources, and poor data-driven decision making. Finding a more liberal notion of keys will bring forward a new technology that accommodates the requirements of modern applications better than primary keys do.

title	*author*	*journal*
The uRNA database	Zwieb C	Nucleic Acids 1997
The uRNA database	Zwieb C	Nucleic Acids 1996
Genome wide detect.	Ragh. R	$\perp$

Table 1 Snippet I of the `literature_references` table

These observations have motivated us to introduce SQL keys with a well-founded semantics. For this purpose, we interpret occurrences of $\perp$ as *no information* [36]. We use possible worlds that result from an SQL table by replacing independently each occurrence of $\perp$ by a value from the corresponding domain, and where missing information is represented by the distinguished domain 'value' N/A for *not applicable*. Our data model therefore accommodates unknown and missing information, and is the first to associate a possible world semantics with the *no information* interpretation. In each possible world, occurrences of N/A are handled just as every other domain value. In particular, the equality relation between two domain values extends to N/A. That is, N/A equals the domain value v if and only if $v = N/A$. This makes perfect sense as every unknown information represented by $\perp$ has already been replaced by an actual domain value, different from N/A, in every possible world. Nevertheless, all our results also hold if we interpreted two different occurrences of N/A in a possible world as unequal. In conclusion, each possible world can be handled as a relation of total tuples in which duplicate tuples may occur. As usual, a possible world w satisfies a key X if and only if there are no two tuples in w that have distinct tuple identities and matching values on all the attributes in X. For example, the possible world w_1 of I in Table 2 satisfies keys such as $\{journal\}$, $\{author,journal\}$ and $\{title,journal\}$, and the possible world w_2 of I in Table 3 satisfies the keys $\{author,journal\}$ and $\{title,journal\}$ but violates the key $\{journal\}$. In particular, world w_1 models the

[1] The similar *journal* values of the first and second row are not different by mistake: Our keys deal with the integrity dimension of data, and not the accuracy dimension.

title	*author*	*journal*
The uRNA database	Zwieb C	Nucleic Acids 1997
The uRNA database	Zwieb C	Nucleic Acids 1996
Genome wide detect.	Ragh. R	N/A

Table 2 Possible World w_1 of snippet I in Table 1

title	*author*	*journal*
The uRNA database	Zwieb C	Nucleic Acids 1997
The uRNA database	Zwieb C	Nucleic Acids 1996
Genome wide detect.	Ragh. R	Nucleic Acids 1997

Table 3 Possible World w_2 of snippet I in Table 1

case in which the paper *Genom wide detect.* by *Ragh. R* has not been published in a journal. When required we can also stipulate minimality requirements on keys.

Our approach naturally suggests two semantics of SQL keys. A *possible key* $p\langle X\rangle$ is satisfied by an SQL table I if there is some possible world of I in which the key X is satisfied. A *certain key* $c\langle X\rangle$ is satisfied by an SQL table I if the key X is satisfied in every possible world of I. In particular, certain keys do not prevent the entry of incomplete tuples that can still be identified uniquely, independently of which information $\perp$ occurrences represent. For example, the snapshot I in Table 1 satisfies the possible key $p\langle journal\rangle$ as witnessed by world w_1 in Table 2, but violates the certain key $c\langle journal\rangle$ as witnessed by world w_2 in Table 3. Moreover, I satisfies the certain keys $c\langle title, journal\rangle$ and $c\langle author, journal\rangle$. The example illustrates that primary keys are only sufficient to uniquely identify tuples in SQL tables, while certain keys are also necessary. That is, primary keys uniquely identify all tuples when none of them feature $\perp$ in all their key columns. However, there are SQL tables in which it is not necessary to disallow null marker occurrences in key columns in order to identify every tuple. The impact areas of keys and uniqueness constraints, as mentioned before, apply to certain and possible keys. It is therefore important to investigate basic problems about possible and certain keys, ranging from their semantic definition and syntactic characterization, to the computational complexity of their associated implication problem, the ability to discover them from given SQL tables and to recognize keys that are meaningful in a given application domain, the description of the largest families of non-redundant keys, and the identification of index designs that help validate keys efficiently. Our contributions can be summarized as follows:

1. We propose the first possible world semantics for keys over SQL tables under the most basic interpretation of null marker occurrences $\perp$ as *no information*. Here, replacements of $\perp$ by Codd's null marker N/A for *inapplicable* mean that this occurrence of $\perp$ represents missing information, while replacements of $\perp$ by actual domain values mean that this occurrence of $\perp$ represents unknown information. Keys are possible when they hold in some possible world, while keys are certain when they hold in all possible worlds.

2. We establish simple syntactic characterizations to validate the satisfaction of possible and certain keys directly on the given SQL table. Permitting possible worlds to be multisets means that possible keys provide a semantics for `UNIQUE` constraints.

3. We characterize the implication problem for the combination of possible and certain keys and `NOT NULL` constraints axiomatically, and by an algorithm that works in linear time in the input. We can therefore efficiently compute *the* minimal cover of our constraints to reduce the amount of integrity maintenance to a minimal level necessary. Our algorithm requires only little time to eliminate all future redundant enforcements of keys, which saves more time when data sets become bigger.

4. We address the data-driven discovery of possible and certain keys. Exploiting hypergraph transversals, we establish a compact algorithm to compute a cover for the possible and certain keys that hold on a given table. We applied the algorithm to real-world data, showing that possible and certain keys occur frequently in practice.

5. We complement the data-driven discovery of possible and certain keys with a schema-driven approach towards their discovery. For this purpose, we investigate structural and computational aspects of Armstrong tables. Given some set of possible keys, certain keys and `NOT NULL` constraints, an Armstrong table for this set is an SQL table that satisfies the constraints in this set but violates all possible keys, certain keys and `NOT NULL` constraints that are not implied by the set. For example, snapshot I of Table 1 is an Armstrong table for $p\langle journal\rangle$, $c\langle title, journal\rangle$, $c\langle author, journal\rangle$ and the `NOT NULL` constraints on *title* and *author*. Despite much more involved challenges as encountered in the idealized special case of relations, we characterize when Armstrong tables exist and how to compute them in these cases. While being, in theory, worst-case double exponential in the input, experiments show that our algorithm produces tables with less than ten rows within a few milliseconds on average.

6. Meaningful keys rarely consist of attributes that all have a small number of domain values. Nevertheless, we investigate possible and certain keys under a small domain requirement, such as `boolean`. While our syntactic characterization of certain keys carries over, we show that deciding the satisfaction of possible key sets becomes NP-hard. Firstly, this means the notion of certain keys is robust. Secondly, the NP-hardness result prompts us to adopt - for finite domains - the syntac-

tic notion of UNIQUE instead of the semantic notion of a possible key. This ensures that our contributions to reasoning and discovery are retained for finite domains. Finally, our results also justify the syntactic notion of UNIQUE from a computational point of view. In addition, we show how to construct Armstrong tables for whenever every certain key contains some NOT NULL attribute with an infinite domain and every possible key contains some attribute with an infinite domain.

7. For a database designer it is a natural question to ask how large non-redundant families of integrity constraints can potentially be. Answers to this question provide the designer with upper bounds on how complex integrity maintenance can become. This may result in the requirement to restrict the size of keys. One may then ask how large non-redundant families for keys of restricted size can become. Using extremal set theory we identify the non-redundant families of possible keys and certain keys that attain maximum cardinality under given NOT NULL constraints, even when we limit the keys to those that respect an upper bound on the number of attributes. In other words, our results characterize minimal covers of maximum cardinality.

8. We propose an indexing scheme for certain keys. In comparison to enforcing keys on a data set with 100 million tuples and without indices, our scheme improves the enforcement of certain keys on inserts by a factor of 10^4. It works only marginally slower than the enforcement of primary keys, provided that the certain keys have only a small number of columns in which null markers can occur. Exploiting our data-driven discovery algorithm from before, we have found only certain keys in which at most two columns feature null markers.

Our findings show how certain keys attain the golden standard of Codd's principle of entity integrity under the requirements of modern applications.

Organization. Section 2 discusses related work. Possible and certain keys are introduced in Section 3 where we also establish their syntactic characterization and solutions to their implication and discovery problems. Structural and computational aspects of Armstrong tables are investigated in Section 4. All these results are reviewed under a small domain assumption in Section 5. Extremal problems are studied in Section 6. An efficient indexing scheme for the enforcement of certain keys is established in Section 7. Results of our experiments are presented in Section 8. We conclude in Section 9. Section 5 uses material from Section 4, which is based on Section 3. Section 6 can be read after Section 3. Section 7 can be read after Section 3. Finally, Section 8 refers to different results from Sections 3, 4, and 7.

2 Related Work

Integrity constraints enforce the semantics of application domains in database systems. They form a cornerstone of database technology [2]. Entity integrity is one of the three inherent integrity rules proposed by Codd [10]. Keys and foreign keys are the only ones amongst around 100 classes of constraints [2] that enjoy built-in support by SQL database systems. In particular, entity integrity is enforced by primary keys [40]. Core problems investigate reasoning [19,20,24], Armstrong databases [21], and discovery [25,37,38,49]. Applications include consistent query answers [33], data cleaning [18], exchange [17], fusion [45], integration [7], profiling [43], quality [47], and security [5], schema design [15], query optimization [26], and view maintenance [46].

Surrogate keys ('auto-increment' fields) are often seen as a practical solution to enforce entity integrity. Frequent arguments are performance improvements and robustness under schema changes. In reality, semantic keys also contribute to that [3]. We illustrate how surrogate keys can cause entity duplication. Table 4 shows a real-world sample S of the ncbi_taxonomy table from the PFAM (protein families) data set. Here, *ncbi_taxid* is a surrogate primary key.

The first two rows of S illustrate that some entities, a combination of a species and taxonomy, were duplicated despite the presence of a surrogate key. If, instead, the meaningful certain key $c\langle species, taxonomy\rangle$ had been enforced, this violation of entity integrity would have been avoided. Such cases are not an exception. In New Zealand, hospital patients are identified by their National Health Index (NHI) number. In a 12-month duplicate-resolution programme in 2003, over 125,000 duplicate NHI numbers were resolved [44]. This means there are people with more than one NHI number. Unavailability of all NHI numbers for the same patient means that the health history is incomplete, which may lead to fatal incorrect decisions. Nevertheless, keys do not aim at solving the duplicate detection problem. Intuitively, the violation of meaningful keys should cause only a small fraction of tuples to duplicate entities.

One of the most important extensions of the relational model [10] is incomplete information, due to the high demand for the correct handling of such information in applications. A landmark paper for extending algebra operators to incomplete information is [27]. The two most prolific proposals for null marker interpretations are "value unknown at present" [10] and "no information" [36]. The former has a well-founded possible world semantics but missing information is not covered; and while the latter can express unknown and

ncbi_taxid	*species*	*taxonomy*
39378	Catenula sp.	Eukaryota;Metazoa;Platyhelminthes;Turbellaria;Catenulida;Catenulidae;Catenula;
66404	Catenula sp.	Eukaryota;Metazoa;Platyhelminthes;Turbellaria;Catenulida;Catenulidae;Catenula;
131567	cellular organisms	$\perp$

Table 4 Snippet S of the `ncbi_taxonomy` data set

missing information, previous research did not associate a possible world semantics with it. To the best of our knowledge, our research is the first to combine the best of both: SQL tables can feature "no information" nulls to express unknown and missing information, and we obtain possible worlds from SQL tables by replacing $\perp$ with N/A to represents missing information, and replacing $\perp$ with actual domain values to represent unknown information.

Our data model therefore constitutes a principled general approach to the semantics of constraints. Levene and Loizou introduced strong and weak functional dependencies (FDs) [34]. A weak FD holds on some possible world, while a strong FD holds on every possible world. Hence, strong/weak FDs and possible/certain keys are closely related. However, neither are certain keys special cases of strong FDs, nor are possible keys special cases of weak FDs. The reason is that we permit duplicate tuples in possible worlds, which is necessary to obtain entity integrity. Consider I_1 in Table 5: If we prohibited duplicates, *title* would become a certain key even though we have "no information" on *title* for the second tuple in I_1, which makes it impossible to distinguish the first and second tuple in I_1 based on *title*. Under multiset semantics, keys are no longer special cases of FDs, since the former prohibit duplicate tuples and the latter do not. Duplicates occur naturally in applications, such as data integration, and should be accommodated by the semantics of keys. For example, the snippet I_1 in Table 5 satisfies the strong FD $title \rightarrow author$, i.e., $\{title\}$ is a strong key for I_1, but $c\langle title\rangle$ does not hold on I_1. In fact, replacing $\perp$ in I_1 with 'uRNA' results in duplicates that violate the key $\{title\}$. Similarly, the snippet I_2 in Table 5 satisfies the weak FD $author \rightarrow journal$, i.e., $\{author\}$ is a weak key for I_2, but $p\langle author\rangle$ does not hold on I_2. In fact, replacing $\perp$ in I_2 with 'Acids' results in duplicates that violate the key $\{author\}$. The snippet I_2 further illustrates the benefit of duplicates in possible worlds. This guarantees that `UNIQUE` coincides with possible keys, but not with weak keys: I_2 violates `UNIQUE`$(author)$ and $p\langle author\rangle$, but satisfies the weak key $\{author\}$.

The focus in [34] is on axiomatization and implication of strong/weak FDs. While Armstrong tables were claimed to always exist [34], we discovered a technical error in the proof. Section 4.5 presents a set of strong and weak FDs for which no Armstrong table exists.

I_1

title	*author*
uRNA	Zwieb
$\perp$	Zwieb

I_2

author	*journal*
Zwieb C	$\perp$
Zwieb C	Acids

Table 5 SQL tables I_1 and I_2

The principle of entity integrity has been challenged before [51,35], both following the notion of a key set. A relation satisfies a key set if, for each pair of distinct tuples, there is some key in the key set on which the two tuples are total and distinct. A certain key is equivalent to a key set consisting of all the singleton subsets of the key attributes, e.g., $c\langle title, journal\rangle$ corresponds to the key set $\{\{title\}, \{journal\}\}$. However, our work is different in that we study the interaction with possible keys and `NOT NULL` attributes, establish a possible world semantics, and study different problems. The implication problem for the sole class of primary keys is examined in [22]. As the key columns of primary keys are `NOT NULL`, the class of primary keys behaves differently from both possible and certain keys.

Our findings may be important for other data models such as XML [23] and RDF [8] where incomplete information is inherent, probabilistic databases [6,28] and also description logics [8].

Some of our results were announced in [32]. The current paper contains proofs of all results. These are fundamental for understanding our techniques and often contain useful constructions. We also showed how to extend our results from [32] from only unknown to both unknown and missing information. Section 5 is new and establishes results for finite domains. Section 6 is also new and investigates extremal set problems. Finally, we present additional examples and experiments that further motivate our research and illustrate our findings.

Summary. Certain keys are a natural approach to identify entities in SQL tables. Surprisingly, they have not received more attention yet. The combination of certain and possible keys under `NOT NULL` constraints is particularly relevant to SQL. The presence of certain keys means that the problems studied here are substantially different from previous work.

3 Possible and Certain Keys

After some preliminaries we introduce possible and certain keys. Subsequently, we characterize these notions syntactically, and derive a simple axiomatic and linear time algorithmic characterization of their implication problem. We show that possible and certain keys enjoy a unique minimal representation. Finally, we exploit hypergraph transversals to discover possible and certain keys from a given table.

3.1 Preliminaries

We begin with basic terminology. Let $\mathfrak{A} = \{A_1, A_2, \ldots\}$ be a (countably) infinite set of distinct symbols, called *attributes*. Attributes represent column names of tables. A *table schema* is a finite non-empty subset T of $\mathfrak{A}$. Each attribute A is associated with an infinite domain $dom(A)$ which represents the possible values that can occur in column A. We assume that each domain $dom(A)$ contains the symbol 'N/A', which is short for *not applicable*. We include 'N/A' in each domain out of convenience, and stress that 'N/A' is not an actual value, but rather a marker that represents missing information in the form of Codd's null marker *inapplicable*. The marker 'N/A' will not feature in SQL tables, but only in their possible worlds. In order to encompass missing and unknown information in SQL tables, the domain of each attribute also contains the *no information* null marker, denoted by $\perp$ [36]. As with 'N/A', $\perp$ is not a value, but we include $\perp$ in attribute domains as a convenience. In particular, the equality relation over the domain of each attribute extends to 'N/A' and $\perp$ as for every other domain value. We define that 'N/A'='N/A' as it does not seem feasible to stipulate two occurrences of 'N/A' as unequal. Nevertheless, all our results also hold if we define 'N/A'≠'N/A'. Thus the semantics of possible and certain keys is independent of whether we stipulate equality or inequality between every two different occurrences of 'N/A' in possible worlds.

For attribute sets X and Y we may write XY for their set union $X \cup Y$. If $X = \{A_1, \ldots, A_m\}$, then we may write $A_1 \cdots A_m$ for X. A *tuple* over T is a function $t : T \to \bigcup_{A \in T} dom(A)$ with $t(A) \in dom(A)$ for all $A \in X$. For $X \subseteq T$ let $t[X]$ denote the restriction of the tuple t over T to X. We say that a tuple t is *X-total* if $t[A] \neq \perp$ for all $A \in X$. A tuple t over T is said to be a *total tuple* if it is T-total. A *table* I over T is a finite multiset of tuples over T. As SQL only features one designated null marker that encompasses missing and unknown information, we only allow occurrences of $\perp$ in SQL tables, but not occurrences of 'N/A'. A table I over T is *(X-)total* if every tuple $t \in I$ is (X-)total. Let t, t' be tuples over T. We define *weak/strong similarity* of t, t' on $X \subseteq T$ as follows:

$$t[X] \sim_w t'[X] :\Leftrightarrow \forall A \in X. \; (t[A] = t'[A] \vee t[A] = \perp \vee t'[A] = \perp)$$
$$t[X] \sim_s t'[X] :\Leftrightarrow \forall A \in X. \; (t[A] = t'[A] \neq \perp)$$

Weak and strong similarity become identical for tuples that are X-total. In such "classical" cases we denote similarity by $t[X] \sim t'[X]$. We will use the phrase t, t' *agree* interchangeably for t, t' *are similar*.
A *null-free subschema* (NFS) over T is a set T_S where $T_S \subseteq T$. The NFS T_S over T is satisfied by a table if it is T_S-total. SQL attributes can be declared `NOT NULL`, so the set of these attributes forms an NFS. We may refer to the pair (T, T_S) as table schema.

We say that $X \subseteq T$ is a *key* for the total table I, denoted by $I \vdash X$, if there are no two tuples $t, t' \in I$ that have distinct tuple identities and agree on X. Given a table I, a *possible world* of I is obtained by independently replacing every occurrence of $\perp$ in I with a domain value different from $\perp$. If $\perp$ represent missing information, then it is replaced by 'N/A'. Otherwise, it represents unknown information and is replaced by some actual domain value. We say that $X \subseteq T$ is a *possible/certain key* for I, denoted by $p\langle X \rangle$ and $c\langle X \rangle$ respectively, if the following hold:

$$I \vdash p\langle X \rangle :\Leftrightarrow X \text{ is a key for some possible world of } I$$
$$I \vdash c\langle X \rangle :\Leftrightarrow X \text{ is a key for every possible world of } I .$$

We illustrate this semantics on our running example.

Example 1 For $T = \{\text{title}, \text{author}, \text{journal}\}$ and $T_S = \{\text{title}, \text{author}\}$ let I denote the instance from Table 1. Then $I \vdash p\langle journal \rangle$ as $\perp$ can be replaced by a domain value different from the journals in I. Furthermore, $I \vdash c\langle title, journal \rangle$ as the first two rows are unique on journal, and the last row is unique on title. Finally, $I \vdash c\langle author, journal \rangle$ as the first two rows are unique on journal, and the last row is unique on author. □

For a set Σ of constraints, such as possible and certain keys, table I over T *satisfies* Σ if I satisfies every $\sigma \in \Sigma$. If for some $\sigma \in \Sigma$, I does not satisfy σ we say that I *violates* σ (and violates Σ). A table I over (T, T_S) is a table I over T that satisfies T_S. A table I over (T, T_S, Σ) is a table I over (T, T_S) that satisfies Σ.

When discussing possible and certain keys, the following notions of strong and weak anti-keys are useful. Let I be a table over (T, T_S) and $X \subseteq T$. We say that X is a *strong/weak anti-key* for I, denoted by $\neg_p\langle X \rangle$ and $\neg_c\langle X \rangle$ respectively, if $p\langle X \rangle$ and $c\langle X \rangle$, respectively, do *not* hold on I. We may also say that $\neg_p\langle X \rangle$ and/or

$\neg_c\langle X\rangle$ *hold* on I. We write $\neg\langle X\rangle$ to denote an anti-key which may be either strong or weak. A set Σ of constraints over (T, T_S) *permits* a set Π of strong and weak anti-keys if there is a table I over (T, T_S, Σ) such that every anti-key in Π holds on I. Permitting every single anti-key in a set is not the same as permitting the set of anti-keys as a whole. Checking whether Σ permits a single anti-key can be done easily using Theorem 2, while Proposition 4 shows that deciding whether Σ permits a set of anti-keys is NP-complete. We illustrate the semantics on our running example.

Example 2 Let T, T_S, and I be as in Example 1. Then $I \vdash \neg_p\langle title, author\rangle$ as the first two rows will agree on title and author in every possible world. Furthermore, $I \vdash \neg_c\langle journal\rangle$ as $\perp$ could be replaced by either of the two journals listed in I, resulting in possible worlds that violate the key {journal}. □

3.2 Syntactic Characterization

While possible worlds provide a well-founded semantics to possible and certain keys, it is infeasible to explore infinitely many possible worlds to validate if a given possible or certain key holds on a given SQL table. The following result characterizes the semantics of possible and certain keys syntactically. This means we can directly validate our keys on the given SQL table.

Our characterization shows, in particular, that possible keys capture the `UNIQUE` constraint in SQL. Therefore, we have established a first formal semantics for this constraint. Moreover, Theorem 1 provides a foundation for developing efficient algorithms that effectively exploit our keys in data processing.

Theorem 1 *$X \subseteq T$ is a possible (certain) key for I iff no two tuples in I with distinct tuple identities are strongly (weakly) similar on X.*

Proof (possible key ⇒ not strongly similar) Let X be a possible key for I and $t, t' \in I$ have distinct tuple identities. Then there is a possible word $\rho(I)$ of I for which X is a key. Denote by $\rho(t), \rho(t')$ the copies of t, t' in $\rho(I)$. Since X is a key for $\rho(I)$, there is some $A \in X$ with $\rho(t)[A] \neq \rho(t')[A]$. Since only $\perp$ values get replaced, $t[A] \neq t'[A]$ or $t[A] = \perp$ or $t'[A] = \perp$ holds. In either case t, t' are not strongly similar.

(possible key ⇐ not strongly similar) Let no two tuples with distinct identity in I be strongly similar on X. We construct a possible world $\rho(I)$ of I by replacing every $\perp$ occurrence in I with a distinct domain value that did not occur in I previously. Such a replacement exists since we assume, in line with previous work, that domains are infinite and tables are finite. Let again $\rho(t), \rho(t') \in \rho(I)$ be two distinct copies of $t, t' \in I$. Since t, t' are not strongly similar on X, there exists $A \in X$ with $t[A] \neq t'[A]$ or $t[A] = \perp$ or $t'[A] = \perp$. As our replacement values are unique in $\rho(I)$, we have $\rho(t)[A] \neq \rho(t')[A]$ in all three cases. Thus $\rho(t), \rho(t')$ are not similar on X, so X is a possible key for I.

(not certain key ⇒ weakly similar) Let X not be a certain key for I. Then there exists a possible world $\rho(I)$ of I and tuples $t, t' \in I$ with distinct identities such that $\rho(t)[X] \sim \rho(t')[X]$. Thus for every $A \in X$ we have $\rho(t)[A] = \rho(t')[A]$ and hence $t[A] = t'[A]$ or $t[A] = \perp$ or $t'[A] = \perp$, i.e., $t[X] \sim_w t'[X]$.

(not certain key ⇐ weakly similar) Let $t, t' \in I$ with $t[X] \sim_w t'[X]$. Then we can construct a possible world $\rho(I)$ of I which replaces $\perp$ values on t, t' as follows:

- If $t[A] = t'[A] = \perp$ let $\rho(t)[A] = \rho(t')[A]$ be arbitrary.
- If $t[A] = \perp \wedge t'[A] \neq \perp$ let $\rho(t)[A] = t'[A]$.
- If $t[A] \neq \perp \wedge t'[A] = \perp$ let $\rho(t')[A] = t[A]$.

In each case, $\rho(t)[A] = \rho(t')[A]$ and thus $\rho(t)[X] \sim \rho(t')[X]$. Hence, X is not a certain key for I. □

We illustrate the syntactic characterization of our key semantics from Theorem 1 on our running example.

Example 3 Let T, T_S, and I be as in Example 1. Then $I \vdash p\langle journal\rangle$ as no two I-tuples with different tuple identities strongly agree on journal. Furthermore, $I \vdash c\langle title, journal\rangle$ as no two I-tuples with different tuple identity weakly agree on title and journal. Finally, $I \vdash c\langle author, journal\rangle$ as no two I-tuples with different tuple identity weakly agree on author and journal. □

Theorem 1 implies that the satisfaction of keys can also be characterized with SQL's three-valued logic, where $\perp = v$ evaluates to `unknown` [9]. Given that, for a certain key the default key criterion (i.e. pairwise comparison of all tuples) always evaluates to `true`, for possible keys to `unknown`, and for non-keys to `false`.

3.3 Implication

Many data management tasks benefit from the ability to decide the implication problem of semantic constraints. In our context, the implication problem can be defined as follows. Let (T, T_S) denote the schema under consideration. For a set $\Sigma \cup \{\varphi\}$ of constraints over (T, T_S) we say that Σ *implies* φ, denoted by $\Sigma \models \varphi$, if and only if every table over (T, T_S) that satisfies Σ also satisfies φ. The *implication problem* for a class $\mathcal{C}$ of constraints is to decide, for an arbitrary (T, T_S) and an arbitrary set $\Sigma \cup \{\varphi\}$ of constraints in $\mathcal{C}$, whether Σ implies φ. For possible and certain keys the implication problem can be characterized as follows.

Theorem 2 *Let Σ be a set of possible and certain keys. Then i) Σ implies $c\langle X\rangle$ iff $c\langle Y\rangle \in \Sigma$ for some $Y \subseteq X$ or $p\langle Z\rangle \in \Sigma$ for some $Z \subseteq X \cap T_S$, and ii) Σ implies $p\langle X\rangle$ iff $c\langle Y\rangle \in \Sigma$ or $p\langle Y\rangle \in \Sigma$ for some $Y \subseteq X$.*

Proof The "if" directions ($\Leftarrow$) follow from Theorem 1. We show next the "only if" directions ($\Rightarrow$) by contraposition.

i) Let $c\langle Y\rangle \notin \Sigma$ for every $Y \subseteq X$ and $p\langle Z\rangle \notin \Sigma$ for every $Z \subseteq X \cap T_S$. We show that Σ does not imply $c\langle X\rangle$ by constructing a table I over (T, T_S, Σ) that violates $c\langle X\rangle$. Consider a table $I = \{t, t'\}$ over (T, T_S) with the following properties:
   - $t = (0, \ldots, 0)$
   - $t'[X \cap T_S] = (0, \ldots, 0)$
   - $t'[X \setminus T_S] = (\bot, \ldots, \bot)$
   - $t'[T \setminus X] = (1, \ldots, 1)$

   Now consider Theorem 1. Since t, t' weekly agree on X only, the only certain keys $c\langle Y\rangle$ violated by I are those with $Y \subseteq X$. Hence no certain keys in Σ are violated by I. Since t, t' strongly agree on $X \cap T_S$ only, the only possible keys $p\langle Z\rangle$ violated by I are those with $Z \subseteq X \cap T_S$. Hence no possible keys in Σ are violated by I. Hence I satisfies Σ but violates $c\langle X\rangle$, so Σ does not imply $c\langle X\rangle$.

ii) Analogous with $t'[X] = (0, \ldots, 0)$.

This concludes the proof. □

Thus, a certain key is implied by Σ iff it contains a certain key from Σ or its `NOT NULL` columns contain a possible key from Σ. Similarly, a possible key is implied by Σ iff it contains a possible or certain key from Σ. We exemplify Theorem 2 on our running example.

Example 4 Let (T, T_S) be as in Example 1 and define $\Sigma = \{c\langle title, journal\rangle, c\langle author, journal\rangle, p\langle journal\rangle\}$. By Theorem 2, Σ implies $c\langle title, author, journal\rangle$ and $p\langle title, journal\rangle$, but Σ implies neither $c\langle journal\rangle$ nor $p\langle title, author\rangle$. This is confirmed by Table 1, which satisfies $c\langle title, author, journal\rangle$ and $p\langle title, journal\rangle$, but violates $c\langle journal\rangle$ and $p\langle title, author\rangle$. □

A consequence of Theorem 2 is that the implication of our keys can be decided with just one scan over the input. The size of the input is defined as the total number of attribute occurrences in the input.

Corollary 1 *The implication problem for the class of possible and certain keys can be decided in time linear in the size of the input.* □

Corollary 1 indicates that the semantics of our keys can be exploited efficiently in many data management tasks. We illustrate this on the following example.

Example 5 Suppose we want to find all distinct combinations of authors and journals from the current instance over (T, T_S) from Example 1. As part of evaluating the query

```
SELECT DISTINCT author, journal
FROM T WHERE journal IS NOT NULL;
```

an optimizer, that can reason about our keys, may check if $c\langle author, journal\rangle$ is implied by the set Σ of specified constraints on (T, T_S) together with the additional `NOT NULL` constraint on *journal*, enforced by the `WHERE` clause. In this case, as for the set Σ in Example 4, the `DISTINCT` clause can be omitted, saving the expensive operation of duplicate removal. □

Similar techniques allow us to characterize the implication problem of strong and weak anti-keys as well. Proposition 1 is the dual version of Theorem 2, see the appendix for the proof.

Proposition 1 *Let Π be a set of strong and weak anti-keys. Then i) Π implies $\neg_p\langle X\rangle$ iff $\neg_p\langle Y\rangle \in \Pi$ for some $Y \supseteq X$, or $\neg_c\langle Z\rangle \in \Pi$ for some Z with $X \subseteq Z \cap T_S$, and ii) Π implies $\neg_c\langle X\rangle$ iff $\neg_c\langle Y\rangle \in \Pi$ or $\neg_p\langle Y\rangle \in \Pi$ for some $Y \supseteq X$.*

Example 6 Let (T, T_S) be as in Example 1, and define $\Pi = \{\neg_p\langle title, author\rangle, \neg_c\langle journal\rangle\}$. Then Proposition 1 shows that Π implies $\neg_p\langle author\rangle$ and $\neg_c\langle title\rangle$, but neither $\neg_p\langle journal\rangle$ nor $\neg_c\langle author, journal\rangle$. □

3.4 Axiomatization

We now establish an axiomatic characterization of the implication problem, which provides us with a tool to understand and reason about the interaction of possible keys, certain keys and `NOT NULL` constraints.

Let $\Sigma \cup \{\varphi\}$ denote a set of possible and certain keys over (T, T_S). Let $\Sigma^* = \{\varphi \mid \Sigma \models \varphi\}$ denote the *semantic closure* of Σ. In order to determine the semantic closure, one can utilize a syntactic approach by applying *inference rules* of the form $\frac{\text{premise}}{\text{conclusion}}$ condition, where rules without a premise are called *axioms*. 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 $\Sigma^+_{\mathfrak{R}} = \{\varphi \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$ denote the *syntactic closure* of Σ under inferences by $\mathfrak{R}$. The set $\mathfrak{R}$ is *sound* (*complete*) if for every table schema (T, T_S) and for every set Σ we have $\Sigma^+_{\mathfrak{R}} \subseteq \Sigma^*$ ($\Sigma^* \subseteq \Sigma^+_{\mathfrak{R}}$). The (finite) set $\mathfrak{R}$ is said to be a (finite) *axiomatization* if $\mathfrak{R}$ is both sound and complete.

Corollary 2 *The following axioms are sound and complete for the implication of possible keys, certain keys and* `NOT NULL` *constraints.*

$$\textit{p-Extension: } \frac{p\langle X\rangle}{p\langle XY\rangle} \quad \textit{c-Extension: } \frac{c\langle X\rangle}{c\langle XY\rangle}$$

$$\textit{Weakening: } \frac{c\langle X\rangle}{p\langle X\rangle} \quad \textit{Strengthening: } \frac{p\langle X\rangle}{c\langle X\rangle}\, X \subseteq T_S$$

Proof Soundness follows from Theorem 1. Indeed, tuples with distinct identities that are strongly (weakly) similar on XY are also strongly (weakly) similar on X, showing the soundness of the extension rules. Moreover, tuples with distinct identities that are strongly similar on X are also weakly similar on X, showing soundness of the weakening rule. Soundness of the strengthening rule follows from the fact that strong and weak similarity coincide on every attribute declared `NOT NULL`.

For showing completeness let $\Sigma \models c\langle X\rangle$. By Theorem 2 there exist either $c\langle Y\rangle \in \Sigma$ with $Y \subseteq X$ or $p\langle Z\rangle \in \Sigma$ with $Z \subseteq X \cap T_S$. In the former case $c\langle X\rangle$ can be derived via c-Extension, in the latter via Strengthening and c-Extension. Finally let $\Sigma \models p\langle X\rangle$. By Theorem 2 there exist $Y \subseteq X$ with $c\langle Y\rangle \in \Sigma$ or $p\langle Y\rangle \in \Sigma$. In the former case $p\langle X\rangle$ can be derived via Weakening and p-Extension, in the latter via p-Extension alone. □

A simple application of the inference rules is shown on our running example.

Example 7 Let (T, T_S, Σ) be given as in Example 4. Then $p\langle author, journal\rangle$ can be inferred from Σ by applying p-Extension to $p\langle journal\rangle$, or by applying Weakening to $c\langle author, journal\rangle$. □

The axiomatization from Corollary 2 provides us with means to effectively enumerate *all* keys implied by a given key set. The algorithmic solution from Theorem 2 provides us with means to decide for a given key whether it is implied by a given key set.

3.5 Minimal Covers

A *cover* of Σ is a set Σ' where every element is implied by Σ and which implies every element of Σ. Hence, a cover is just a representation. Minimal representations limit the validation of constraints to those necessary. People also find minimal representations easier to work with. For possible and certain keys, a unique minimal representation exists.

We say that i) $p\langle X\rangle \in \Sigma$ is *non-minimal* if $\Sigma \models p\langle Y\rangle$ for some $Y \subset X$ or $\Sigma \models c\langle X\rangle$, and ii) $c\langle X\rangle \in \Sigma$ is *non-minimal* if $\Sigma \models c\langle Y\rangle$ for some $Y \subset X$. We call a key σ with $\Sigma \models \sigma$ *minimal* iff σ is not non-minimal. We call $\sigma \in \Sigma$ *redundant* iff $\Sigma \setminus \{\sigma\} \models \sigma$, and *non-redundant* otherwise. We call Σ *minimal* (*non-redundant*) iff all keys in Σ are minimal (non-redundant), and *non-minimal* (*redundant*) otherwise.

Due to the logical equivalence of $p\langle X\rangle$ and $c\langle X\rangle$ for $X \subseteq T_S$, certain keys can be both minimal and redundant while possible keys can be both non-minimal and non-redundant. For example, given $T = A = T_S$ and $\Sigma = \{p\langle A\rangle, c\langle A\rangle\}$, $c\langle A\rangle$ is minimal because Σ does not imply $c\langle\emptyset\rangle$, and $c\langle A\rangle \in \Sigma$ is also redundant because $\{p\langle A\rangle\}$ implies $c\langle A\rangle$ since $A \in T_S$. Given $T = A = T_S$ and $\Sigma = \{p\langle A\rangle\}$, $p\langle A\rangle$ is non-minimal because $c\langle A\rangle$ is implied by Σ since $A \in T_S$, and $p\langle A\rangle$ is non-redundant because it is not implied by the set $\Sigma' = \Sigma \setminus \{p\langle A\rangle\}$.

Proposition 2 *The set Σ_{min} of all minimal possible and certain keys for Σ is a non-redundant cover of Σ. Indeed, Σ_{min} is the only minimal cover of Σ.*

Proof For every key $\sigma \in \Sigma$ there is a minimal key σ' with $\Sigma \models \sigma' \models \sigma$, found by application of the definition for minimal keys. Hence Σ_{min} is a cover of Σ.

Now assume $\sigma \in \Sigma_{min}$ is redundant. Then $\Sigma_{min} \setminus \{\sigma\} \models \sigma$, and by application of the definition for minimal keys, there must exist $\sigma' \in \Sigma_{min} \setminus \{\sigma\}$ with $\sigma' \models \sigma$. Since σ is minimal this can only happen for $\sigma = c\langle X\rangle, \sigma' = p\langle X\rangle$ for some $X \subseteq T_S$. But then σ' is not minimal, which contradicts $\sigma' \in \Sigma_{min}$. Hence Σ_{min} is non-redundant. □

We may hence talk about *the* minimal cover of Σ, and illustrate this concept on our running example.

Example 8 Let (T, T_S) be as in Example 1 and Σ' consist of $p\langle journal\rangle$, $p\langle author, journal\rangle$, $c\langle title, journal\rangle$, $c\langle author, journal\rangle$, and $c\langle title, author, journal\rangle$. Then $\Sigma'_{\min} = \Sigma$ from Example 4. □

A strong anti-key $\neg_p\langle X\rangle$ is *non-maximal* if $\neg\langle Y\rangle$ with $X \subset Y$ is an anti-key implying $\neg_p\langle X\rangle$ on (T, T_S). A weak anti-key $\neg_c\langle X\rangle$ is *non-maximal* if $\neg\langle Y\rangle$ with $X \subset Y$ is an anti-key or $\neg_p\langle X\rangle$ is a strong anti-key. Anti-keys are *maximal* unless they are non-maximal. We denote the set of maximal strong anti-keys by $\mathcal{A}^s_{max}$, the set of maximal weak anti-keys by $\mathcal{A}^w_{max}$ and their disjoint union by $\mathcal{A}_{max}$.

3.6 Key Discovery

Our next goal is to discover all certain and possible keys that hold in a given table I over (T, T_S). If T_S is not given it is trivial to find a maximum T_S. The discovery of constraints from data reveals semantic information useful for database administration and applications. Combined with human expertise, meaningful

constraints that are not specified or meaningless constraints that hold accidentally may be revealed.

Keys can be discovered from total tables by computing the agree sets of all pairs of distinct tuples, and then computing the transversals for their complements [13]. On general tables we distinguish between strong and weak agree sets, motivated by strong and weak similarity. Given two tuples t, t' over T, the *weak (strong) agree set* of t, t' is the (unique) maximal subset $X \subseteq T$ such that t, t' are weakly (strongly) similar on X. Given a table I over T, we denote by $\mathcal{AG}^w(I), \mathcal{AG}^s(I)$ the set of all maximal agree sets of distinct tuples in I:

$$\mathcal{AG}^w(I) := \max\{X \mid \exists t \neq t' \in I.t[X] \sim_w t'[X]\}$$
$$\mathcal{AG}^s(I) := \max\{X \mid \exists t \neq t' \in I.t[X] \sim_s t'[X]\} .$$

We shall write $\mathcal{AG}^w, \mathcal{AG}^s$ when I is clear from the context. It follows from Theorem 1 that $c\langle X\rangle$ does not hold on I (and $\neg_c\langle X\rangle$ holds on I) if and only if there is some $Y \in \mathcal{AG}^w(I)$ with $X \subseteq Y$. Similarly, $p\langle X\rangle$ does not hold on I (and $\neg_p\langle X\rangle$ holds on I) if and only if there is some $Y \in \mathcal{AG}^s(I)$ with $X \subseteq Y$.

Complements and transversals are standard notions for which we now introduce notation [13]. Let X be a subset of T and $\mathcal{S}$ a set of subsets of T. The transversal set of $\mathcal{S}$ consists of all minimal attribute subsets of T, minimal under set inclusion, that have non-empty intersection with all elements of $\mathcal{S}$. We denote complements of attribute sets by $\overline{X} = T \setminus X$, complements of sets of attribute sets by $\overline{\mathcal{S}} = \{\overline{X} \mid X \in \mathcal{S}\}$, and transversal sets by $Tr(\mathcal{S}) = \min_{\subseteq} \{Y \subseteq T \mid \forall X \in \mathcal{S}.Y \cap X \neq \emptyset\}$. Our main result on key discovery is that the certain (possible) keys that hold in I are the transversals of the complements for all weak (strong) agree sets in I.

Corollary 3 *Let I be a table over (T, T_S), and Σ_I the set of all certain and/or possible keys that hold on I. Then*

$$\Sigma := \{c\langle X\rangle \mid X \in Tr(\overline{\mathcal{AG}^w})\} \cup \{p\langle X\rangle \mid X \in Tr(\overline{\mathcal{AG}^s})\}$$

is a cover of Σ_I.

Proof By Theorem 1 X is a certain (possible) key for I

iff no distinct tuples $t, t' \in I$ weakly (strongly) agree on X
iff X is not a subset of any weak (strong) agree set
iff X intersects with $\overline{Y}$ for every weak (strong) agree set Y
iff X is a transversal of $\overline{\mathcal{AG}^w}$ ($\overline{\mathcal{AG}^s}$)

Due to the extension rules of Corollary 2 the minimal transversals are sufficient to form a cover. □

Our goal was to show that hypergraph transversals can be used to discover our keys in real-world data. Results of those experiments are presented in Section 8. Optimizations and scalability of our method to big data are left for future research [1, 43]. We illustrate the discovery method on our running example.

Example 9 Let I denote Table 1 over (T, T_S) from Example 1. Then

- $\mathcal{AG}^w(I) = \{\{title, author\}, \{journal\}\}$,
- $\overline{\mathcal{AG}^w} = \{\{journal\}, \{title, author\}\}$,
- $Tr(\overline{\mathcal{AG}^w}) = \{\{title, journal\}, \{author, journal\}$,
- $\mathcal{AG}^s(I) = \{\{title, author\}\}$,
- $\overline{\mathcal{AG}^s} = \{\{journal\}\}$, and
- $Tr(\overline{\mathcal{AG}^s}) = \{\{journal\}\}$.

Therefore, the set Σ from Example 4 is a cover of the set of possible and certain keys that hold on I. □

4 Armstrong Tables

Armstrong tables are widely regarded as user-friendly, exact representations of constraint sets [4, 16, 21, 38]. For a class $\mathcal{C}$ of constraints and a set Σ of constraints in $\mathcal{C}$, a $\mathcal{C}$-Armstrong table I for Σ satisfies Σ and violates all the constraints in $\mathcal{C}$ not implied by Σ. Therefore, given an Armstrong table I for Σ the problem of deciding for an arbitrary constraint φ in $\mathcal{C}$ whether Σ implies φ reduces to the problem of verifying whether φ holds on I. The ability to compute an Armstrong table for Σ provides us with a data sample that is a perfect semantic summary of Σ. Unfortunately, classes $\mathcal{C}$ may not enjoy Armstrong tables. That is, there are sets Σ for which no $\mathcal{C}$-Armstrong table exists [16]. Classically, the classes of keys and functional dependencies do enjoy Armstrong relations [4, 38]. However, the combined class of possible keys, certain keys, and `NOT NULL` constraints does not enjoy Armstrong tables. Even though the situation is involved, we will characterize when Armstrong tables do exist, and compute them in these cases.

4.1 Acquisition Framework

Applications benefit from the ability of data engineers to acquire the keys that are semantically meaningful in the domain of the application. As engineers do usually not know much about the domain they communicate with domain experts. However, domain experts do usually not know much about databases, which leads to a communication problem between engineers and domain experts. We establish two major tools to improve

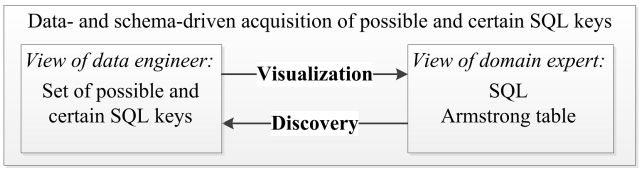

Fig. 1 Acquisition Framework for SQL Keys

communication, as shown in the agile framework of Figure 1. Here, data engineers use our algorithm to visualize an abstract set Σ of keys as an Armstrong table I_Σ, which is inspected jointly with domain experts. During inspection of I_Σ the experts may simply notice flaws in the current perception of the meaningful constraint set Σ. They may also change I_Σ or provide new data samples. Our discovery algorithm from Corollary 3 is then applied to discover the keys that hold in the given sample. This alternating process of visualization and discovery is continued until consensus is reached.

4.2 Definition and Motivating Examples

An instance I over (T, T_S) is a *pre-Armstrong table* for (T, T_S, Σ) if for every key φ over T, φ holds on I iff $\Sigma \models \varphi$. We call I an *Armstrong table* iff it is pre-Armstrong and for every attribute $A \in T \setminus T_S$ there is a tuple $t \in I$ with $t[A] = \perp$. There are cases where a pre-Armstrong table but no Armstrong table exists.

Example 10 Let $(T, T_S, \Sigma) = (AB, A, \{c\langle B\rangle\})$. The following is easily seen to be a pre-Armstrong table:

A	B
0	0
0	1

Now let I be an instance over (T, T_S, Σ) with $t \in I$ such that $t[B] = \perp$. The existence of some tuple $t' \in I$ with $t' \neq t$ would violate $c\langle B\rangle$, so $I = \{t\}$. That means $c\langle A\rangle$ holds on I even though $\Sigma \not\models c\langle A\rangle$, so I is not Armstrong. □

There are cases where no pre-Armstrong table exists.

Example 11 Let $(T, T_S) = (ABCD, \emptyset)$ and

$$\Sigma = \{c\langle AB\rangle, c\langle CD\rangle, p\langle AC\rangle, p\langle AD\rangle, p\langle BC\rangle, p\langle BD\rangle\}$$

Then a pre-Armstrong table I must disprove the certain keys $c\langle AC\rangle$, $c\langle AD\rangle$, $c\langle BC\rangle$, $c\langle BD\rangle$ while respecting the possible keys $p\langle AC\rangle$, $p\langle AD\rangle$, $p\langle BC\rangle$, $p\langle BD\rangle$. In each of these four cases, we require two tuples t, t' which are weakly but not strongly similar on the corresponding key sets (e.g. $t[AC] \sim_w t'[AC]$ but $t[AC] \not\sim_s t'[AC]$). This is only possible when t or t' are $\perp$ on one of the key attributes. This ensures the existence of tuples $t_{AC}, t_{AD}, t_{BC}, t_{BD}$ with

$$t_{AC}[A] = \perp \vee t_{AC}[C] = \perp,\ t_{AD}[A] = \perp \vee t_{AD}[D] = \perp,$$
$$t_{BC}[B] = \perp \vee t_{BC}[C] = \perp,\ t_{BD}[B] = \perp \vee t_{BD}[D] = \perp$$

These tuples are pairwise different because the certain keys $c\langle AB\rangle$, $c\langle CD\rangle$ must hold. If $t_{AC}[A] \neq \perp$ and $t_{AD}[A] \neq \perp$ it follows that $t_{AC}[C] = \perp$ and $t_{AD}[D] = \perp$. This means $t_{AC}[CD] \sim_w t_{AD}[CD]$, contradicting $c\langle CD\rangle$. Hence, there has to be some $t_A \in \{t_{AC}, t_{AD}\}$ with $t_A[A] = \perp$. Similarly we get $t_B \in \{t_{BC}, t_{BD}\}$ with $t_B[B] = \perp$. Then $t_A[AB] \sim_w t_B[AB]$ contradicting $c\langle AB\rangle$. Consequently, it is impossible to satisfy Σ and violate all p-keys and c-keys not implied by Σ. □

4.3 Structural Characterization

We will now characterize when Armstrong tables exist, and show how to construct them whenever possible. Overcoming the technical challenges has rewards in theory and practice. In practice, Armstrong tables facilitate the acquisition of requirements, see Figure 1. This is particularly appealing to our keys since the experiments in Section 8 confirm that i) key sets for which Armstrong tables do not exist are rare, and ii) keys that represent real application semantics can be enforced efficiently in SQL database systems. Our findings illustrate the impact of nulls on the theory of Armstrong databases, and have revealed a technical error in previous research [34], as shown in Section 4.5.

We will now begin to develop a structural characterization of (pre-)Armstrong tables, i.e., to provide sufficient and necessary conditions that allow us to decide whether a given table I is (pre-)Armstrong for a given set Σ of possible and certain keys over a given schema (T, T_S). We start with a lemma that characterizes when a given table that satisfies Σ is pre-Armstrong for Σ. A (pre-)Armstrong table must violate all possible and certain keys not implied by Σ. In other words, it must satisfy all strong and weak anti-keys implied by Σ. For this purpose, the table must contain for each strong (weak) anti-key, a pair of distinct tuples that strongly (weakly) agrees on the strong (weak) anti-key. However, it suffices when the table exhibits these strong (weak) agree sets for the maximal strong (weak) anti-keys because every pair of tuples that strongly (weakly) agrees on some attribute set also strongly (weakly) agrees on each of its subsets. We thus obtain the following result.

Lemma 1 *Let I be an instance over (T, T_S) such that Σ holds on I. Then I is a pre-Armstrong table of Σ iff*

i) for every strong anti-key $\neg_p\langle X\rangle \in \mathcal{A}^s_{max}$ there exist distinct tuples $t, t' \in I$ with $t[X] \sim_s t'[X]$, and

ii) for every weak anti-key $\neg_c\langle X\rangle \in \mathcal{A}^w_{max}$ there exist distinct tuples $t, t' \in I$ with $t[X] \sim_w t'[X]$.

Proof Keys implied by Σ hold on I by assumption, so we only need to examine keys on T that are not implied.

($\Rightarrow$) Clear by Theorem 1.

($\Leftarrow$) Assume i) and ii) hold. Let $c\langle X\rangle, X \subseteq T$ be a certain key with $\Sigma \not\models c\langle X\rangle$. Then $\neg_c\langle X\rangle$ is a weak anti-key and there exists a maximal (weak or strong) anti-key $\neg\langle Y\rangle$ with $X \subseteq Y$. In either case (by i) or ii)) there exist $t, t' \in I, t \neq t'$ with $t[Y] \sim_w t'[Y]$ and hence $t[X] \sim_w t'[X]$. Thus $c\langle X\rangle$ is violated by I.
Let $p\langle X\rangle, X \subseteq T$ be a possible key with $\Sigma \not\models p\langle X\rangle$. Then $\neg_p\langle X\rangle$ is a strong anti-key and there exists a maximal strong anti-key $\neg_p\langle Y\rangle$ with $X \subseteq Y$. By ii) there exist $t, t' \in I, t \neq t'$ with $t[Y] \sim_s t'[Y]$ and hence $t[X] \sim_s t'[X]$. Thus $p\langle X\rangle$ is violated by I.

This concludes the proof. □

Example 11 presented a case in which every table that satisfies the conditions of Lemma 1 must violate Σ. Hence, no pre-Armstrong table can exist for Σ.

We are now moving on to the challenging structural characterization of (pre-)Armstrong tables. Firstly, we describe the (academic) case in which Σ implies every possible and certain key. This happens precisely when Σ implies the empty key $c\langle\emptyset\rangle$. The empty key holds on an instance I iff it contains at most one tuple. An Armstrong table for such Σ is given by a single tuple that features $\perp$ on those columns outside of T_S, and actual domain values on columns in T_S.

In all other cases, a (pre-)Armstrong table must violate the empty key and contain at least two tuples. We show now that every such instance violates every certain key $c\langle X\rangle$ for which two tuples exist such that for every attribute A of X, at least one of the tuples is $\perp$ on A. More generally, we define the *$\perp$-base* of tuples $t_1, \ldots, t_n$ over T as the set of all attributes where some tuple is $\perp$:

$$\perp\text{-base}(t_1, \ldots, t_n) := \{A \in T \mid t_1[A] = \perp \vee \ldots \vee t_n[A] = \perp\}.$$

Lemma 2 *Let I be an instance over (T, T_S) with $|I| \geq 2$. Then I violates $c\langle\perp\text{-base}(t, t')\rangle$ for every $t, t' \in I$.*

Proof Let $X := \perp\text{-base}(t, t')$. If $t \neq t'$ then $t[X] \sim_w t'[X]$ so $c\langle X\rangle$ is violated. Otherwise $t = t'$ and $t[X] = (\perp, \ldots, \perp)$. Since I contains at least two tuples, there must exist $t'' \in I$ with $t \neq t''$, and we have $t[X] \sim_w t''[X]$ so $c\langle X\rangle$ is again violated. □

The core challenge towards a structural characterization of (pre-)Armstrong tables is to identify reconcilable situations between the given possible keys, the implied maximal weak anti-keys, and the null-free subschema. In fact, possible keys require some possible world in which all tuples disagree on some key attribute while weak anti-keys require some possible world in which some tuples agree on all key attributes. So, whenever a weak anti-key contains a possible key, the situation is only reconcilable by the use of $\perp$. However, such occurrences of $\perp$ can cause unintended weak similarity between tuple pairs, as seen in Example 11. Consequently, standard Armstrong table construction techniques [4,21,38], which essentially deal with each anti-constraint in isolation, cannot be applied here.

The main structure we require in our characterization is a set $\mathcal{W}$ that consists of sets of attributes $A \in T - T_S$. Intuitively, each of these sets consists of the attributes on which some tuple is $\perp$, i.e., $\mathcal{W}$ consists of the sets $\perp$-base(t) for tuples t in the given table. For two tuples t and t' of the table let $V = \perp$-base(t) and $W = \perp$-base(t'). Then t and t' are weakly similar on $V \cup W$, i.e., $V \cup W$ must be a weak anti-key in order for the table to satisfy all given certain keys. Therefore, the set system $\mathcal{W}$ describes a structure of sub-schemata on which null markers co-occur in tuple pairs. More generally, let $\mathcal{V}, \mathcal{W} \subseteq \mathcal{P}(T)$ be two sets of sets. The *cross-union* of $\mathcal{V}$ and $\mathcal{W}$ is defined as $\mathcal{V} \bowtie \mathcal{W} := \{V \cup W \mid V \in \mathcal{V}, W \in \mathcal{W}\}$. We abbreviate the cross-union of a set $\mathcal{W}$ with itself by $\mathcal{W}^{\times 2} := \mathcal{W} \bowtie \mathcal{W}$. The condition that the union of elements from $\mathcal{W}$ must form a weak anti-key can be stated by saying that every element of $\mathcal{W}^{\times 2}$ forms a weak anti-key, see condition i) below. In addition, a pre-Armstrong table must also satisfy all given possible keys. In particular, whenever such a possible key $p\langle X'\rangle$ is contained in some maximal weak anti-key X, then there must be some tuple pair in the table whose set $Y \in \mathcal{W}^{\times 2}$ of co-occurring null markers must non-trivially intersect with X', see condition ii) below. Finally, for every Armstrong table it must also hold that for every nullable attribute in $T - T_S$ there is some tuple that is $\perp$ on that attribute, see property iii) below. In what follows we first show that the three conditions are all necessary for a given table to be Armstrong. Subsequently, we will show how the existence of a set system $\mathcal{W}$ with these three conditions is also sufficient to construct an Armstrong table. This will establish the following structural characterization of (pre-)Armstrong tables.

Theorem 3 *Let $\Sigma \not\models c\langle\emptyset\rangle$. There exists some pre-Armstrong table for (T, T_S, Σ) iff there exists a set $\mathcal{W} \subseteq \mathcal{P}(T \setminus T_S)$ with the following properties:*

i) Every element of $\mathcal{W}^{\times 2}$ forms a weak anti-key.
ii) For every maximal weak anti-key $\neg_c\langle X\rangle \in \mathcal{A}^w_{max}$, there exists $Y \in \mathcal{W}^{\times 2}$, such that for every possible key $p\langle X'\rangle \in \Sigma$ with $X' \subseteq X$, we have $Y \cap X' \neq \emptyset$.

There exists an Armstrong table for (T, T_S, Σ) iff i) and ii) hold as well as

iii) $\bigcup \mathcal{W} = T \setminus T_S$.

Proof Let I be a pre-Armstrong table for (T, T_S, Σ) and $\Sigma \not\models c\langle\emptyset\rangle$. Define $\mathcal{W} \subseteq \mathcal{P}(T \setminus T_S)$ as

$$\mathcal{W} := \{\bot\text{-base}(t) \mid t \in I\}$$

We will show that conditions i) and ii) hold.

i) Let $Y_1, Y_2 \in \mathcal{W}$. Then there must exist $t_{Y_1}, t_{Y_2} \in I$ with $\bot\text{-base}(t_{Y_1}, t_{Y_2}) = Y_1Y_2$, so I violates $c\langle Y_1Y_2\rangle$ by Lemma 2. As I respects Σ, Σ permits $\neg_c\langle Y_1Y_2\rangle$.
ii) For every maximal weak anti-key $\neg_c\langle X\rangle \in \mathcal{A}^w_{max}$ there exist two distinct tuples $t, t' \in I$ such that $t[X] \sim_w t'[X]$. Now let $p\langle X'\rangle \in \Sigma$ with $X' \subseteq X$. As I respects Σ we cannot have $t[X'] \sim_s t'[X']$, i.e., t, t' are weakly but not strongly similar on X'. This is only possible for $t[A] = \bot$ or $t'[A] = \bot$ for some $A \in X'$. But that means $A \in X' \cap Y_1Y_2$ with $Y_1 := \bot\text{-base}(t), Y_2 := \bot\text{-base}(t')$.

If I is an Armstrong table, then there exists a tuple $t_A \in I$ with $t_A[A] = \bot$ for every $A \in T \setminus T_S$. Hence
iii) $T \setminus T_S \subseteq \bigcup\{\bot\text{-base}(t_A) \mid A \in T \setminus T_S\} \subseteq \bigcup \mathcal{W}$

The remaining "if" direction ($\Leftarrow$) will be shown in Theorem 4. □

We will now exploit Theorem 3 to devise a *general* construction of (pre-)Armstrong tables whenever they exist. In the construction, every maximal anti-key is represented by two new tuples. Strong anti-keys are represented by two tuples that strongly agree on the anti-key, while weak anti-keys are represented by two tuples that weakly agree on some suitable attributes of the anti-key as determined by the set $\mathcal{W}$ from Theorem 3. Finally, the set $\mathcal{W}$ permits us to introduce $\bot$ in nullable columns that do not yet feature an occurrence of $\bot$. For the construction we assume without loss of generality that all attributes have integer domains.

Construction 1 (Armstrong Table)
Let $\mathcal{W} \subseteq \mathcal{P}(T \setminus T_S)$ satisfy conditions *i)* and *ii)* of Theorem 3. We construct an instance I over (T, T_S) as follows.

I) For every strong anti-key $\neg_p\langle X\rangle \in \mathcal{A}^s_{max}$ add tuples $t^s_X, t^{s\prime}_X$ to I with

$$\begin{array}{ll} t^s_X[X] = (i, \ldots, i) & t^{s\prime}_X[X] = (i, \ldots, i) \\ t^s_X[T \setminus X] = (j, \ldots, j) & t^{s\prime}_X[T \setminus X] = (k, \ldots, k) \end{array}$$

where i, j, k are distinct integers not used before.
II) For every weak anti-key $\neg_c\langle X\rangle \in \mathcal{A}^w_{max}$ we add tuples $t^w_X, t^{w\prime}_X$ to I with

$$\begin{array}{ll} t^w_X[X \setminus Y_1] = (i, \ldots, i) & t^{w\prime}_X[X \setminus Y_2] = (i, \ldots, i) \\ t^w_X[X \cap Y_1] = (\bot, \ldots, \bot) & t^{w\prime}_X[X \cap Y_2] = (\bot, \ldots, \bot) \\ t^w_X[T \setminus X] = (j, \ldots, j) & t^{w\prime}_X[T \setminus X] = (k, \ldots, k) \end{array}$$

where $Y_1, Y_2 \in \mathcal{W}$ meet the condition for $Y = Y_1 \cup Y_2$ in *ii)* of Theorem 3, and i, j, k are distinct integers not used previously.
III) If condition *iii)* of Theorem 3 also holds for $\mathcal{W}$, then for every $A \in T \setminus T_S$ for which there is no t in I with $t[A] = \bot$, we add a tuple t_A to I with

$$t_A[T \setminus A] = (i, \ldots, i) \qquad t_A[A] = \bot$$

where i is an integer not used previously. □

Indeed, Construction 1 yields a (pre-)Armstrong table whenever one exists.

Theorem 4 *Let $\Sigma \not\models c\langle\emptyset\rangle$ and I a table generated by Construction 1. Then I is a pre-Armstrong table over (T, T_S, Σ). If condition iii) of Theorem 3 holds for $\mathcal{W}$, then I is an Armstrong table.*

Proof We first show the conditions of Lemma 1 are met.

i) Let $\neg_p\langle X\rangle$ be a strong anti-key in $\mathcal{A}^s_{max}$. Then I contains the tuples $t^s_X, t^{s\prime}_X$ with

$$t^s_X[X] = (i, \ldots, i) \sim_s (i, \ldots, i) = t^{s\prime}_X[X]$$

ii) Let $\neg_c\langle X\rangle$ be a weak anti-key in $\mathcal{A}^w_{max}$. Then I contains the tuples $t^w_X, t^{w\prime}_X$ with

$$t^w_X[X \setminus Y_1Y_2] = (i, \ldots, i) \sim_w (i, \ldots, i) = t^{w\prime}_X[X \setminus Y_1Y_2]$$
$$t^w_X[X \cap Y_1] = (\bot, \ldots, \bot) \sim_w t^{w\prime}_X[X \cap Y_1]$$
$$t^w_X[X \cap Y_2] \sim_w (\bot, \ldots, \bot) = t^{w\prime}_X[X \cap Y_2]$$

It remains to show that I is a valid instance over (T, T_S, Σ), i.e., that I honors T_S and does not violate constraints in Σ. Honoring of T_S is clear by choice of $Y, Z, A \subseteq T \setminus T_S$.

i) Let $p\langle X'\rangle \in \Sigma$ and $t, t' \in I, t \neq t'$ with $t[X'] \sim_s t'[X']$. Since $X' \neq \emptyset$ and tuples constructed for different anti-keys use unique values, we must have $\{t, t'\} = \{t^s_X, t^{s\prime}_X\}$ or $\{t, t'\} = \{t^w_X, t^{w\prime}_X\}$ for some maximal anti-key $\neg\langle X\rangle$ with $X' \subseteq X$. The former cannot happen since $p\langle X'\rangle \in \Sigma$ implies $\Sigma \models p\langle X\rangle$, so $\{t, t'\} = \{t^w_X, t^{w\prime}_X\}$ and $\neg_c\langle X\rangle$ is a maximal weak anti-key. But then $\bot\text{-base}(t, t')$ intersects with X' due to condition ii), so $t[X'] \sim_s t'[X']$ cannot hold.
ii) Let $c\langle X'\rangle \in \Sigma$ and $t, t' \in I, t \neq t'$ with $t[X'] \sim_w t'[X']$.
   – If $X' \not\subseteq \bot\text{-base}(t, t')$ we have again $\{t, t'\} = \{t^s_X, t^{s\prime}_X\}$ or $\{t, t'\} = \{t^w_X, t^{w\prime}_X\}$ for some maximal anti-key $\neg\langle X\rangle$ with $X' \subseteq X$. Either way $\neg_c\langle X\rangle$ and thus $\neg_c\langle X'\rangle$ is a weak anti-key, which contradicts $\Sigma \models c\langle X'\rangle$.

– If $X' \subseteq \bot\text{-base}(t,t')$ then $X' \subseteq Y$ for some $Y \in \mathcal{W}^{\times 2}$. But $\neg_c\langle Y\rangle$ is a weak anti-key by condition i), again contradicting $\Sigma \models c\,\langle X'\rangle$.

If condition iii) of Theorem 3 holds for $\mathcal{W}$, then construction step III) ensures that I is Armstrong. □

We illustrate Construction 1 on our running example and show how Table 1 can be derived from it.

Example 12 Let T, T_S, and Σ be given as in Example 4. This gives us the maximal strong and weak anti-keys

$$\mathcal{A}_{max} = \{\neg_p\langle author, title\rangle, \neg_c\langle journal\rangle\}$$

with the set $\mathcal{W} = \{journal\}$ meeting the conditions of Theorem 3. Now Construction 1 produces the Armstrong table

title	*author*	*journal*
0	0	0
0	0	1
1	1	⊥
2	2	⊥

.

Here, we can remove either the third or the fourth tuple because removal of either tuple would preserve the weak agree set $\{journal\}$. After removal and suitable substitution we obtain Table 1. □

The conditions of Theorem 3 are difficult to test in general, due to the large number of candidate sets $\mathcal{W}$. However, there are cases where testing becomes simple.

Corollary 4 *Let $\Sigma \not\models c\,\langle\emptyset\rangle$.*

i) If $\Sigma \not\models c\,\langle X\rangle$ for every $X \subseteq T \setminus T_S$ then there exists an Armstrong table for (T, T_S, Σ).
ii) If $\Sigma \models c\,\langle X\rangle$ for some $X \subseteq T \setminus T_S$ with $|X| \leq 2$ then there does not exist an Armstrong table for (T, T_S, Σ).

Proof i) follows from Theorem 3 with $\mathcal{W} = \{T \setminus T_S\}$. For case ii), let I be an Armstrong table for (T, T_S, Σ), and $AB \subseteq T \setminus T_S$. Then I contains tuples t_A, t_B with $t_A[A] = \bot$ and $t_B[B] = \bot$, so $AB \subseteq \bot\text{-base}(t_A, t_B)$ is a weak anti-key by Lemma 2. □

Example 13 Let $(T = NDGS, T_S = N)$ and Σ consist of $c\,\langle ND\rangle$ and $p\,\langle S\rangle$ with Name N, Date of Birth D, Gender G and Security Number S. This gives us the maximal strong and weak anti-keys

$$\mathcal{A}_{max} = \{\neg_p\langle NG\rangle, \neg_p\langle DG\rangle, \neg_c\langle NGS\rangle, \neg_c\langle DGS\rangle\}$$

with the set $\mathcal{W} = \{DGS\}$ meeting the conditions of Theorem 3. Construction 1 yields the Armstrong table:

Name	*DoB*	*Gender*	*SNumber*
Ian (0)	10/03/1950 (0)	M (0)	04-3452-8903 (0)
Ian (0)	01/06/1975 (1)	M (0)	15-9385-2948 (1)
Bor (1)	02/04/1981 (2)	F (1)	23-5039-1293 (2)
Cam (2)	02/04/1981 (2)	F (1)	39-3920-4813 (3)
Kel (3)	03/07/1965 (3)	⊥	⊥
Kel (3)	04/02/1989 (4)	⊥	⊥
Don (4)	⊥	⊥	⊥
Sid (5)	⊥	⊥	⊥

with the integers of the construction indicated. □

4.4 Computational Characterization

We present a worst-case double exponential time algorithm for computing Armstrong tables whenever they exist. Our experiments in Section 8 show that these worst cases do not arise in practice, and our computation is very efficient. For example, our algorithm requires milliseconds when brute force approaches would require $2^{2^{20}}$ operations. While the exact complexity of the Armstrong table existence problem remains open, we establish results which suggest that the theoretical worst-case bound is difficult to improve upon.

Algorithm. Our goal is to compute the set $\mathcal{W}$ of Theorem 3 whenever it exist. We first observe that maximal anti-keys can be constructed using transversals. The proof of the following lemma is given in the appendix.

Lemma 3 *Let Σ^P, Σ^C be the sets of possible and certain keys in Σ. For readability we will identify attribute sets with the keys or anti-keys induced by them. Then*

$$\mathcal{A}^s_{max} = \{\ X \in \overline{Tr(\Sigma^C \cup \Sigma^P)}\ \mid \neg(X \subseteq T_S \wedge \exists Y \in \mathcal{A}^w_{max}.\ X \subset Y)\ \}$$
$$\mathcal{A}^w_{max} = \overline{Tr(\Sigma^C \cup \{X \in \Sigma^P \mid X \subseteq T_S\})} \setminus \mathcal{A}^s_{max}$$

For computing $\mathcal{A}^s_{max}$ and $\mathcal{A}^w_{max}$ by Lemma 3, anti-keys contained in strictly larger weak anti-keys cannot be set-wise maximal weak anti-keys. Hence, the set of all maximal weak anti-keys can be computed as

$$\mathcal{A}^w_{max} = \overline{Tr(\Sigma^C \cup \{X \in \Sigma^P \mid X \subseteq T_S\})} \setminus \overline{Tr(\Sigma^C \cup \Sigma^P)}$$

The difficult part in deciding existence of (and then computing) an Armstrong table given (T, T_S, Σ) is to check existence of (and construct) a set $\mathcal{W}$ meeting the conditions of Theorem 3, in cases where Corollary 4 does not apply. Blindly testing all subsets of $\mathcal{P}(T \setminus T_S)$ becomes infeasible when $T \setminus T_S$ contains more than 4 elements (for 5 elements, up to $2^{32} \approx 4,000,000,000$ sets would need to be checked). To ease discussion we will rephrase condition ii) of Theorem 3. Let $\mathcal{W} \subseteq \mathcal{P}(T)$. We say that

– $\mathcal{W}$ *supports* $Y \subseteq T$ if $Y \subseteq Z$ for some $Z \in \mathcal{W}$.
– $\mathcal{W}$ $\vee$*-supports* $\mathcal{V} \subseteq \mathcal{P}(T)$ if $\mathcal{W}$ supports *some* $Y \in \mathcal{V}$.
– $\mathcal{W}$ $\wedge\vee$*-supports* $\mathcal{T} \subseteq \mathcal{P}(\mathcal{P}(T))$ if $\mathcal{W}$ $\vee$-supports *every* $\mathcal{V} \in \mathcal{T}$.

We write $Y \Subset \mathcal{W}$ to indicate that $\mathcal{W}$ supports Y.

Lemma 4 *Let $\mathcal{T}_X$ be the transversal set in $T \setminus T_S$ of all possible keys that are subsets of X, and $\mathcal{T}$ the set of all such transversals for all maximal weak anti-keys:*

$$\mathcal{T}_X := Tr(\{X' \cap (T \setminus T_S) \mid p\,\langle X'\rangle \in \Sigma_{min} \wedge X' \subseteq X\})$$
$$\mathcal{T} := \{\mathcal{T}_X \mid X \in \mathcal{A}^w_{max}\}$$

Then condition ii) of Theorem 3 can be rephrased as:

ii') $\mathcal{W}^{\times 2}$ $\wedge\vee$-supports $\mathcal{T}$.

Proof Condition ii) states that for every $\neg_c\langle X\rangle \in \mathcal{A}^w_{max}$ there exists $Y \in \mathcal{W}^{\times 2}$ which traverses $\{X' \mid p\,\langle X'\rangle \in \Sigma_{min} \wedge X' \subseteq X\}$. Since $Y \subseteq T \setminus T_S$ this means Y traverses $\{X' \cap (T \setminus T_S) \mid p\,\langle X'\rangle \in \Sigma_{min} \wedge X' \subseteq X\}$. Hence Y is a superset of a minimal transversal in $\mathcal{T}_X$, i.e. $\mathcal{W}^{\times 2}$ $\vee$-supports $\mathcal{T}_X$. This holds for every $\neg_c\langle X\rangle \in \mathcal{A}^w_{max}$, so $\mathcal{W}^{\times 2}$ $\wedge\vee$-supports $\mathcal{T}$. These deductions also hold in reverse order. □

We propose the following: For each $\mathcal{T}_X \in \mathcal{T}$ and each minimal transversal $t \in \mathcal{T}_X$ we generate all non-trivial bi-partitions[2] $\mathcal{B}_\mathcal{X}$ (or just a trivial partition for transversals of cardinality < 2). We then add to $\mathcal{W}$ one such bi-partition for every $\mathcal{T}_X$ to meet condition ii), and combine them with all single-attribute sets $\{A\} \subseteq T \setminus T_S$ to meet condition iii). This is done for every possible combination of bi-partitions until we find a set $\mathcal{W}$ that meets condition i), or until we have tested them all. We then optimize this strategy: If a set $\mathcal{P}_X$ is already $\vee$-supported by $\mathcal{W}^{\times 2}$ (which at the time of checking will contain only picks for some sets $\mathcal{T}_X$), we may remove $\mathcal{P}_X$ from further consideration, as long as we keep all current picks in $\mathcal{W}$. In particular, since all single-attribute subsets of $T \setminus T_S$ are added to $\mathcal{W}$, we may ignore all $\mathcal{P}_X$ containing a set of size 2 or less. In pseudo-code, this strategy is given in Algorithm 1.

Proposition 3 *Algorithm Armstrong-Set is correct.*

Proof Lines 2 and 4 return sets $\mathcal{W}$ satisfying some condition of Corollary 4, and thus also conditions i) to iii). Failing that, $\mathcal{W}$ is defined so that condition iii) of Theorem 3 holds. Since function Extend-Support returns a superset of $\mathcal{W}$, condition iii) is invariant for $\mathcal{W}$.

Condition i) of Theorem 3 holds for the initial $\mathcal{W}$ due to line 3. Every subsequent enlargements of $\mathcal{W}$ in line 18 ensure condition i), due to the check in line 17. Hence, condition i) is invariant for $\mathcal{W}$.

It remains to show condition ii). For $\mathcal{W} = \{\{A\} \mid A \in T \setminus T_S\}$ the set $\mathcal{W}^{\times 2}$ already $\vee$-supports transversal sets containing transversals of cardinality 2 or less. Thus for every extension $\mathcal{W}_+$ of $\mathcal{W}$ for which $\mathcal{W}^{\times 2}$ $\wedge\vee$-supports the set $\mathcal{T}$ after removal of such transversal sets in line 8, the set $\mathcal{W}_+^{\times 2}$ $\wedge\vee$-supports the original $\mathcal{T}$. Thus condition ii) follows directly from the correctness of function Extend-Support and Lemma 4. This correctness can be shown recursively: If the check in line 13 holds $\mathcal{W}^{\times 2}$ already $\vee$-supports $\mathcal{T}_X$. If it does not, $\vee$-support of $\mathcal{T}_X$ is ensured by extending $\mathcal{W}$ with Y_1, Y_2 in line 18. The recursive call in line 14 or 18 ensures that $\mathcal{W}'^{\times 2}$ $\wedge\vee$-supports $\mathcal{T} \setminus \{\mathcal{T}_X\}$.

It remains to argue that function Armstrong-Set returns such a set $\mathcal{W}$ (rather than $\perp$) whenever one exists. Essentially we are examining all minimal [3] sets $\mathcal{W}$ which meet conditions ii) and iii), with a shortcut in line 4 which is justified due to Corollary 4. In lines 8 and 14 we only omit cases leading to non-minimal sets

Algorithm 1 Armstrong-Set

Input: T, T_S, Σ
Output: $\mathcal{W} \subseteq \mathcal{P}(T \setminus T_S)$ meeting conditions i) to iii) of Theorem 3 if such $\mathcal{W}$ exists, $\perp$ otherwise
1: **if** $\neg\exists c\,\langle X\rangle \in \Sigma$ with $X \subseteq T \setminus T_S$ **then**
2: **return** $\{T \setminus T_S\}$
3: **if** $\exists c\,\langle X\rangle \in \Sigma$ with $X \subseteq T \setminus T_S$ and $|X| \leq 2$ **then**
4: **return** $\perp$
5: $\mathcal{W} := \{\{A\} \mid A \in T \setminus T_S\}$
6: $\mathcal{A}^w_{max} := \overline{Tr(\Sigma^C \cup \{X \in \Sigma^P \mid X \subseteq T_S\})} \setminus \overline{Tr(\Sigma^C \cup \Sigma^P)}$
7: $\mathcal{T} := \{\, Tr(\{X' \cap (T \setminus T_S) \mid p\,\langle X'\rangle \in \Sigma_{min} \wedge X' \subseteq X\}) \mid X \in \mathcal{A}^w_{max} \,\}$
8: $\mathcal{T} := \mathcal{T} \setminus \{\mathcal{T}_X \in \mathcal{T} \mid \exists Y \in \mathcal{T}_X.\, |Y| \leq 2\}$
9: **return** Extend-Support$(\mathcal{W}, \mathcal{T})$

Subroutine Extend-Support$(\mathcal{W}, \mathcal{T})$
Input: $\mathcal{W} \subseteq \mathcal{P}(T \setminus T_S)$ meeting conditions i) and iii) of Theorem 3, $\mathcal{T} \subseteq \mathcal{P}(\mathcal{P}(T \setminus T_S))$
Output: $\mathcal{W}' \supseteq \mathcal{W}$ meeting conditions i) and iii) of Theorem 3 such that $\mathcal{W}'^{\times 2}$ $\wedge\vee$-supports $\mathcal{T}$ if such $\mathcal{W}'$ exists, $\perp$ otherwise
10: **if** $\mathcal{T} = \emptyset$ **then**
11: **return** $\mathcal{W}$
12: $\mathcal{T} := \mathcal{T} \setminus \{\mathcal{T}_X\}$ for some $\mathcal{T}_X \in \mathcal{T}$
13: **if** $\mathcal{W}^{\times 2}$ $\vee$-supports $\mathcal{T}_X$ **then**
14: **return** Extend-Support$(\mathcal{W}, \mathcal{T})$
15: **for all** $Y \in \mathcal{T}_X$ **do**
16: **for all** non-trivial bi-partitions $Y = Y_1 \cup Y_2$ **do**
17: **if** $(\mathcal{W} \cup \{Y_1, Y_2\})^{\times 2}$ contains no certain key **then**
18: $\mathcal{W}' :=$ Extend-Support$(\mathcal{W} \cup \{Y_1, Y_2\}, \mathcal{T})$
19: **if** $\mathcal{W}' \neq \perp$ **then**
20: **return** $\mathcal{W}'$
21: **return** $\perp$

[2] A partition of cardinality two.

[3] w.r.t. the $\wedge$-support ordering $\mathcal{W} \lesssim \mathcal{W}' :\Leftrightarrow \forall Y \in \mathcal{W}.\exists Y' \in \mathcal{W}'.Y \subseteq Y'$

$\mathcal{W}$. Since condition i) holds for a set $\mathcal{W}$ if it holds for any larger[3] set $\mathcal{W}'$, examining only minimal sets $\mathcal{W}$ is sufficient to find a set $\mathcal{W}$ meeting all conditions of Theorem 3, should one exist. □

We illustrate the construction with an example.

Example 14 Let $(T, T_S) = (ABCDE, \emptyset)$ and

$$\Sigma = \left\{ \begin{array}{l} p\langle A\rangle, p\langle B\rangle, p\langle CD\rangle, \\ c\langle ABE\rangle, c\langle ACE\rangle, c\langle ADE\rangle, c\langle BCE\rangle \end{array} \right\}$$

Neither condition of Corollary 4 is met, so Algorithm Armstrong-Set initializes/computes $\mathcal{W}$, $\mathcal{A}^w_{max}$ and $\mathcal{T}$ as

$$\begin{aligned} \mathcal{W} &= \{A, B, C, D, E\} \\ \mathcal{A}^w_{max} &= \{ABCD, AE, BDE, CDE\} \\ \mathcal{T} &= \{\{ABC, ABD\}\} \end{aligned}$$

Then, in method Extend-Support, $\mathcal{T}_x = \{ABC, ABD\}$ which is not $\vee$-supported by $\mathcal{W}^{\times 2}$. The non-trivial bi-partitions (Y_1, Y_2) of ABC are (A, BC), (AC, B), and (AB, C). None of these are suitable for extending $\mathcal{W}$, as the extension $(\mathcal{W} \cup \{Y_1, Y_2\})^{\times 2}$ contains the certain keys BCE, ACE and ABE, respectively. The non-trivial bi-partitions of the second set ABD are (A, BD), (AD, B), and (AB, D). As (AD, B) and (AB, D) are again unsuitable, (A, BD) can be used to extend $\mathcal{W}$ to the Armstrong set

$$\begin{aligned} \mathcal{W}' &= \text{Extend-Support}(\{A, BD, C, E\}, \emptyset) \\ &= \{A, BD, C, E\} \end{aligned}$$

If we add $c\langle BDE\rangle$ to the schema, then (A, BD) becomes unsuitable and no Armstrong set exists. □

Hardness. The main complexity of constructing Armstrong relations for keys in the relational model goes into the computation of all anti-keys. This is just the beginning for Algorithm 1, line 6. When computing the possible combinations of bi-partitions in line 16, Algorithm 1 becomes worst-case double exponential. We provide evidence that this worst-case bound may be difficult to improve upon. For this purpose, we study the key/anti-key satisfiability problem. Given a schema (T, T_S, Σ) and a set $\mathcal{A}^w \subseteq \mathcal{P}(T)$ of weak anti-keys, we say that $(T, T_S, \Sigma, \mathcal{A}^w)$ is satisfiable if there is a table I over (T, T_S, Σ) such that every element of $\mathcal{A}^w$ is a weak anti-key for I. The key/anti-key satisfiability problem is to decide if the input $(T, T_S, \Sigma, \mathcal{A}^w)$ is satisfiable.

Lemma 5 *A schema $(T, T_S, \Sigma, \mathcal{A}^w)$ is satisfiable iff there is a set $\mathcal{W} \subseteq \mathcal{P}(T \setminus T_S)$ such that:*

i) Every element of $\mathcal{W}^{\times 2}$ is a weak anti-key for Σ, and

ii) For every weak anti-key $\neg_c\langle X\rangle \in \mathcal{A}^w$ and for every possible key $p\langle X'\rangle \in \Sigma$ with $X' \subseteq X$ there exists $Y \in \mathcal{W}^{\times 2}$ with $Y \cap X' \neq \emptyset$.

Proof Analogous to the proof of Theorem 3. □

The pre-Armstrong existence problem can be reduced to the key/anti-key satisfiability problem by computing the set of maximal weak anti-keys for Σ. We followed this approach in Algorithm 1 that computes a set $\mathcal{W}$ of Theorem 3 whenever such a set exists. Therefore, by showing key/anti-key satisfiability to be NP-complete, we show that every attempt to decide Armstrong existence more efficiently will likely need to take a very different route. As $\mathcal{W}$ can be exponential in the size of Σ, every such approach must not compute $\mathcal{W}$ at all. The NP-hard problem we reduce to the key/anti-key satisfiability problem is *monotone 1-in-3 SAT* [48], which asks: Given a set of 3-clauses without negations, does there exist a truth assignment such that every clause contains exactly one true literal?

Proposition 4 *The key/anti-key satisfiability problem is NP-complete.*

Proof We will reduce the monotone 1-in-3 SAT problem to it. Let $\mathcal{S}^{AT}$ be an arbitrary set of 3-clauses without negation. We construct an instance of the key/anti-key satisfiability problem as follows.

$$\begin{aligned} T &:= XYZ \cup \bigcup_{\mathcal{C} \in \mathcal{S}^{AT}} \mathcal{C} \\ \mathcal{A}^w &:= \{\neg_c\langle XABC\rangle, \neg_c\langle YABC\rangle \mid ABC \in \mathcal{S}^{AT}\} \cup \{Z\} \\ \Sigma &:= \{p\langle A\rangle \mid A \in T\} \cup \\ &\qquad \left\{ \begin{array}{l} c\langle ZABC\rangle, c\langle XYAB\rangle, \\ c\langle XYAC\rangle, c\langle XYBC\rangle \end{array} \middle| \, ABC \in \mathcal{S}^{AT} \right\} \end{aligned}$$

We claim that $(T, \emptyset, \Sigma, \mathcal{A}^w)$ is satisfiable iff $\mathcal{S}^{AT}$ is 1-in-3 satisfiable.

Let I be a table satisfying $(T, \emptyset, \Sigma, \mathcal{A}^w)$, and $\mathcal{W}$ as in Lemma 5. We may assume w.l.o.g. that $\mathcal{W}$ is *downward closed*, i.e., that $\mathcal{W} = \bigcup_{w \in \mathcal{W}} \mathcal{P}(w)$.

Since every attribute in T is nullable and a possible key, we must have $\mathcal{A}^w \subseteq \mathcal{W}^{\times 2}$. In particular $Z \in \mathcal{W}$, and for every $ABC \in \mathcal{S}^{AT}$ we have $XABC, YABC \in \mathcal{W}^{\times 2}$. Since $c\langle ZABC\rangle \in \Sigma$ we cannot have $ABC \in \mathcal{W}$. This leaves XA, XB or $XC \in \mathcal{W}$, and similarly YA, YB or $YC \in \mathcal{W}$. The certain keys

$$c\langle XYAB\rangle, c\langle XYAC\rangle, c\langle XYBC\rangle \in \Sigma$$

mean that $XA \in \mathcal{W}$ prohibits both $YB \in \mathcal{W}$ and $YC \in \mathcal{W}$ so $YA \in \mathcal{W}$ must hold. Conversely $YA \in \mathcal{W}$ prohibits both $XB \in \mathcal{W}$ and $XC \in \mathcal{W}$. By symmetrical argument exactly one of XA, XB, XC lies in $\mathcal{W}$.

Thus $\mathcal{S}^{AT}$ is 1-in-3 satisfiable with

$$A \to \begin{cases} \text{true} & \text{if } XA \in \mathcal{W} \\ \text{false} & \text{if } XA \notin \mathcal{W} \end{cases}$$

Conversely let $\mathcal{S}^{AT}$ be 1-in-3 satisfiable with truth assignment $\mathcal{L}$. Then the set

$$\mathcal{W} := \{XA, YA, BC \mid ABC \in \mathcal{S}^{AT} \wedge \mathcal{L}(A)\} \cup \{Z\}$$

meets the conditions of Lemma 5.

The above shows NP-hardness. If there is an instance I satisfying the given keys and anti-keys, it contains a subtable with at most $2 \cdot |\mathcal{A}^w|$ tuples, and thus can be guessed and verified in polynomial time. □

In summary, the pre-Armstrong existence problem is solved as follows: First, we construct the sets of maximal weak (and strong) anti-keys. These sets can be exponential in the size of Σ. Second, we then reduce the problem of finding $\mathcal{W}$ to the NP-complete key/anti-key satisfiability problem.

The construction of the anti-keys is in EXPTIME, and finding $\mathcal{W}$ is in NP (EXPTIME in our algorithm), but $\mathcal{W}$ is exponential in the size of the input Σ because $\mathcal{W}$ includes the exponentially large set of anti-keys. So, overall the problem lies in NEXPTIME, and our algorithm runs in double-exponential time and exponential space. The NP-hardness proof of the key/anti-key satisfiability problem does not actually prove anything about the hardness of checking the existence of $\mathcal{W}$, in particular as the second part of finding $\mathcal{W}$ from the keys and anti-keys covers only a subset, say X, of the problem space for key/anti-key satisfiability (where we do not require a relationship between keys and anti-keys here). But we know that every PTIME solution would need to find a way to solve the key/anti-key satisfiability problem for instances in X in PTIME, and NP-hardness of the key/anti-key satisfiability problem suggests that this may not be easy (or possible).

Despite the poor theoretical worst-case bounds of the problem, our experiments in Section 8 show that our algorithm is typically very efficient.

4.5 Weak and Strong FDs resist Armstrong relations

Note that there is a "strong similarity" between possible and certain keys and the weak and strong functional dependencies discussed in [34]. Despite this, [34, Theorem 6.1] claims that for a set of strong and weak functional dependencies, an Armstrong relation always exists. Unfortunately that claim is wrong, with the proof not considering all tuple pairs in Case 2, "if" direction.

We must note here that Armstrong tables in [34] correspond to our pre-Armstrong tables, they do not consider `NOT NULL` constraints, and a weak/strong FD $X \to T$ may hold on a relation over T while the corresponding possible/certain key $_{(p/c)}\langle X \rangle$ does not. Hence non-existence of a (pre-)Armstrong table for a set of possible and certain keys does not imply non-existence of an Armstrong table for the corresponding set of weak and strong FDs.

Thus, to show that the claim in [34, Theorem 6.1] is wrong, rather than just its proof, we provide a counter example next. Following the notation of [34], we denote weak FDs by $\Diamond(X \to Y)$ and strong FDs by $\Box(X \to Y)$.

Example 15 (no FD (pre-)Armstrong table)
Let $T = ABCDE$ and

$$\Sigma = \left\{ \begin{array}{l} \Box(AB \to E), \Box(CD \to E), \\ \Diamond(AC \to E), \Diamond(AD \to E), \\ \Diamond(BC \to E), \Diamond(BD \to E), \\ \Box(E \to ABCD) \end{array} \right\}$$

Note the similarity to Example 11. Now every Armstrong table I must disprove the strong FDs

$$\Box(AC \to E), \Box(AD \to E), \Box(BC \to E), \Box(BD \to E)$$

while respecting the weak FDs

$$\Diamond(AC \to E), \Diamond(AD \to E), \Diamond(BC \to E), \Diamond(BD \to E)\ .$$

I cannot contain $\perp$ in column E, or else $\Box(E \to ABCD)$ forces all tuples to be strongly similar on $ABCD$, causing non-implied FDs to hold.

In each of the four cases, we require two tuples t, t' which violate a strong FD $\Box(X \to E)$ but satisfy $\Diamond(X \to E)$. Thus

$$\begin{array}{l} t[X] \sim_w t'[X] \text{ and } t[E] \not\sim_s t'[E] \text{ , and} \\ t[X] \not\sim_s t'[X] \quad \text{or} \quad t[E] \sim_w t'[E] \end{array}$$

Since $t[E], t'[E] \neq \perp$ they cannot be weakly similar without being strongly similar. Hence we have $t[X] \not\sim_s t'[X]$, meaning either $t[X]$ or $t'[X]$ must contain $\perp$.

Using the same arguments as in Example 11, we obtain (e.g.) $t_A[AB] \sim_w t_B[AB]$ with $t_A[AB]$ and/or $t_B[AB]$ containing $\perp$. However, from $\Box(AB \to E)$ and $\Box(E \to ABCD)$ it then follows that $t_A[AB] \sim_s t_B[AB]$, contradicting the presence of $\perp$. □

5 Finite Domains

So far, we have assumed that domains are infinite, or at least "sufficiently large". While the domains of some key attributes are usually sufficiently large, there may be real-world situations in which all domains are fairly

small, e.g. `boolean`. We will now investigate possible and certain keys under finite and infinite domains, that is, we do not assume domains to be infinite. For the obvious reason, we do require the domains of all key attributes to contain at least two elements.

5.1 Syntactic Characterization

Certain keys enjoy a robust syntactic characterization.

Corollary 5 *Permitting finite domains, $X \subseteq T$ is a certain key for I iff no two tuples in I with distinct tuple identities are weakly similar on X.*

Proof The proof of Theorem 1 does not make use of the infinite domain assumption for certain keys. □

Possible keys are not robust, as the following examples illustrate.

Example 16 (possible-key satisfaction)
Assuming the domain of *Gender* only contains `M` and `F`, consider the following instance I:

Name	*Gender*	*DoB*
Alex	`F`	23/07/2000
Alex	$\perp$	04/01/1999
Ben	`M`	04/01/1999

Here, the possible key $p\langle Gender\rangle$ does not hold on I. However, no tuples are strongly similar on *Gender*, and $p\langle Gender\rangle$ does hold on every 2-subset of I, that is, every subset of cardinality 2.

Furthermore, the possible keys $p\langle Name, Gender\rangle$ and $p\langle Gender, DoB\rangle$ each hold individually, that is, for each possible key there exists a possible world satisfying that key. However, neither of the two possible worlds satisfies both keys. □

This shows that pairwise tuple comparison alone is insufficient to decide whether a possible key holds, and that it is insufficient to decide satisfaction of each possible key in isolation. This makes it much harder to enforce possible keys – NP-hard in fact, as we show now.

Problem 1 (possible-key satisfaction)
Given a schema T, an instance I over T and a set Σ of keys over T, does there exist a possible world of I satisfying Σ?

Proposition 5 *The possible key satisfaction problem is NP-complete for finite domains.*

Proof The membership in NP follows from the fact that we can guess a possible world of I and verify in time polynomial in the input (the instance I and the key set Σ) that I satisfies Σ.

To show NP-hardness, we will reduce 3-SAT to it. Let $\mathcal{V} = \{v_1, \ldots, v_n\}$ denote the set of variables, and $\mathcal{C} = \{c_1, \ldots, c_m\}$ the set of clauses after removing all tautologies. We construct T, I and Σ as follows:

- T contains an attribute A_i with sufficiently large domain for every clause $c_i \in \mathcal{C}$, and an attribute B_i with boolean domain for every variable $v_i \in \mathcal{V}$.
- Σ contains a key $A_iB_jB_kB_l$ for every clause $c_i = \{(\neg)v_j, (\neg)v_k, (\neg)v_l\} \in \mathcal{C}$.
- I contains a tuple t_0 with
  - $t_0[A_i] = 0$ for $i = 1 \ldots n$
  - $t_0[B_i] = \perp$ for $i = 1 \ldots m$

  and a tuple t_i for every clause $c_i \in \mathcal{C}$ with
  - $t_i[A_i] = 0$
  - $t_i[A_j] = i$ for $j \neq i$
  - $t_i[B_j] = F$ for $v_j \in c_i$
  - $t_i[B_j] = T$ for $v_j \notin c_i$

Now each possible world $\rho(I)$ corresponds to a truth assignment via $v_i = \rho(t_o[A_i])$, and one may verify that this truth assignment satisfies $\mathcal{C}$ iff Σ holds on $\rho(I)$. □

For these reasons we shall abandon our semantic notion of possible keys for finite domains, and investigate the combination of certain keys, uniqueness constraints and `NOT NULL` constraints instead. According to the SQL standard, uniqueness constraints enjoy the syntactic characterization of Theorem 1. That is, `UNIQUE`(X) holds on table I if and only if I does not contain two tuples with different tuple identities that are strongly similar on X. In addition, when we speak about finite domains, the notion of strong anti-keys is derived from that of uniqueness constraints. That is, X is a strong anti-key for I, denoted by $\neg_p\langle X\rangle$, if `UNIQUE`(X) does *not* hold on I.

5.2 Implication, Anti-keys and Key Discovery

Our results about implication of certain and possible keys, in particular Theorem 2 (implication), Corollaries 1 (implication complexity) and 2 (axiomatization), and Proposition 2 (minimal cover) continue to hold for finite domains when we replace possible keys by uniqueness constraints. This can be seen as follows (recall that we require domains to contain at least 2 values):

(i) In Theorem 2 the "if" direction follows from Theorem 1 which still holds, and the proof for the "only if" direction only uses instances with 2 tuples and thus still works under finite domains.

(ii) The remaining corollaries and proposition follow from Theorem 2.

Similarly our results about the implication between anti-keys, namely Proposition 1, hold over finite domains when we replace possible keys with uniqueness constraints. This follows from its proof, which uses only 2 distinct domain values.

Finally, Corollary 3 (key discovery) holds over finite domains when we replace possible keys by uniqueness constraints, as it only depends on Theorems 1 and 2.

5.3 Armstrong Tables

During the construction of Armstrong tables, having sufficiently many domain values is crucial to prevent interaction of different tuples. As the following example shows, finite domains can already prevent the existence of Armstrong tables in the relational case.

Example 17 (No Relational Armstrong Table)
Consider relational schema $T = ABC$ with the keys $\Sigma = \{A, BC\}$. If the domain of A contains only two distinct values, then every Armstrong table contains at most 2 tuples. To satisfy the maximal anti-keys B and C, these tuples must be identical on both B and C, which would violate the key BC. Thus no Armstrong table exists. □

In the example above, the problem is caused by a key containing only attributes with finite domains, restricting the number of tuples allowed in every instance. Clearly this limits the practical usability of such tables.

Note also that the question of whether an Armstrong table exists under finite domains is at least as hard as finding the size of a *minimum-sized* Armstrong table under infinite domains (same trick as in Example 17), a problem for which no exact results are known.

For these reasons we will focus on Armstrong tables for instances where no restriction is imposed on the number of tuples, that is, where every key contains at least one attribute with an infinite domain. This allows us to obtain a result similar to Corollary 4 for finite domains.

Theorem 5 *If every certain key $c\langle X\rangle \in \Sigma$ contains an attribute $A \in T_S$ with infinite domain, and every uniqueness constraint* `UNIQUE`$(Y) \in \Sigma$ *contains an attribute B with infinite domain, then there exists an Armstrong table for (T, T_S, Σ).*

Proof We construct an Armstrong table I by adapting Construction 1 with $\mathcal{W} = \{T \setminus T_S\}$ to finite domains, as follows. Denote the attributes with finite and infinite domain by $T_F \cup T_\infty = T$ respectively. We further abbreviate $T_{S,F} := T_S \cap T_F$ and $T_{S,\infty} := T_S \cap T_\infty$.

I) For every strong anti-key $\neg_p\langle X\rangle \in \mathcal{A}^s_{max}$ add tuples $t^s_X, t^{s\prime}_X$ to I with

$$\begin{array}{ll} t^s_X[X \cap T_\infty] = (i, \ldots, i) & t^{s\prime}_X[X \cap T_\infty] = (i, \ldots, i) \\ t^s_X[X \cap T_F] = (0, \ldots, 0) & t^{s\prime}_X[X \cap T_F] = (0, \ldots, 0) \\ t^s_X[T_\infty \setminus X] = (j, \ldots, j) & t^{s\prime}_X[T_\infty \setminus X] = (k, \ldots, k) \\ t^s_X[T_F \setminus X] = (0, \ldots, 0) & t^{s\prime}_X[T_F \setminus X] = (1, \ldots, 1) \end{array}$$

where i, j, k are distinct integers not used prior.

II) For every weak anti-key $\neg_c\langle X\rangle \in \mathcal{A}^w_{max}$ we add tuples $t^w_X, t^{w\prime}_X$ to I with

$$\begin{array}{ll} t^w_X[X \cap T_{S,\infty}] = (i, \ldots, i) & t^{w\prime}_X[X \cap T_{S,\infty}] = (i, \ldots, i) \\ t^w_X[X \cap T_{S,F}] = (0, \ldots, 0) & t^{w\prime}_X[X \cap T_{S,F}] = (0, \ldots, 0) \\ t^w_X[X \setminus T_S] = (\bot, \ldots, \bot) & t^{w\prime}_X[X \setminus T_S] = (\bot, \ldots, \bot) \\ t^w_X[T_\infty \setminus X] = (j, \ldots, j) & t^{w\prime}_X[T_\infty \setminus X] = (k, \ldots, k) \\ t^w_X[T_F \setminus X] = (0, \ldots, 0) & t^{w\prime}_X[T_F \setminus X] = (1, \ldots, 1) \end{array}$$

where i, j, k are distinct integers not used prior.

III) Add a tuple $t_\bot$ to I with

$$\begin{aligned} t_\bot[T_{S,F}] &= (0, \ldots, 0) \\ t_\bot[T_{S,\infty}] &= (i, \ldots, i) \\ t_\bot[T \setminus T_S] &= \bot \end{aligned}$$

where i is an integer not used previously.

Clearly all maximal strong and weak anti-keys hold on I, and all columns in $T \setminus T_S$ contain a $\bot$ value. It remains to show that Σ holds on I.

By construction all matching tuple-pairs $t^s_X, t^{s\prime}_X$ and $t^w_X, t^{w\prime}_X$ use values on attributes with infinite domains that are either $\bot$ or distinct from all other tuples. As all keys contain an attribute with infinite domain (and `NOT NULL` for certain keys), no key in Σ can be violated by non-matching tuple-pairs.

Consider `UNIQUE`$(Y) \in \Sigma$. For a strong anti-key $\neg_p\langle X\rangle \in \mathcal{A}^s_{max}$ we must have $Y \nsubseteq X$. As the matching tuple-pair $t^s_X, t^{s\prime}_X$ differs on $Y \setminus X$, it does not violate `UNIQUE`(Y). For a weak anti-key $\neg_c\langle X\rangle \in \mathcal{A}^w_{max}$ we must have $Y \nsubseteq X \cap T_S$. As the matching tuple-pair $t^w_X, t^{w\prime}_X$ either differs or is $\bot$ outside of $X \cap T_S$, it does not violate `UNIQUE`(Y).

Consider $c\langle Y\rangle \in \Sigma$. For a strong or weak anti-key $\neg\langle X\rangle \in \mathcal{A}_{max}$ we must have $Y \nsubseteq X$. As the matching tuple-pair $t^s_X, t^{s\prime}_X$ or $t^w_X, t^{w\prime}_X$ strongly differs on $Y \setminus X$, it does not violate $c\langle Y\rangle$. □

The equivalence of possible keys with SQL `UNIQUE` requires infinite domains (or large enough, finite, domains). This section has established several results for SQL `UNIQUE` which even hold for finite domains. Database users may find reassurance in SQL's adoption of the syntactic definition of `UNIQUE`, since it enables all of these results. Indeed, we showed that a possible world semantics for `UNIQUE` "only" works when some large enough domains are available.

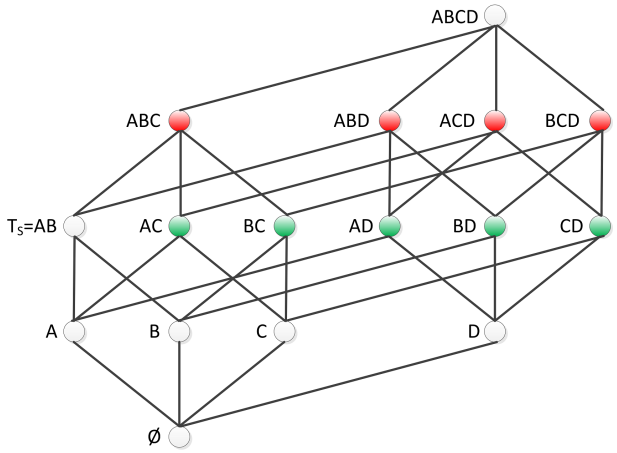

Fig. 2 Maximum Non-redundant Family of Certain Keys (in red) and Possible Keys (in green)

6 Combinatorics of SQL Keys

Data engineers want to know how complex the maintenance of their data can grow. Here, we provide answers to the basic question which non-redundant families of possible and certain keys attain maximum cardinality. We even provide a complete solution to this question when an upper bound on the number of key attributes is stipulated. The results provide further insight to other problems we have studied in this article: They tell us how large non-redundant inputs to the implication problem or Armstrong computation may become, how large non-redundant outputs to the discovery problem may become, and how many indices we may need to specify to support all access paths that uniquely identify entities. Our solution is derived by applying techniques from extremal set theory. The result is interesting from a combinatorial perspective itself, as it generalizes the famous theorem by Sperner [50].

Throughout this section we write $[n] := \{1, \ldots, n\}$ instead of $T = \{A_1, \ldots, A_n\}$. For $X \subseteq [n]$, we use the notations $X^{(i)} := \{Y \subseteq X : |Y| = i\}$ and $X^{(\leq i)} := \{Y \subseteq X : |Y| \leq i\}$. A family $\mathcal{A} \subseteq [n]^{(\leq n)}$ is an *anti-chain* (or *Sperner family*) if $X \not\subseteq Y$ for all distinct $X, Y \in \mathcal{A}$. For a set Σ of possible and certain keys over $[n]$, define $\mathcal{F} := \{X \mid c\langle X\rangle \in \Sigma\}$ and $\mathcal{G} := \{X \mid p\langle X\rangle \in \Sigma\}$. The following theorem is just a simple translation of Theorem 2 into a language to which we can apply extremal set theory.

Proposition 6 *Let Σ be a set of possible and certain keys, and T_S the set of all attributes declared* `NOT NULL`*. Then Σ is non-redundant if and only if (1) $\mathcal{F}$ is an anti-chain, (2) $\mathcal{G}$ is an anti-chain, (3) $\forall F \in \mathcal{F}, G \in \mathcal{G}: F \not\subseteq G$, and (4) $\forall F \in \mathcal{F}, G \in \mathcal{G}: G \not\subseteq F \cap T_S$.* □

Using Proposition 6 the problem we study reads as follows. Given non-negative integers n, k with $k \leq n$ and a set $T_S \subseteq [n]$, find all pairs $(\mathcal{F}, \mathcal{G}) \in [n]^{(\leq k)} \times [n]^{(\leq k)}$ that satisfy conditions (1)–(4) of Proposition 6 and maximize $|\mathcal{F} \cup \mathcal{G}|$. Note that $\mathcal{F}$ and $\mathcal{G}$ must be disjoint because of (3). In what follows, a complete solution is given.

First, we briefly discuss the case when $T_S = [n]$. In this case, (1)–(4) are satisfied if and only if $\mathcal{F} \dot\cup \mathcal{G} \subseteq [n]^{(\leq k)}$ is an anti-chain. If $k > n/2$, then by Sperner's Theorem [50] $|\mathcal{F} \cup \mathcal{G}|$ attains its maximum if and only if $\mathcal{F}$ and $\mathcal{G}$ form a partition of $[n]^{(\lfloor n/2 \rfloor)}$ or of $[n]^{(\lceil n/2 \rceil)}$, respectively. If $k \leq n/2$, then $|\mathcal{F} \cup \mathcal{G}|$ is maximized if and only if $\mathcal{F}$ and $\mathcal{G}$ form a partition of $[n]^{(k)}$. This is well-known and follows from the original proof of Sperner's Theorem [50]. In the sequel, we will assume that $0 \leq |T_S| < n$. Note that the bound (1) below does not hold when $T_S = [n]$ for even n and $k > n/2$.

Theorem 6 *Let n, a, k be non-negative integers with $n \geq 2$, $a < n$ and $k \leq n$, and let $T_S \subseteq [n]$ with $|T_S| = a$. Furthermore, let $\mathcal{F}, \mathcal{G} \subseteq [n]^{(\leq k)}$ be anti-chains such that $F \not\subseteq G$ and $G \not\subseteq F \cap T_S$ for all $F \in \mathcal{F}$ and $G \in \mathcal{G}$. Then*

$$|\mathcal{F} \cup \mathcal{G}| \leq \binom{n+1}{m} - \binom{a}{m-1}, \tag{1}$$

where $m := \min\{k, \lfloor n/2 \rfloor + 1\}$. This bound is the best possible, and equality is attained if and only if

(i) $\mathcal{F} \dot\cup \mathcal{G} = [n]^{(m)} \cup ([n]^{(m-1)} \setminus T_S^{(m-1)})$, *or*

(ii) $\mathcal{F} \dot\cup \mathcal{G} = [n]^{(m-1)} \cup ([n]^{(m-2)} \setminus T_S^{(m-2)})$, *where n is even, $k > n/2$, and $a \leq n/2 - 2$ or $a = n - 1$*

Furthermore, for $k > 1$, (i) implies

$$[n]^{(m)} \setminus T_S^{(m)} \subseteq \mathcal{F} \quad \text{and} \quad [n]^{(m-1)} \setminus T_S^{(m-1)} \subseteq \mathcal{G}$$

while (ii) implies

$$[n]^{(m-1)} \setminus T_S^{(m-1)} \subseteq \mathcal{F} \quad \text{and} \quad [n]^{(m-2)} \setminus T_S^{(m-2)} \subseteq \mathcal{G}\,.$$

Note that $[n]^{(m)}$ and $[n]^{(m-1)}$ are the two largest disjoint anti-chains over $[n]^{(\leq k)}$. To prevent $F \subseteq G$, F must contain the larger sets. To prevent $G \subseteq F \cap T_S$ we drop small subsets of T_S. The remaining subsets of T_S may be attributed to either F or G as they do not conflict with others. This reflects the equivalence of certain/possible keys over `NOT NULL` attributes. While case (i) always presents a non-redundant family of maximum cardinality, case (ii) occurs because for even n and large k, an additional pair of largest disjoint anti-chains exists: $[n]^{(m-1)}$ and $[n]^{(m-2)}$. The difference between the resulting sets lies in the size of $T_S^{(m-1)}$ and $T_S^{(m-2)}$, leading to the conditions for a. The trivial case $n = 1$ is excluded above to avoid certain technicalities in its proof.

Example 18 For table schema ($T = \{A, B, C, D\}, T_S = \{A, B\}$), or $n = 4$ and $a = 2$, the maximum cardinality of a non-redundant family of possible and certain keys is nine. The bound is attained by the set Σ where

$$\Sigma = \left\{ \begin{array}{l} c\langle A, B, C\rangle, c\langle A, B, D\rangle, c\langle A, C, D\rangle, c\langle B, C, D\rangle, \\ p\langle A, C\rangle, p\langle B, C\rangle, p\langle A, D\rangle, p\langle B, D\rangle, c\langle B, D\rangle \end{array} \right\}.$$

The associated powerset lattice is shown in Figure 2, where certain keys are marked red and possible keys are marked blue. □

The proof of Theorem 6 can be found in the appendix.

7 Enforcing Certain Keys

Certain keys can uniquely identify entities in an SQL table. However, the unique identification of entities must also be efficient in order to meet Codd's rule of entity integrity. In practice, an efficient enforcement of certain keys requires index structures. Finding suitable index structures for certain keys is non-trivial as weak similarity is not transitive. Hence, classical indices will not work directly. We present an index scheme, based on multiple classical indices, which allows us to check certain keys efficiently, provided there are few nullable key attributes. While an efficient index scheme seems elusive for larger sets of nullable attributes, our experiments from Section 8 suggest that most certain keys in practice have few nullable attributes.

Let (T, T_S) be a table schema and $X \subseteq T$. A *certain-key-index* for $c\langle X\rangle$ is a collection of indices $\mathfrak{I}_Y$ on subsets Y of X which include all `NOT NULL` attributes of X, that is, $\mathfrak{I}_{c\langle X\rangle} := \{\mathfrak{I}_Y \mid X \cap T_S \subseteq Y \subseteq X\}$. Here we treat $\perp$ as regular value for the purpose of indexing, i.e., we do index tuples with $\perp$ "values". When indexing a table I, each tuple in I is indexed in each $\mathfrak{I}$.

Obviously, $|\mathfrak{I}_{c\langle X\rangle}| = 2^n$, where $n := |X \setminus T_S|$, which makes maintenance efficient only for small n. When checking if a tuple exists that is weakly similar to some given tuple, we only need to consult a single index, but within that index we must perform up to 2^n lookups.

Proposition 7 *Let t be a tuple on (T, T_S) and $\mathfrak{I}_{c\langle X\rangle}$ a certain-key-index for table I over (T, T_S). Define*

$$K := \{A \in X \mid t[A] \neq \perp\}.$$

Then the existence of a tuple in I weakly similar to t can be checked with at most 2^k lookups in $\mathfrak{I}_K$, where $k := |K \setminus T_S|$.

Proof The tuple $t[K]$ can be weakly similar to every tuple in $\{t' \mid \forall A \in K \setminus T_S(t'(A) = t(A) \text{ or } t'(A) = \perp)\}$. Hence, there are up to 2^k tuples in $\mathfrak{I}_K$ that need to be looked up. □

As $|K \setminus T_S|$ is bounded by $|X \setminus T_S|$, lookup is efficient whenever indexing is efficient.

Example 19 Consider schema $(ABCD, A, \{c\langle ABC\rangle\})$ with table I over it:

$I =$

A	B	C	D
1	$\perp$	$\perp$	1
2	2	$\perp$	2
3	$\perp$	3	$\perp$
4	4	4	$\perp$

The certain-key-index $\mathfrak{I}_{c\langle ABC\rangle}$ for $c\langle ABC\rangle$ consists of:

$\mathfrak{I}_A$

A
1
2
3
4

$\mathfrak{I}_{AB}$

A	B
1	$\perp$
2	2
3	$\perp$
4	4

$\mathfrak{I}_{AC}$

A	C
1	$\perp$
2	$\perp$
3	3
4	4

$\mathfrak{I}_{ABC}$

A	B	C
1	$\perp$	$\perp$
2	2	$\perp$
3	$\perp$	3
4	4	4

These tables represent attribute values that we index by, using every standard index structure such as B-trees. When checking whether tuple $t := (2, \perp, 3, 4)$ is weakly similar on ABC to some tuple $t' \in I$ (and thus violating $c\langle ABC\rangle$ when inserted), we perform lookup on $\mathfrak{I}_{AC}$ for tuples t' with $t'[AC] \in \{(2, 3), (2, \perp)\}$. □

Our indexing approach works for every type of index (e.g. hash, B-tree), as we only require fast lookup of exact matches. Some index structures however, such as B-tree and its variants, also allow efficient lookup of prefix matches. That is, an index over $ABCD$ also supports lookups on A, AB and ABC. Such index structures are available in most DBMSs, so we can exploit them to reduce the number of indexes required. If $X = YA$ contains only a single `NULL` attribute A, we can support $c\langle X\rangle$ with a single prefix index on YA, as this allows lookups for both Y and YA. If $X = YAB$ contains two `NULL` attributes A and B, we can support $c\langle X\rangle$ with two prefix indices on YAB and YB. While such an approach cannot prevent an exponential growth in the number of indices required to achieve performance guarantees (each prefix index only supports a linear number of prefix sets), it significantly reduces the number of indices required in practice. When building a prefix index it helps to order the nullable columns, so that columns with high selectivity and few null marker occurrences (potentially contradicting measures) appear first. High selectivity ensures that we gain greater precision out of the index when the column is used, and few null marker occurrences ensure that follow-up columns are less likely to be excluded from use. The potential for additional gains from these techniques is rather small, because a single index on the columns that are `NOT NULL` tends to have excellent performance already, as illustrated in the next section.

8 Experiments

We conducted several experiments to evaluate various aspects of our work. Firstly, we discovered possible and certain keys in public data sets. Secondly, we tested our algorithms for computing and deciding the existence of Armstrong tables. Lastly, we considered storage space and time requirements for our index scheme. We used the following data sets for our experiments:

- GO-termdb (Gene Ontology)
  `www.geneontology.org/`
- IPI (International Protein Index)
  `www.ebi.ac.uk/IPI`
- LMRP (Local Medical Review Policy)
  `www.cms.gov/medicare-coverage-database/`
- PFAM (protein families)
  `pfam.sanger.ac.uk/`
- RFAM (RNA families)
  `rfam.sanger.ac.uk/`

These data sets were chosen for their availability in non-curated database format. This was important because null marker occurrences and inconsistencies have often been cleaned in curated databases. In principle, there is potential for bias, for example related to the interpretation of null markers, but it is unclear to us why or in which form that would occur. The experiments were run on a Dell Latitude E5530, Intel core i7, CPU 2.9GHz with 8GB RAM on a 64-bit operating system. The experiments regarding key enforcement were conducted in MySQL version 5.6.

8.1 Key Discovery

Examining the schema definition does not suffice to decide what types of key constraints hold or should hold on a database. Certain keys with $\perp$ occurrences cannot be expressed in current SQL databases, so would be lost. Even constraints that could and should be expressed are often not declared. Even if `NOT NULL` constraints are declared, one frequently finds that these are invalid, resulting in work-arounds such as empty strings. We mined data tables for possible and certain keys, with focus on finding certain keys with $\perp$ occurrences. For deciding if a column is `NOT NULL`, we ignored all schema-level declarations, and instead tested whether a column contained $\perp$ occurrences. To alleviate string formatting issues (such as "Zwieb C." vs "Zwieb C.;") we normalized strings by trimming non-word non-decimal characters, and interpreting the empty string as $\perp$. This pre-processing was necessary because several of our test data sets were available only in CSV format, where $\perp$ occurrences would have been exported as empty strings. Tables with less than two tuples were ignored. In the figures reported, we exclude tables `pfamA` and `lcd` from the PFAM and LMRP data sets, as they contain over 100,000 (`pfamA`) and over 2000 (`lcd`) minimal keys, respectively, almost all of which appear to be 'by chance', and thus would distort results completely. Table 6 lists the number of minimal keys of each type discovered in the 130 tables. We distinguish between possible and certain keys with `NULL` columns, and keys not containing `NULL` columns. For the latter, possible and certain keys coincide.

Key Type	Occurrences
Certain Keys with `NULL`	45
Possible Keys with `NULL`	237
Keys without `NULL`	309

Table 6 Number of keys found by type

Two factors likely have a significant impact on the figures in Table 6. First, constraints may only hold accidentally, especially on small tables. For example, Table 1 of the RFAM data set satisfies $c\langle title, journal\rangle$ and $c\langle author, journal\rangle$, with `NULL` column *journal*. Only the first key appears to be sensible. Second, constraints that should sensibly hold may be violated due to lack of enforcement. Certain keys with $\perp$ occurrences, which cannot easily be expressed in existing systems, are likely to suffer from this effect more than others. Indeed, most SQL users know how to enforce a primary key but likely find it challenging to enforce a certain key, for example by triggers. We thus consider our results qualitative rather than quantitative. Still, they indicate that certain keys do appear in practice, and may benefit from explicit support by database systems.

We argue that more meaningful keys can be found by relaxing conditions during key discovery. In an effort to identify potentially meaningful certain keys that are violated due to lack of enforcement, we have mined all 130 tables for *certain near-keys of order* n. These are satisfied by a given data set if there are no more than n tuples with distinct tuple identities that are pairwise weakly similar on the given set of attributes. Pairwise weak similarity is sufficient and necessary for the existence of a possible world in which the n tuples agree on the given attributes. Certain keys are certain near-keys of order 1. For $n = 1, \ldots, 5$ we obtained the results in Table 7. In total there are 324 certain near-keys with null occurrences in some key column.

The order of a certain near-key indicates the maximum degree of violation in an instance. We now use the frequency by which violations occur in order to determine whether a certain near-key represents a meaningful certain key and does not only hold accidentally on an instance I. Thus, we define the *conflict rate* of I for

Order n of certain near-key	#NULLs	Frequency	Min conflict rate	Max conflict rate
1	1	41	0	0
1	2	4	0	0
2	1	107	<1	100
2	2	13	<1	20
3	1	45	<1	70
3	2	12	1	6
4	1	40	1	37
4	2	6	7	42
4	3	1	54	54
5	1	43	<1	100
5	2	11	7	42
5	3	1	55	55

Table 7 Number of certain near-keys found up to order 5, and their minimum and maximum conflict rates

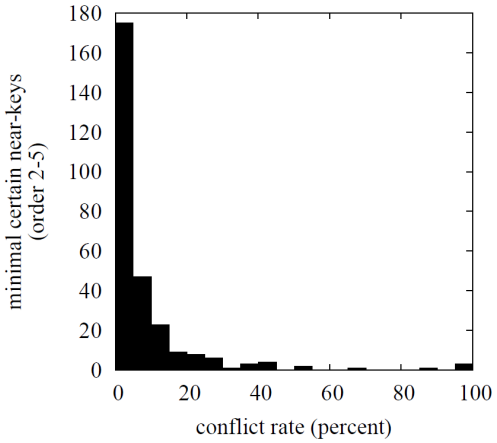

Fig. 3 Distribution of conflict rates

a certain near-key of order n as the percentage of rows in I that are weakly similar on the given attributes to at least one other row in I. The last two columns of Table 7 show the minimum and maximum conflict rates for the corresponding certain near-keys. Low conflict rates can be taken as an indicator for keys to be meaningful. However, this fails when many duplicates exist. The distribution of conflict rates over all minimal certain near-keys with null columns, of orders 2-5, is shown in Figure 3, where order 1 is omitted as conflict rates are 0. For many of the minimal certain near-keys identified, their conflict rate is small. This confirms their status as potentially meaningful certain keys. Ultimately however, not knowing the ground truth of what keys are meaningful, it is impossible to classify keys as meaningful or accidental.

The mining of certain keys from all 130 tables took a total of 246 seconds, while the mining of certain near-keys of up to order 5 from all 130 tables required less than 5.6 times as much time, namely 1369 seconds. These numbers report the averages over ten runs.

8.2 Armstrong Tables

We applied Algorithm 1 and Construction 1 to compute Armstrong tables for the 130 tables we mined possible and certain keys from. As each certain key contained some `NOT NULL` column, Corollary 4 applied in all cases. Each Armstrong table was computed within a few milliseconds, and they contained only 7 tuples on average.

We also tested our algorithms against 1 million randomly generated table schemas and sets of keys. Each table contained 5-25 columns, with each column having a 50% chance of being `NOT NULL`, and 5-25 keys of size 1-5, with equal chance of being possible or certain. To avoid overly silly examples, we removed certain keys with only 1 or 2 `NULL` attributes and no `NOT NULL` attributes. Hence, case ii) of Corollary 4 never applies.

In all but 85 cases an Armstrong table existed, with average computation time in the order of a few milliseconds. For larger random schemas, transversal computation can become a bottleneck [11,13].

Together these results suggest that, although Algorithm 1 is worst-case double exponential, such hard cases need to be carefully constructed and arise neither in practice nor by chance (at least not frequently).

8.3 Indexing

The efficiency of our index scheme for certain keys with $\perp$ occurrences depends directly on the number of `NULL` attributes in the certain key. Thus the central question is how many `NULL` attributes occur in certain keys in practice. For 45 certain keys with $\perp$ occurrences, discovered from the data sets in our experiments, 41 certain keys have only 1 column with null marker occurrences, while 4 certain keys have 2 columns with null marker occurrences, see Table 7. In addition, Table 7 shows that even for certain near-keys between orders 2-5, at most three columns had occurrences of $\perp$, and those with only one column dominated.

Therefore, certain keys with $\perp$ occurrences contain mostly only a single attribute on which $\perp$ occurs (requiring 2 standard indices), and rarely more than two attributes (requiring 4 standard indices). Assuming these results generalize to other data sets, certain keys with $\perp$ occurrences can be enforced efficiently in practice.

We used triggers to enforce certain keys under different combinations of B-tree indices, and compared these to the enforcement of the corresponding primary keys. For all experiments the schema was ($T = ABCDE, T_S = A$) and the table I over T contained 100M tuples. In each experiment we inserted 10,000 times one tuple and took the average time to perform this operation. This includes the time for maintaining the index structures

Index	$T = ABCDE, T_S = A$		
	$X = AB$	$X = ABC$	$X = ABCD$
$PK(X)$	0.451	0.491	0.615
$\mathfrak{I}_X$	0.764	0.896	0.977
$\mathfrak{I}_{T_S}$	0.723	0.834	0.869
$\mathfrak{I}_{c\langle X\rangle}$	0.617	0.719	1.143

Table 8 Average Times to Enforce Keys on X

involved. We enforced $c\langle X\rangle$ in the first experiment for $X = AB$, in the second experiment for $X = ABC$ and in the third experiment for $X = ABCD$, incrementing the number of `NULL` columns from 1 to 3. The distribution of permitted $\perp$ occurrences was evenly spread amongst the 100M tuples, and also amongst the 10,000 tuples to be inserted. Altogether, we run each of the experiments for 3 index structures: i) $\mathfrak{I}_X$, ii) $\mathfrak{I}_{T_S}$, iii) $\mathfrak{I}_{c\langle X\rangle}$. The times were compared against those achieved under declaring a primary key $PK(X)$ on X, where we had 100M X-total tuples. Our results are shown in Table 8. All times are in milliseconds.

Hence, certain keys can be enforced efficiently as long as we involve the columns in T_S in some index. Just having $\mathfrak{I}_{T_S}$ ensures a performance similar to that of the corresponding primary key. Indeed, the `NOT NULL` attributes of a certain key suffice to identify most tuples uniquely. Our experiments confirm these observations even for certain keys with three `NULL` columns, which occur rarely in practice. Of course, $\mathfrak{I}_{T_S}$ cannot guarantee the efficiency bounds established for $\mathfrak{I}_{c\langle X\rangle}$ in Proposition 7. We stress the importance of indexing: enforcing $c\langle X\rangle$ on our data set without an index resulted in a performance loss in the order of 10^4.

Memory consumption is proportional to the number of indices used. In particular, it is no worse than classic indices if only one index is used. In other cases, consumption will be larger by up to (and probably close to) a factor of $2^{\sharp\text{nullable key-columns}}$.

9 Conclusion and Future Work

Primary key columns must not feature null markers and therefore oppose the requirements of modern applications, inclusive of high volumes of incomplete data. We studied keys over SQL tables in which the designated null marker may represent missing or unknown data. Both interpretations can be handled by a possible world semantics. A key is possible (certain) to hold on a table if some (every) possible world of the table satisfies the key. Possible keys capture SQL's `UNIQUE` constraint, and certain keys generalize SQL's primary keys by permitting null markers in key columns. We established solutions to several computational problems related to possible and certain keys under `NOT NULL` constraints. These include axiomatic and linear-time algorithmic characterizations of their implication problem, minimal representations of keys, discovery of keys from a given table, and structural and computational properties of Armstrong tables. Using extremal set theory, we also characterized which families of possible and certain keys attain maximum cardinality under given `NOT NULL` constraints, even under given upper bounds on their size. Experiments confirm that our solutions work effectively and efficiently in practice. This also applies to enforcing certain keys, by utilizing known index structures. Our findings from public data confirm that certain keys have only few columns with null marker occurrences. Certain keys thus achieve the goal of Codd's rule for entity integrity while accommodating the requirements of modern applications. This distinct advantage over primary keys comes only at a small price in terms of update performance.

This research encourages various future work. Our keys have been extended to possible and certain functional dependencies, which enable SQL schema normalization [31]. The exact complexity of deciding the existence of Armstrong tables should be determined. It is worth investigating optimizations to reduce the time complexity of computing Armstrong tables, and their size. Domain or database experts may not be available during the acquisition of keys. Such situations call for automated support in indicating that a target key holds on the Armstrong table, or why it is violated. This may prompt the experts to change the keys (table), and informed interaction may still progress until the experts are satisfied. A combination of the discovery and Armstrong table algorithms result in informative Armstrong samples for our keys, similar to classical data dependencies [39]. Evidently, the existence and computation of Armstrong relations for sets of weak and strong functional dependencies requires new attention [34]. Scalable approximate discovery is an important problem as meaningful keys may be violated, but research has only started for total [25,49] and possible keys [25]. These algorithms may be extended to handle the weak similarity of tuples and thus the discovery of certain keys. Other index schemes may prove valuable to enforce certain keys. It is interesting to combine possible and certain keys with possibilistic or probabilistic keys [6,29], or inclusion dependencies with nulls [30,41,42].

Acknowledgements We thank Mozhgan Memari for carrying out the experiments for key enforcement. This research is partially supported by the Marsden Fund Council from New Zealand Government funding, by the Natural Science Foundation of China (Grant No. 61472263) and the Australian Research Council (Grants No. DP140103171).

References

1. Z. Abedjan, L. Golab, and F. Naumann. Profiling relational data: a survey. *VLDB J.*, 24(4):557–581, 2015.
2. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
3. S. Aleksic, M. Celikovic, S. Link, I. Lukovic, and P. Mogin. Faceoff: Surrogate vs. natural keys. In *ADBIS*, pages 543–546, 2010.
4. C. Beeri, M. Dowd, R. Fagin, and R. Statman. On the structure of Armstrong relations for functional dependencies. *J. ACM*, 31(1):30–46, 1984.
5. J. Biskup. *Security in Computing Systems - Challenges, Approaches and Solutions*. Springer, 2009.
6. P. Brown and S. Link. Probabilistic keys for data quality management. In *CAiSE*, pages 118–132, 2015.
7. A. Calì, D. Calvanese, and M. Lenzerini. Data integration under integrity constraints. In *Seminal Contributions to Information Systems Engineering*, pages 335–352. 2013.
8. D. Calvanese, W. Fischl, R. Pichler, E. Sallinger, and M. Simkus. Capturing relational schemas and functional dependencies in RDFS. In *AAAI*, pages 1003–1011, 2014.
9. E. F. Codd. Extending the database relational model to capture more meaning. *ACM Trans. Database Syst.*, 4(4):397–434, 1979.
10. E. F. Codd. *The Relational Model for Database Management, Version 2*. Addison-Wesley, 1990.
11. J. David, L. Lhote, A. Mary, and F. Rioult. An average study of hypergraphs and their minimal transversals. *Theor. Comput. Sci.*, 596:124–141, 2015.
12. S. Doherty. The future of enterprise data. `http://insights.wired.com/profiles/blogs/the-future-of-enterprise-data#axzz2owCB8FFn`, 2013.
13. T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM J. Comput.*, 24(6):1278–1304, 1995.
14. K. Engel. *Sperner Theory*. Cambridge Uni Press, 1997.
15. R. Fagin. A normal form for relational databases that is based on domains and keys. *ACM Trans. Database Syst.*, 6(3):387–415, 1981.
16. R. Fagin. Horn clauses and database dependencies. *J. ACM*, 29(4):952–985, 1982.
17. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.
18. W. Fan, F. Geerts, and X. Jia. A revival of constraints for data cleaning. *PVLDB*, 1(2):1522–1523, 2008.
19. M. Hannula, J. Kontinen, and S. Link. On independence atoms and keys. In *CIKM*, pages 1229–1238, 2014.
20. M. Hannula, J. Kontinen, and S. Link. On the finite and general implication problems of independence atoms and keys. *J. Comput. Syst. Sci.*, 82(5):856–877, 2016.
21. S. Hartmann, M. Kirchberg, and S. Link. Design by example for SQL table definitions with functional dependencies. *VLDB J.*, 21(1):121–144, 2012.
22. S. Hartmann, U. Leck, and S. Link. On Codd families of keys over incomplete relations. *Comput. J.*, 54(7):1166–1180, 2011.
23. S. Hartmann and S. Link. Efficient reasoning about a robust XML key fragment. *ACM Trans. Database Syst.*, 34(2), 2009.
24. S. Hartmann and S. Link. The implication problem of data dependencies over SQL table definitions. *ACM Trans. Database Syst.*, 37(2):13, 2012.
25. A. Heise, Jorge-Arnulfo, Quiane-Ruiz, Z. Abedjan, A. Jentzsch, and F. Naumann. Scalable discovery of unique column combinations. *PVLDB*, 7(4):301–312, 2013.
26. I. Ileana, B. Cautis, A. Deutsch, and Y. Katsis. Complete yet practical search for minimal query reformulations under constraints. In *SIGMOD*, pages 1015–1026, 2014.
27. T. Imielinski and W. L. Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.
28. A. K. Jha, V. Rastogi, and D. Suciu. Query evaluation with soft keys. In *PODS*, pages 119–128, 2008.
29. H. Köhler, U. Leck, S. Link, and H. Prade. Logical foundations of possibilistic keys. In *JELIA*, pages 181–195, 2014.
30. H. Köhler and S. Link. Inclusion dependencies reloaded. In *CIKM*, pages 1361–1370, 2015.
31. H. Köhler and S. Link. SQL schema design. In *SIGMOD*, http://dx.doi.org/10.1145/2882903.2915239, 2016.
32. H. Köhler, S. Link, and X. Zhou. Possible and certain SQL keys. *PVLDB*, 8(11):1118–1129, 2015.
33. P. Koutris and J. Wijsen. The data complexity of consistent query answering for self-join-free conjunctive queries under primary key constraints. In *PODS*, pages 17–29, 2015.
34. M. Levene and G. Loizou. Axiomatisation of functional dependencies in incomplete relations. *Theor. Comput. Sci.*, 206(1-2):283–300, 1998.
35. M. Levene and G. Loizou. A generalisation of entity and referential integrity. *ITA*, 35(2):113–127, 2001.
36. Y. E. Lien. On the equivalence of database models. *J. ACM*, 29(2):333–362, 1982.
37. J. Liu, J. Li, C. Liu, and Y. Chen. Discover dependencies from data - A review. *IEEE TKDE*, 24(2):251–264, 2012.
38. H. Mannila and K.-J. Räihä. *Design of Relational Databases*. Addison-Wesley, 1992.
39. F. D. Marchi and J. Petit. Semantic sampling of existing databases through informative armstrong databases. *Inf. Syst.*, 32(3):446–457, 2007.
40. J. Melton. ISO/IEC 9075-2: 2003 (SQL/foundation). ISO standard, 2003.
41. M. Memari and S. Link. Index design for enforcing partial referential integrity efficiently. In *EDBT*, pages 217–228, 2015.
42. M. Memari, S. Link, and G. Dobbie. SQL data profiling of foreign keys. In *ER*, pages 229–243, 2015.
43. F. Naumann. Data profiling revisited. *SIGMOD Record*, 42(4):40–49, 2013.
44. New Zealand Ministry of Health. National health index questions and answers. `http://www.health.govt.nz/our-work/health-identity/national-health-index/nhi-information-health-consumers/national-health-index-questions-and-answers#duplicates`. Accessed: 1 March 2016.
45. R. Pochampally, A. D. Sarma, X. L. Dong, A. Meliou, and D. Srivastava. Fusing data with correlations. In *SIGMOD*, pages 433–444, 2014.
46. K. A. Ross, D. Srivastava, and S. Sudarshan. Materialized view maintenance and integrity constraint checking: Trading space for time. In *SIGMOD*, pages 447–458, 1996.
47. B. Saha and D. Srivastava. Data quality: The other face of big data. In *ICDE*, pages 1294–1297, 2014.
48. T. J. Schaefer. The complexity of satisfiability problems. In *STOC*, pages 216–226, 1978.
49. Y. Sismanis, P. Brown, P. J. Haas, and B. Reinwald. GORDIAN: Efficient and scalable discovery of composite keys. In *VLDB*, pages 691–702, 2006.
50. E. Sperner. Ein Satz über Untermengen einer endlichen Menge. *Math. Z.*, 27:544–548, 1928.
51. B. Thalheim. On semantic issues connected with keys in relational databases permitting null values. *Elektr. Informationsverarb. Kybern.*, 25(1/2):11–20, 1989.