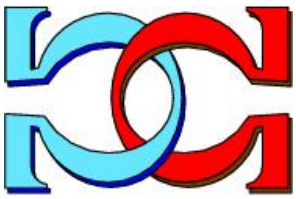
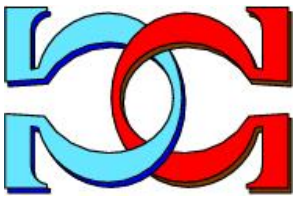
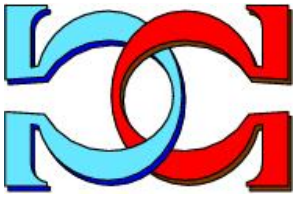
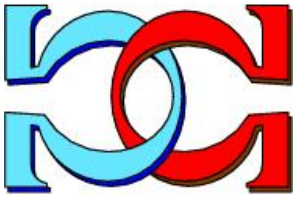


CDMTCS
Research
Report
Series





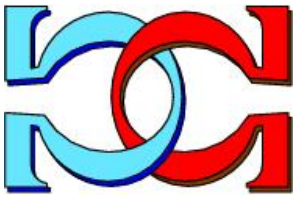
# Contextual Keys

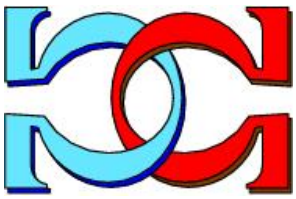


**Ziheng Wei**

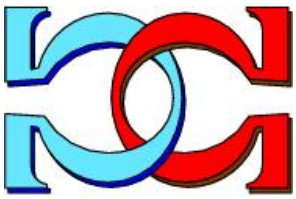
**Sebastian Link**

**Jiamou Liu**
The University of Auckland,
Auckland, New Zealand





CDMTCS-508
July 2017



Centre for Discrete Mathematics and
Theoretical Computer Science

# Contextual Keys

Ziheng Wei
The University of Auckland, New Zealand
z.wei@auchlanduni.ac.nz

Sebastian Link
The University of Auckland, New Zealand
s.link@auckland.ac.nz

Jiamou Liu
The University of Auckland, New Zealand
j.liu@auckland.ac.nz

July 4, 2017

**Abstract**

Much work has been done on extending the relational model of data to encompass incomplete information. In particular, a plethora of research has examined the semantics of integrity constraints in the presence of null markers. We propose a new approach whose semantics relies exclusively on fragments of complete data within an incomplete relation. For this purpose, we introduce the class of contextual keys. Users can specify the context of a key as a set of attributes that selects the sub-relation of tuples with no null marker occurrences on the attributes of the context. Then the key uniquely identifies the tuples within the sub-relation. The standard notion of a key over complete relations is the special case of a contextual key whose context consists of all attributes. SQL unique constraints form the special case of a contextual key whose context coincides with the set of key attributes. We establish structural and computational characterizations of the associated implication problem, and of their Armstrong databases. The computation of Armstrong databases has been implemented in a tool, and experiments provide insight into the actual run-time behavior of the algorithms that complement our detailed computational complexity analysis.

**Keywords:** Armstrong relation; Data and knowledge intelligence; Decision support; Incomplete data; Key; Reasoning; Requirements analysis

# 1 Introduction

Keys are core enablers 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, probabilistic models, XML, RDF, and graphs. They help 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 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 facilitate the data integration process, help with the detection of duplicates and anomalies, provide guidance in repairing data, and return consistent answers to queries over dirty data. The discovery of keys is one of the core activities in data profiling.

An important and rich area of research is to extend the relational model of data to encompass incomplete information. This is due to the importance of incomplete information for applications. A plethora of different extensions exist, but many are based on the use of a special symbol as placeholder for incomplete information, also known as the *null marker*. The semantics of the null marker can vary greatly, for example "unknown at present" [8], "non-existence" [33], "inapplicable" [4, 8], "no information" [38] and "open" [7]. Similarly, several extensions of the notion of a key from complete to incomplete relations have been investigated in the research literature. Examples constitute SQL's primary and candidate keys as well as `UNIQUE` constraints [19], weak and strong keys [28], Codd keys [11], possible and certain keys [19, 23, 24], and key sets [29, 35]. Candidate keys are minimal sets of attributes that enable us to uniquely identify tuples in an incomplete relation and where no null markers are permitted to occur in the columns of the key. They are a result of Codd's principle of entity integrity. The principle has been challenged by several researchers, including Thalheim [35], Levene and Loizou [29], and Köhler et al. [23]. For example, certain keys can uniquely identify rows in a table even though null markers may occur in the key columns. Similarly, for all pairs of distinct tuples there is some key in a key set on which the two tuples have no null occurrences and are unique.

Table 1 shows an incomplete relation, where $\perp$ denotes a null marker occurrence. Interestingly, this relation does not satisfy any candidate key, any certain key, nor any key set. It does not satisfy any candidate key as null markers occur in *Department* and *Manager*, and there are different tuples with the same value on *Employee*. The relation violates every certain key since the two null marker occurrences may be replaced by the values *Toys* and *Burns*, respectively, resulting in two different tuples that have matching values on all attributes. The relation violates every key set as the first and second tuple

Table 1: A relation with no candidate key, no certain key, and no key set

| *Employee* | *Department* | *Manager* |
|---|---|---|
| Homer | Toys | Burns |
| Homer | $\perp$ | $\perp$ |
| Marge | Toys | Burns |

are incomplete on *Department* and on *Manager*, and have matching values on *Employee*.

Common to all extended notions of keys is the target of uniquely identifying all tuples in incomplete relations, even tuples with null marker occurrences. The example in Table 1 shows that this target cannot always be achieved. In fact, any semantics of a key that depends on the interpretation of null markers can easily become problematic. This holds especially when data is integrated from different sources, which may rely on different interpretations of null markers. Interestingly, SQL's `UNIQUE` constraint enforces uniqueness only for those tuples of an incomplete relation that are complete on the attributes of the `UNIQUE` constraint. For example, the incomplete relation in Table 1 satisfies the unique constraints `UNIQUE`(*Employee, Department*) and `UNIQUE`(*Employee, Manager*), but violates the unique constraints `UNIQUE`(*Employee*) and `UNIQUE`(*Department, Manager*). This approach sparked our idea of giving up any false hope that tuples can be uniquely identified in the presence of null marker occurrences. In SQL's `UNIQUE` constraint this given set of attributes forms the unique constraint itself. That, however, is a requirement that should be relaxed, as the incomplete relation in Table 1 illustrates. In fact, `UNIQUE`(*Employee, Department*) can distinguish between the first and third tuple by the values on *Employee* already, and does not require values on *Department*. Note that uniqueness only holds for the tuples which are complete on *Employee* and *Department*, and uniqueness does not hold for the tuples that are complete on *Employee* only.

Motivated by these examples, we propose the new notion of *contextual keys* for incomplete relations. Contextual keys target the unique identification of those tuples in an incomplete relation that are complete on a user-specified set of attributes. Contextual keys consist of a pair of attributes $(C, K)$ such that $K \subseteq C$. The user-specified set of attributes $C$ is called the *context*, and selects the *scope* of the key, which is defined as the subset of tuples in a given incomplete relation that are complete on all the attributes of the context. The set $K$ of a contextual key $(C, K)$ is called the key and uniquely identifies tuples in the scope of the key. For example, ({*Employee, Department*},{*Employee*}) and ({*Employee, Manager*},{*Employee*}) are both contextual keys that are satisfied by the incomplete relation in Table 1. Here, both contexts {*Employee, Department*} and {*Employee, Manager*} have the same scope in the incomplete relation, which consists of the first and third tuple, and the key {*Employee*} uniquely identifies tuples in this scope. The incomplete relation in Table 1 does not satisfy any of the following contextual keys: ({*Employee*},{*Employee*)}, ({*Employee, Department*},{*Department*}) and ({*Employee, Manager*},{*Manager*}). In particular, SQL's constraint `UNIQUE`($X$) is satisfied by an incomplete relation if and only if the relation satisfies the contextual key $(X, X)$.

This paper introduces contextual keys, and provides first evidence that they exhibit good computational properties. Due to its importance in automating data management, we are interested in the axiomatic and algorithmic characterizations of the implication

problem associated with contextual keys. We also want to provide computational support for the acquisition of contextual keys that are meaningful for an application domain. We will now detail our contributions.

**Contributions.** Our contributions are at least threefold. Firstly, we propose a novel class of keys for incomplete databases, named contextual keys. Secondly, we characterize the implication problem of contextual keys by a finite axiomatization and by a linear-time algorithm. An immediate application of the algorithm is to compute a non-redundant set of contextual keys, thereby minimizing the overhead of enforcing contextual keys on relations. Thirdly, we investigate structural and computational properties of Armstrong relations for contextual keys, providing a computational tool that aids with the acquisition of contextual keys. While the problem of finding an Armstrong relation is precisely exponential, we show that our algorithm is conservative in its use of time and space, as the output Armstrong relation is guaranteed to have a number of tuples that is at most quadratic in the minimum number of tuples required. For transfer into practice, we have implemented our algorithm in a prototype system. Experiments with the prototype system complement our theoretical complexity analysis, and illustrate - on average - how quickly Armstrong relations for contextual keys can be computed, how many tuples our output contains, and how many null markers occur in the output. For example, for a fixed schema with 15 attributes, and a set of contextual keys with 100 attributes, our algorithm computes an Armstrong relation with 86 tuples and 200 null marker occurrences in about 10 seconds.

**Organization.** We discuss related work in Section 2. Our central notion of contextual keys is introduced in Section 3, where we also characterize the associated implication problem axiomatically and algorithmically. In Section 4 we investigate structural and computational properties of Armstrong relations. Finally, we conclude and sketch future work in Section 5.

# 2 Related Work

While about 100 different classes of data dependencies are known [36], keys arguably constitute the most import class among all of them. Keys have been studied in-depth on complete data [30, 36], and have been extended to most other data models, including nested [37], object-relational [17], XML [14, 13, 12], and models of uncertainty [2, 3, 5, 18].

The first section has already examined various proposals for notions of keys over incomplete relations. These include candidate keys that respect Codd's principle of entity integrity [11], possible and certain keys [19], weak and strong keys [28], as well as key sets [29, 35]. Contextual keys are different from all of these proposals in the sense that they do not target the unique identification of all tuples in an incomplete relation. The idea of contextual keys is to target the unique identification of only those tuples that are complete on a user-specified set of attributes. This idea generalizes SQL's `UNIQUE` constraint where the user-defined set coincides with the set of attributes on which the values are unique. We believe contextual keys are particularly useful in modern applications, such as data integration, where different occurrences of missing information may require different semantics. The semantics of contextual keys is clearly

defined as it does not depend on the semantics of null marker occurrences. Furthermore, contextual keys empower users to link their completeness requirements on the quality of their data with their uniqueness requirements.

In our research we investigate the same computational problems that have been studied for previous notions of keys. More specifically, we tackle the associated implication problem as well as the structural and computational properties of Armstrong relations. The importance of these problems is well-established in the literature and practice. Efficient solutions to the implication problem help address many data management problems, as listed in the introduction. Furthermore, Armstrong relations are useful for the acquisition of meaningful contextual keys, similar to the case of Armstrong relations for other classes of data dependencies [1, 26, 27, 34].

# 3 Fundamentals of Contextual Keys

Let $\mathfrak{A} = \{A_1, A_2, \cdots\}$ be a countable and infinite set of distinct symbols, called *attributes*. A *relation schema* is a finite, non-empty set of attributes, normally denoted as $R$. Each attribute $A \in \mathfrak{A}$ is associated with a domain $dom(A)$. We assume that the domain of every attribute contains a distinguished null marker, which we denote by $\perp$. This is for simplicity, and we emphasize that $\perp$ is not a domain value but a marker. In what follows we will use the relation schema $\textsc{Staff} = \{Employee, Department, Manager\}$ from Table 1 as a running example for illustrating our concepts and results. We refer to the attributes *Employee*, *Department*, *Manager* as $E$, $D$, and $M$, respectively.

A *tuple* $t$ over $R$ is a function which maps each $A \in R$ to a value in $dom(A)$, namely $t(A) \in dom(A)$. A *relation* $r$ over $R$ is a finite set of tuples over $R$. The size of the relation, denoted as $|r|$ is the number of tuples the relation contains. Let $X = \{A_1, A_2, \cdots, A_m\}$ be a set of attributes. For simplicity, we sometimes write $X$ as $A_1A_2\cdots A_m$, and the union of $X$ and another attribute set $Y$ as $XY$. Let $X \subseteq R$. For a tuple $t$ over $R$, we use the notation $t(X)$ to denote the projection of $t$ onto $X$. To stipulate completeness, we say a tuple $t$ over $R$ is *$X$-total* if and only if $t(A) \neq \perp$ for all $A \in X$. Furthermore, We use $r^X$ to denote $\{t \in r \mid t \textit{ is } X\text{-total}\}$.

A *constraint* of a class $\mathcal{C}$ is a statement which enforces semantic properties on a given collection of data. For instance, *keys* are a class of integrity constraints which stipulate that the identity of tuples is determined by the values on a given set of attributes. Let $\Sigma$ be a set of constraints over class $\mathcal{C}$. We use $|\Sigma|$ to denote the number of constraints in $\Sigma$, and $||\Sigma||$ to denote the total number of attributes in $\Sigma$. For a class $\mathcal{C}$ of integrity constraints, we are interested in the implication problem associated with $\mathcal{C}$. The $\mathcal{C}$-implication problem is to decide whether for an arbitrary given relation schema $R$, and an arbitrarily given set $\Sigma \cup \{\varphi\}$ of integrity constraints from class $\mathcal{C}$ on $R$, $\Sigma$ implies $\varphi$ (written as $\Sigma \models \varphi$), that is, whether every relation over $R$ that satisfies all the elements in $\Sigma$ also satisfies $\varphi$. Solutions to the implication problem provide users with a better understanding of the interaction of the integrity constraints, and algorithms can be developed that compute better representations of a set of integrity constraints. For example, a set $\Sigma'$ is said to be a *cover* of $\Sigma$ if and only if $\Sigma$ and $\Sigma'$ are satisfied by the same relations. A cover $\Sigma'$ is said to be *non-redundant* if and only if for all $\sigma \in \Sigma'$ it is

the case that $\Sigma' - \{\sigma\}$ does not imply $\sigma$. Being able to decide the implication problem, we can easily compute a non-redundant cover for $\Sigma$ by successively checking for all $\sigma \in \Sigma$ whether $\Sigma - \{\sigma\}$ implies $\sigma$ and removing $\sigma$ from $\Sigma$ whenever that is the case. In practice, checking whether all the constraints in a non-redundant cover are satisfied by a relation ensures that the overhead of constraint validation is minimized. Clearly, the more tuples in a relation, the more time we save when validating constraints with a non-redundant cover.

Next we formally introduce a new class of keys for incomplete relations which we call *contextual keys*.

**Definition 1** *A* contextual key (CK) *over a relation schema $R$ is a statement of the form $(C, K)$ where $K \subseteq C \subseteq R$. The attribute set $C$ is called the* context *of the contextual key. A relation $r$ over $R$* satisfies *the* CK *$(C, K)$, denoted as $r \models (C, K)$, if and only if for all $t, t' \in r^C$, $t(K) = t'(K)$ implies $t = t'$. We call $r^C$ the scope of $r$ with respect to the context $C$.*

Next we illustrate the notion of a contextual key on our running example.

**Example 1** *The incomplete relation $r$ in Table 1 satisfies the contextual keys $(ED, E)$ and $(EM, E)$, but it violates the contextual keys $(E, E)$, $(ED, D)$, and $(EM, M)$. For example, $r$ satisfies $(ED, E)$, since the scope of $r$ with respect to the context $ED$ consists of the first and third tuple of $r$, and the values of these tuples on $E$ are different. Similarly, $r$ does not satisfy $(E, E)$, since the scope of $r$ with respect to the context $E$ is the relation $r$ itself, but the values of the first and second tuples on $E$ are the same.*

For what follows, we require the following concepts. Let $R$ be a relation schema. We define a partial order $\sqsubseteq_R$ over $R$ as $\{((C_1, K_1), (C_2, K_2)) \mid C_1 \subseteq C_2 \subseteq R, K_1 \subseteq K_2 \subseteq R\}$. For any $((C_1, K_1), (C_2, K_2)) \in \sqsubseteq_R$, we write $(C_1, K_1) \sqsubseteq_R (C_2, K_2)$, or $(C_1, K_1) \sqsubset_R (C_2, K_2)$ if $C_1 \subset C_2$ or $K_1 \subset K_2$. We may omit the subscript $R$, if $R$ is clear from the context. Let $\Sigma$ be a set of CKs over $R$. We define the set $\mathsf{CL}(\Sigma) = \{(C, K) \mid \Sigma \not\models (C, K), K \subseteq C \subseteq R\}$. The set of *contextual anti-keys* of $\Sigma$ is $\Sigma^{-1} = \{(C, K) \in \mathsf{CL}(\Sigma) \mid \neg\exists (C', K') \in \mathsf{CL}(\Sigma) : (C, K) \sqsubset (C', K')\}$. Let $r$ be a relation over $R$. We say $t_1$ and $t_2$ *exactly agree* on $(C, K)$ if and only if $t_1(K) = t_2(K)$ and $t_1(A) = \perp \vee t_2(A) = \perp$ for all $A \in R \setminus C$. The *agree set* of $r$ is $ag(r) = \{(C, K) \mid t_1, t_2$ exactly agree on $(C, K)$ for all distinct $t_1, t_2 \in r\}$.

A sensible first step in solving the implication problem is to discover a set of *inference rules* that allows us to mechanically derive exactly those constraints from a given set $\Sigma$ that are implied. We apply inference rules of the form $\frac{\text{premises}}{\text{conclusion}}$. Let $\mathfrak{R}$ be a set of inference rules over class $\mathcal{C}$ and $\varphi$ a constraint of $\mathcal{C}$. We say that $\varphi$ is *derivable* from $\Sigma$ with respect to $\mathfrak{R}$, denoted by $\Sigma \vdash_{\mathfrak{R}} \varphi$, whenever there is some finite sequence $\sigma_1, \sigma_2, \ldots, \sigma_n$ such that $\sigma_n = \varphi$ and for every $i < n$, $\sigma_i \in \Sigma$ or $\sigma_i$ results from the conclusion of some inference rule in $\mathfrak{R}$ with $\sigma_1, \ldots, \sigma_{i-1}$ as premises.

To reason about CKs, we introduce the set $\mathfrak{B}$ of inference rules as shown in Table 2. Our goal is to show that $\mathfrak{B}$ is a sound and complete set of inference rules for contextual keys. This goal can be realized by using the following syntactic characterization of $\mathfrak{B}$.

**Theorem 1** *Let $\Sigma \cup \{(C, K)\}$ be a set of* CK*s over a relation schema $R$. $\Sigma \vdash_{\mathfrak{B}} (C, K)$ if only if there is $(C', K') \sqsubseteq (C, K)$ where $(C', K') \in \Sigma \cup \{(R, R)\}$.*

| | |
|---|---|
| $\dfrac{}{(R,R)}$ *(R-axiom)* | $\dfrac{(C,K)}{(CC',KK')}$ *(Superkey)* |

Table 2: Axiomatization $\mathfrak{B}$

**Proof** Suppose there is $(C',K') \sqsubseteq (C,K)$ where $(C',K') \in \Sigma \cup \{(R,R)\}$. For *sufficiency*, we construct a sequence $\sigma_1 = (C',K'), \sigma_2 = (C'X, K'Y)$ where $X = C \setminus C'$ and $Y = K \setminus K'$. Therefore, $\Sigma \vdash_{\mathfrak{B}} (C,K)$ because $\sigma_2$ results from $\sigma_1$ by applying the *superkey axiom*. For *necessity*, suppose $(C',K') \not\sqsubseteq (C,K)$ for all $(C',K') \in \Sigma \cup \{(R,R)\}$. Consequently, there is no way to apply any axiom in $\mathfrak{B}$ to $\Sigma \cup \{(R,R)\}$. Therefore, $\Sigma \nvdash_{\mathfrak{B}} (C,K)$. ∎

We will now illustrate the usefulness of Theorem 1 on our running example.

**Example 2** *Given the set* $\Sigma = \{(ED,E),(EM,E)\}$ *we can use Theorem 1 to conclude that there is a derivation of the contextual keys* $(ED,ED)$ *and* $(EM,EM)$ *from* $\Sigma$ *by* $\mathfrak{B}$ *since* $(ED,E) \sqsubseteq (ED,ED)$ *and* $(EM,E) \sqsubseteq (EM,EM)$ *hold. Similarly, we can use Theorem 1 to conclude that there is no derivation of the contextual key* $(E,E)$ *from* $\Sigma$ *by* $\mathfrak{B}$ *since neither* $(ED,E) \sqsubseteq (E,E)$ *nor* $(EM,E) \sqsubseteq (E,E)$ *hold.*

**Lemma 1** *Let* $r$ *be a relation over relation schema* $R$*. If* $U \subseteq V \subseteq R$*, then* $r^V \subseteq r^U$*.*

**Proof** Suppose $U \subseteq V \subseteq R$. Take any $t \in r^V$. For any $A \in V$, $t(A) \neq \perp$. Since $U \subseteq V$, $t(A) \neq \perp$ for any $A \in U$. Therefore, $t \in r^U$ and $r^V \subseteq r^U$. ∎

Using Theorem 1, we can establish the following axiomatic characterization.

**Theorem 2** $\mathfrak{B}$ *forms a finite axiomatization for the implication of* CKs*.*

**Proof** For *soundness*, suppose $\Sigma \vdash_{\mathfrak{B}} (C,K)$. By Theorem 1, there is $(C',K') \sqsubseteq (C,K)$ where $(C',K') \in \Sigma \cup \{(R,R)\}$. Take any relation $r$ over $R$ where $r \models \Sigma$. We know that any relation over $R$ satisfies $(R,R)$. Since $(C',K') \in \Sigma \cup \{(R,R)\}$ and $r \models \Sigma$, thus $r \models (C',K')$. Namely, for all $t_1, t_2 \in r^{C'}$, $t_1(K') = t_2(K')$ implies $t_1 = t_2$. Take any $t_1, t_2 \in r^C$. On one hand, since $C' \subseteq C$, by Lemma 1, $r^C \subseteq r^{C'}$ and $t_1, t_2 \in r^{C'}$ . On the other hand, since $K' \subseteq K$, $t_1(K) = t_2(K)$ implies $t_1(K') = t_2(K')$. Consequently, since $t_1(K) = t_2(K)$ implies $t_1(K') = t_2(K')$ and $t_1(K') = t_2(K')$ implies $t_1 = t_2$, by syllogism, hence $t_1(K) = t_2(K)$ implies $t_1 = t_2$. Therefore, $\Sigma \models (C,K)$.

For *completeness*, we prove $\Sigma \models (C,K)$ implies $\Sigma \vdash_{\mathfrak{B}} (C,K)$. Assume $\Sigma \nvdash_{\mathfrak{B}} (C,K)$. Next, we show a counterexample in Table 3.

Table 3: Counterexample relation $r$ over $R$.

| | $K$ | $C \setminus K$ | $R \setminus C$ |
|---|---|---|---|
| $t_1$ | $0 \ldots 0$ | $0 \ldots 0$ | $\perp \ldots \perp$ |
| $t_2$ | $0 \ldots 0$ | $1 \ldots 1$ | $\perp \ldots \perp$ |

Since $\Sigma \not\vdash_{\mathfrak{B}} (C, K)$, by Theorem 1, there is no $(C', K') \sqsubseteq (C, K)$ where $(C', K') \in \Sigma \cup \{(R, R)\}$. If $C' \subseteq C$, then $K' \not\subseteq K$ and $t_1(K) \neq t_2(K)$. If $C' \not\subseteq C$, then $r^{C'} = \emptyset$. In summary of both cases, $r \models \Sigma$. However, $r \not\models (C, K)$ and hence $\Sigma \not\models (C, K)$, which draws a contradiction. Therefore, $\Sigma \models (C, K)$ implies $\Sigma \vdash_{\mathfrak{B}} (C, K)$. ∎

We apply Theorem 2 to our running example.

**Example 3** *Recall that we inferred in Example 2 that we can use $\mathfrak{B}$ to derive the contextual keys $(ED, ED)$ and $(EM, EM)$ from $\Sigma = \{(ED, E), (EM, E)\}$, but we cannot use $\mathfrak{B}$ to derive the contextual key $(E, E)$ from $\Sigma$. Based on the soundness of $\mathfrak{B}$ we can further conclude that the contextual keys $(ED, ED)$ and $(EM, EM)$ are implied by $\Sigma$. Furthermore, based on the completeness of $\mathfrak{B}$ we can conclude that the contextual key $(E, E)$ is not implied by $\Sigma$.*

The axiomatization can be used to explicitly enumerate all contextual keys that are implied by a given set $\Sigma$. In practice, however, one may not be interested in all contextual keys that are implied, but only be interested whether a given contextual key $\varphi$ is implied by $\Sigma$. In such situation, an explicit enumeration of all implied constraints is inefficient and does not make good use of the additional input $\varphi$. For this purpose, our final contribution of this section is a linear-time algorithmic characterization of the implication problem associated with contextual keys. In fact, we use Theorem 1 to obtain this algorithm.

**Algorithm 1** Implication of contextual keys

```
1: INPUT: A set Σ ∪ {(C, K)} of contextual keys over relation schema R
2: OUTPUT: TRUE, if Σ ⊨ (C, K); FALSE, otherwise
3: CK ← FALSE;
4: for each (C', K') ∈ Σ ∪ {(R, R)} do
5:    if C' ⊆ C and K' ⊆ K then
6:       CK ← TRUE;
7: return CK
```

The correctness of Algorithm 1 follows from Theorem 1 and Theorem 2, and the linear-time complexity is easy to observe from the algorithm.

**Theorem 3** *Algorithm 1 decides the implication problem of contextual keys $\Sigma \models (C, K)$ in time $\mathcal{O}(||\Sigma \cup \{(R, R)\}||)$.*

**Proof** Assume that checking if an attribute belongs to a set only takes constant time. Time for checking step 5 is in $\mathcal{O}(|C'| + |K'|)$. Therefore, in total, Algorithm 1 runs in $\mathcal{O}(||\Sigma \cup \{(R, R)\}||)$. ∎

We conclude this section by a final example.

**Example 4** *Using $R = EDM$, $\Sigma = \{(ED, E), (EM, E)\}$ and $\varphi = (E, E)$ as input, Algorithm 1 returns FALSE, since there is no $(C', K') \in \Sigma \cup \{(R, R)\}$ such that $C' \subseteq \{E\}$ and $K' \subseteq \{E\}$.*

# 4 Armstrong Relations for Contextual Keys

Contextual keys can enforce important application semantics within a database system. However, a fundamental problem is to acquire those contextual keys that are meaningful in a given application domain. Database designers usually do not know the domain well and domain experts do not know database constraints. We will now establish computational support for overcoming the communication barrier between designers and experts. As illustrated in Figure 1, designers think in terms of an abstract set $\Sigma$ of contextual keys they perceive meaningful. For them to communicate their current perceived understanding to domain experts, we will establish an algorithm that computes from a given set $\Sigma$ a relation $r_\Sigma$ that perfectly represents $\Sigma$. That is, $r_\Sigma$ satisfies all contextual keys in $\Sigma$ and violates all contextual keys that are not implied by $\Sigma$. Relations with this property are known as *Armstrong relations* [6]. If designers currently perceive an actually meaningful contextual key as meaningless, then this contextual key will be violated in $r_\Sigma$. The point is that domain experts will easily notice this violation because the contextual key is meaningful. The experts can then alert the designers to this inconsistency with the application semantics, and the designer can include the meaningful contextual key in their set $\Sigma$. Such process can be repeated until both designers and experts are happy. The other direction, in which one provides computational support for identifying the set $\Sigma$ of contextual keys that hold in a given relation, is beyond the focus of this article. However, research into this direction is useful as the domain expert may want to change values in an Armstrong relation, or legacy data becomes available to the designers. For the remainder of this section, we will investigate computational and structural properties of Armstrong relations for contextual keys. We begin with the definition of Armstrong relations for contextual keys.

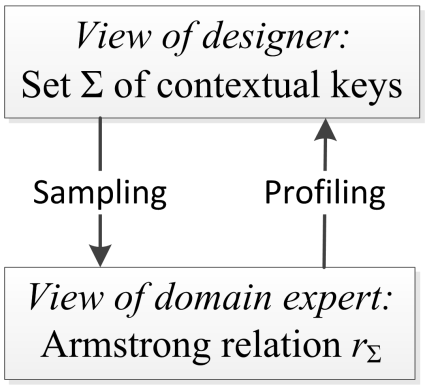


Figure 1: Acquisition Framework

**Definition 2** *Let $\Sigma$ be a set of contextual keys over relation schema $R$. A relation $r$ over $R$ is* Armstrong *for $\Sigma$ if and only if the following property holds for all contextual keys $(C, K)$ over $R$: $r \models (C, K)$ if and only if $\Sigma \models (C, K)$.*

Note the beauty of this definition: Given an Armstrong relation $r$ for $\Sigma$, one can reduce *every* instance $\Sigma \models \varphi$ of the implication problem for contextual keys to checking whether $\varphi$ holds on $r$. In fact, if $r$ satisfies $\varphi$, then $\Sigma$ implies $\varphi$; and if $r$ does not satisfy $\varphi$, then $r$ is not implied by $\Sigma$.

**Example 5** *The relation $r$ in Table 1 is Armstrong for the set*

$$\Sigma = \{(ED, E), (EM, E)\}$$

*over the relation schema $R = EDM$. It is indeed easy to observe that $r$ satisfies $(ED, ED)$ and $(EM, EM)$, which are therefore implied by $\Sigma$. Similarly, $r$ violates $(E, E)$, and $(EDM, DM)$ which are therefore not implied by $\Sigma$.*

We say that a class of constraints $\mathcal{C}$ *enjoys* Armstrong relations if there is an Armstrong relation for every given set of constraints of $\mathcal{C}$.

**Theorem 4** *Contextual keys enjoy Armstrong relations.*

**Proof** Let $\Sigma$ be a set of CKs over relation schema $R$. For all $K \subseteq C \subseteq R$, if $\Sigma \not\models (C, K)$, we construct a set $T_{(C,K)} = \{t_1, t_2\}$ where $t_1, t_2 \in r^C$ and $t_1(A_1) = t_2(A_1)$, $t_1(A_2) \neq t_(A_2)$, $t_1(A_3) = t_2(A_3) = \perp$ for all $A_1 \in K, A_2 \in C \setminus K, A_3 \in R \setminus C$. In addition, we use distinct non-null values for the construction $T$ of two distinct CKs. So, we claim $r = \bigcup_{\Sigma \not\models (C,K)} T_{(C,K)}$ is an Armstrong relation. Take any distinct $t_1, t_2 \in r$. If $t_1, t_2 \in T_{(C,K)}$ where $K \subseteq C \subseteq R$, then they form an example for $\Sigma \not\models (C, K)$; Otherwise, they will not show any violations to any CK $(C, K)$ where $\Sigma \models (C, K)$ because $t_1, t_2$ are constructed from different non-null values. ∎

Next we would like to characterize the structure of Armstrong relations for contextual keys. The following result establishes a necessary and sufficient condition for a given relation to be Armstrong for a given set of contextual keys.

**Theorem 5** *Let $\Sigma$ be set of contextual keys over relation schema $R$. A relation $r$ over $R$ is Armstrong for $\Sigma$ if and only if $\Sigma^{-1} \subseteq ag(r) \subseteq \mathsf{CL}(\Sigma)$.*

**Proof** $\Leftarrow$*:* Suppose $r$ is an Armstrong relation with respect to $\Sigma$. 1) Take any $(C, K) \in \Sigma^{-1}$. Assume $(C, K) \notin ag(r)$. Since $r$ is an Armstrong relation of $\Sigma$ and $(C, K) \in \Sigma^{-1}$, then $r \not\models (C, K)$. Consequently, by $(C, K) \in \Sigma^{-1}$ and $(C, K) \notin ag(r)$, there should be at least a pair of *distinct* $t_1, t_2 \in r$ such that $t_1$ and $t_2$ exactly agree on $(C', K')$ where $(C, K) \sqsubset (C', K')$. Namely, $t_1, t_2 \in r^{C'}$, $t_1(K') = t_2(K')$, and $t_1(C' \setminus K') = t_2(C' \setminus K')$. Since $(C, K)$ is a contextual anti-key, $(C', K')$ should not be a contextual anti-key as well because $(C, K) \sqsubset (C', K')$. However, $t_1, t_2$ cause a violation of $(C', K')$, which is a contradiction to $(C, K) \in \Sigma^{-1}$. Therefore, $(C, K) \in ag(r)$. 2) Take any $(C, K) \in ag(r)$. There are distinct $t_1, t_2 \in r$ such that $t_1, t_2 \in r^C$, $t_1(K) = t_2(K)$ and $t_1(C \setminus K) \neq t_2(C \setminus K)$. Since $t_1 \neq t_2$, $\Sigma \not\models (C, K)$. Therefore, $(C, K) \in \mathsf{CL}(\Sigma)$.

$\Rightarrow$*:* Let $\Sigma^{-1} \subseteq ag(r) \subseteq \mathsf{CL}(\Sigma)$. We show $r \models (C, K)$ if and only if $\Sigma \models (C, K)$ for all $K \subseteq C \subseteq R$. 1) For *sufficiency*, assume $\Sigma \not\models (C, K)$. Consequently, there exists a contextual anti-key $(C', K') \in \Sigma^{-1}$ where $(C, K) \sqsubseteq (C', K')$. Since $\Sigma^{-1} \subseteq ag(r)$, thus $(C', K') \in ag(r)$. So, there are distinct $t_1, t_2 \in r$ where $t_1, t_2 \in r^{C'}$ and $t_1(K') = t_2(K')$. Moreover, $t_1, t_2$ are $C$-total and $t_1(K) = t_2(K)$ because $(C, K) \sqsubseteq (C', K')$. However, since $t_1 \neq t_2$, $r \not\models (C, K)$, which is a contradiction. 2) For *necessity*, suppose $r \not\models (C, K)$. So, there exists distinct $t_1, t_2 \in r$ such that $t_1, t_2$ exactly agree on $(C', K')$ where $(C, K) \sqsubseteq (C', K')$. So, $(C', K') \in ag(r)$. Since $ag(r) \subseteq \mathsf{CL}(\Sigma)$, then $(C', K') \in \mathsf{CL}(\Sigma)$ and $\Sigma \not\models (C', K')$. Therefore ,$\Sigma \not\models (C, K)$ otherwise $\Sigma \models (C', K')$ by Theorem 1. ∎

We apply Theorem 5 to our running example.

**Example 6** *The relation $r$ in Table 1 is indeed Armstrong for*

$$\Sigma = \{(ED, E), (EM, E)\}$$

*because every contextual anti-key is an exact agree set of $r$, and every exact agree set is a contextual key not implied by $\Sigma$. In fact, the contextual anti-keys of $\Sigma$ are $(E, E)$ and $(EDM, DM)$, which are the exact agree sets of the first and second tuple, and the first and third tuple, respectively. Moreover, the exact agree set of the second and third tuple is $(EDM, \emptyset)$, which is not implied by $\Sigma$.*

Given Theorem 5, it is not difficult to see that Algorithm 2 computes a relation that is Armstrong for a given set $\Sigma$ of contextual keys.

**Algorithm 2** Computing Armstrong relations

1: **INPUT:** A set $\Sigma$ of contextual keys over a relation schema $R$
2: **OUTPUT:** An Armstrong relation $r$ for $\Sigma$
3: Let $t_0$ be a tuple over $R$ where $t(A) = 0$ for all $A \in R$;
4: $r_{\text{Armstrong}} \leftarrow \{t_0\}$;
5: $i \leftarrow 1$;
6: **for** each $(C, K) \in \Sigma^{-1}$ **do**
7: Let $t_i$ be a tuple over $R$
8: $t_i(A) \leftarrow \begin{cases} 0 & \text{, if } A \in K \\ i & \text{, if } A \in C \setminus K \\ \perp & \text{, if } A \in R \setminus C \end{cases}$
9: $i++$;
10: $r_{\text{Armstrong}} \leftarrow r_{\text{Armstrong}} \cup \{t_i\}$;
11: **return** $r_{\text{Armstrong}}$;

**Lemma 2** *Algorithm 2 computes an Armstrong relation of size $|\Sigma^{-1}| + 1$ given a set of $\mathsf{CK}s$ $\Sigma$ over relation schema $R$.*

**Proof** Suppose a relation $r$ is computed by Algorithm 2 given $\Sigma^{-1}$. For each $(C, K) \in \Sigma^{-1}$, there is $t_i \in r$ where $t_i(A_1) = 0, t_i(A_2) = i, t_i(A_3) = \perp$ for all $A_1 \in K, A_2 \in C \setminus K, A_3 \in R \setminus C$. Hence, $t_i$ and $t_0$ exactly agree on $(C, K)$ and $\Sigma^{-1} \subseteq ag(r)$. WLOG, assume there are $t_i, t_j$ where $i > j > 0$ such that $t_i, t_j$ exactly agree on $(C, K)$ and $\Sigma \models (C, K)$. According to our construction, the agreed value of two tuples is 0. So, $(C, K) \sqsubseteq (C', K')$ where $t_0, t_i$ exactly agree on $(C', K')$. Since $(C', K')$ is a contextual anti-key and $\Sigma \not\models (C', K')$, $(C, K)$ would also be an anti-key. That is, $\Sigma \not\models (C, K)$. However, it contradicts the assumption $\Sigma \models (C, K)$. Therefore, $\Sigma^{-1} \subseteq ag(r) \subseteq \mathsf{CL}(\Sigma)$. By Theorem 5, $r$ is of size $|\Sigma^{-1}| + 1$. ∎

In fact, Algorithm 2 computes Armstrong relations with conservative use of space.

**Theorem 6** *For every set $\Sigma$ of contextual keys, Algorithm 2 computes an Armstrong relation for $\Sigma$ with $|\Sigma^{-1}| + 1$ tuples. This number is at most the square of the minimum number of tuples required by any Armstrong relation for $\Sigma$.*

**Proof** Let $\Sigma$ be a set of CKs over $R$. By Lemma 2, $r$ is an Armstrong relation of $\Sigma$ generated by Algorithm 2. By Algorithm 2, $|r| = |\Sigma^{-1}|+1$. By Theorem 5, $\Sigma^{-1} \subseteq ag(r)$. The size of the agree set of $r$ is at most $\binom{|r|}{2}$, namely $|ag(r)| \leq \binom{|r|}{2}$. So, $|\Sigma^{-1}| \leq \binom{|r|}{2}$. Consequently, $|\Sigma^{-1}| \leq \frac{|r|\cdot(|r|-1)}{2}$ and $|r| \geq (1 + \sqrt{1 + 8 \cdot |\Sigma^{-1}|})/2$. So, the size $|r_{min}|$ of a minimum-sized Armstrong relation of $\Sigma$ is at least $(1 + \sqrt{1 + 8 \cdot |\Sigma^{-1}|})/2$. Therefore, $|r_{min}|^2 = (1 + \sqrt{1 + 8 \cdot |\Sigma^{-1}|} + 4 \cdot |\Sigma^{-1}|) \geq |r| = |\Sigma^{-1}| + 1$. ∎

In practice, it may be important to focus the attention of the designers and domain experts to certain fragments of an Armstrong relation. For rows, it makes sense to loop through the anti-keys and look at each row pair whose agree set is the anti-key. For columns, one may give the users of the algorithm full control over which columns should be highlighted. A sensible choice is to inspect the columns in the context of an anti-key.

The next example demonstrates the worst case in which the minimum number of tuples required by an Armstrong relation for $\Sigma$ is exponential in the input size.

**Example 7** *Let $R = \{A_1, A_2, \ldots, A_{2n}\}$ where $n$ is a positive integer. Let*

$$\Sigma = \{(A_{2i-1}A_{2i}, A_{2i-1}A_{2i}) \mid i = 1, \ldots, n\}$$

*be a set of* CK*s over $R$. If $n = 2$, then*

$$\Sigma^{-1} = \{(A_1A_3, A_1A_3), (A_1A_4, A_1A_4), (A_2A_3, A_2A_3), (A_2A_4, A_2A_4)\}$$

*In general, $|\Sigma^{-1}| = 2^n$ where $\Sigma$ has size $4n$ .* ∎

As evidenced by Example 7, there is no algorithm that computes Armstrong relations in polynomial time in the input. Extending the currently best known strategy of computing the set $\Sigma^{-1}$ of anti-keys from traditional to contextual keys [34], we establish a characterization of anti-keys that will lead us to an iterative algorithm to compute them.

**Lemma 3** *Let $\Gamma = \Sigma \cup \{(C, K)\}$ be a set of contextual keys over a relation schema $R$. If $(U, X) \in \Gamma^{-1}$, then the following must hold:*

1. *$(U, X) \in \Sigma^{-1}$, or*
2. *there exists $A \in K$ such that $(U, XA) \in \Sigma^{-1}$, or*
3. *there exists $A \in C \setminus K$ such that $(UA, X) \in \Sigma^{-1}$.*

**Proof** Let $\Sigma \cup \{(C, K)\}$ be a set of contextual keys over a relation schema $R$. $\Gamma = \Sigma \cup \{(C, K)\}$. Suppose $(U, X) \in \Gamma^{-1}$. Namely, there exists no $(C', K') \sqsubseteq (U, K)$ where $(C', K') \in \Gamma$. Since $\Sigma \subset \Gamma$, there also exists no $(C', K') \sqsubseteq (U, X)$ where $(C', K') \in \Sigma$. That is $\Sigma \not\models (U, X)$. Thus $(U, X)$ is possible to be in $\Sigma^{-1}$. If $(U, X) \in \Sigma^{-1}$, *conclusion 1* trivially holds. Next, suppose $(U, X) \notin \Sigma^{-1}$. Given $\Sigma \not\models (U, X)$, if $(U, X) \notin \Sigma^{-1}$, then there is $(U', X') \in \Sigma^{-1}$ where $(U, X) \sqsubseteq (U', X')$. Since $(U', X') \in \Sigma^{-1}$ and $(U, X) \in \Gamma$, then $(C, K) \subseteq (U', X')$

*Case $U \supseteq C$.* Firstly, $U' = U$ otherwise $(U', X)$ must be in $\Gamma^{-1}$ instead of $(U, X)$. Thus, $X \subset X'$. Secondly, since $(C, K) \not\sqsubseteq (U, X)$ and $U \supseteq C$, so $K \not\subseteq X$ and there is

$A \in K$ such that $XA = X'$. Thirdly, $(U, X' \setminus \{Y\}) \notin \Gamma^{-1}$ for all $Y \subseteq X$ where $|Y| \geq 2$ because $(U, X' \setminus \{Y\}) \sqsubseteq (U, X)$. Therefore, *conclusion 2* is proved.

*Case* $U \not\supseteq C$. Firstly, since $U \not\supseteq C$, there is $A \in C$ such that $UA = U'$. Note that for any $Y \subseteq C$ and $|Y| \geq 2$, $(U' \setminus Y, X) \notin \Gamma^{-1}$ because $(U' \setminus Y, X) \sqsubseteq (U, X)$. Secondly, since $U \not\supseteq C$, $X = X'$ otherwise if there exists $A \in K$, $(U' \setminus \{A\}, X' \setminus \{A\}) \sqsubset (U', X' \setminus \{A\})$ and it has been covered by the previous case. Therefore, *conclusion 3* is proved. ∎

Lemma 3 will give us an iterative algorithm for computing the anti-keys for a given set of contextual keys. However, in each iteration we still need to validate for each candidate anti-key that it is indeed an anti-key. This can be done efficiently as shown now.

**Lemma 4** *Validating whether a given contextual key is an anti-key for a given set* $\Sigma$ *of contextual keys over relation schema* $R$ *can be done in time* $\mathcal{O}(|R| \cdot ||\Sigma||)$.

**Proof** Suppose $\Sigma \not\models (C, K)$ where $K \subseteq C \subseteq R$. To check if $(C, K) \in \Sigma^{-1}$, we have to check if $(C, K)$ always becomes a CK by adding any $A \in R$. That is, $(C, K)$ is an anti-key if and only if $\forall A \in R \setminus C, \Sigma \models (CA, K)$ and $\forall A \in C \setminus K, \Sigma \models (C, KA)$ is true. For each $A \in R$, testing takes constant time. Afterwards, checking if a test case can be derived from a CK $(C, K)$ is in $\mathcal{O}(|C| + |K|)$. In total, examining all the CKs in $\Sigma$ is in $\mathcal{O}(||\Sigma||)$. Therefore, the time for testing if $(C, K)$ is a contextual anti-key is in $\mathcal{O}(|R| \cdot ||\Sigma||)$. ∎

Algorithm 3 iteratively examines the input keys in $\Sigma$. For each input key $(C, K)$ it checks if any anti-key in $\Sigma^{-1}$ contains $(C, K)$. The algorithm constructs $\Gamma^{-1} = (\Sigma \cup \{(C, K)\})^{-1}$ from $\Sigma^{-1}$ by eliminating attributes in $K$ or $C \setminus K$.

**Algorithm 3** Computing contextual anti-keys

```
1: INPUT: A set Σ of contextual keys over a relation schema R
2: OUTPUT: Σ^{-1}
3: Σ ← Σ ∪ {(R, R)}, Σ' = ∅;
4: Σ^{-1} ← {(R, R)};
5: for each (C, K) ∈ Σ do
6:    Σ' ← Σ' ∪ {(C, K)}
7:    for each (U, X) ∈ Σ^{-1} do
8:       if (C, K) ⊑ (U, X) then
9:          Σ^{-1} ← Σ^{-1} \ {(U, X)}
10:         for each A ∈ C \ K do
11:            Σ^{-1} ← Σ^{-1} ∪ {(U \ {A}, X \ {A})}
12:         for each A ∈ K do
13:            Σ^{-1} ← Σ^{-1} ∪ {(U, X \ {A})}
14:   for each (U, X) ∈ Σ^{-1} do
15:      if ∃A ∈ U \ X ∀(C', K') ∈ Σ' : (C', K') ⋢ (U, XA) or
         ∃A ∈ R \ U ∀(C', K') ∈ Σ' : (C', K') ⋢ (UA, X) then
16:         Σ^{-1} ← Σ^{-1} \ {(U, X)};
17: return Σ^{-1};
```

**Theorem 7** *Algorithm 3 computes* $\Sigma^{-1}$ *given a set of* CKs $\Sigma$ *over relation schema* $R$.

**Proof** We use induction on number of inputs in $\Sigma$ to show the loop at step 5 eventually generates $\Sigma^{-1}$. By the construction of Algorithm 3, an input set of CKs $\Sigma$ is not empty. Let $\Sigma_0 = \emptyset$ and $\Sigma_0^{-1} = \{(R, R)\}$. On the first input $(C_1, K_1) \in \Sigma$, obviously $(C_1, K_1) \sqsubseteq (R, R)$. So, $(R, R)$ cannot be a contextual anti-key of $\Sigma_1 = \Sigma_0 \cup \{(C_1, K_1)\}$. By Lemma 3, if there is $(C, K) \in \Sigma_1^{-1}$, then one can find some corresponding $(C', K') \in \Sigma_0^{-1}$. Since $(R, R)$ is the only one in $\Sigma_0^{-1}$, it could be used to compute contextual anti-keys in $\Sigma_1$ and $(R, R) \notin \Sigma_1^{-1}$ because $(C_1, K_1) \sqsubseteq (R, R)$. Let $\Gamma_1^{-1} = \{(R \setminus \{A\}, R \setminus \{A\}) \mid A \in C_1 \setminus K_1\} \cup \{(R, R \setminus \{A\}) \mid A \in K_1\}$. By such construction, $(C, K) \in \Gamma_1^{-1}$ for all $(C, K) \in \Sigma_1^{-1}$ because $(C, K) \not\sqsubseteq (R, R)$. In addition step 14 removes redundancy in $\Gamma_1^{-1}$. Therefore, the non-redundant cover of $\Gamma_1^{-1}$ is equal to $\Sigma_1^{-1}$. By the statements similar to above, we can show that the non-redundant cover of $\Sigma_i^{-1}$ is equal to $\Sigma_i^{-1}$ for all $1 \leq i \leq |\Sigma|$. Therefore, Algorithm 7 is correct. ∎

To evaluate the efficiency of our approach, we conducted experiments with Algorithm 2 and Algorithm 3. We randomly generated sets of CKs over a relation schema $R$. For each set $\Sigma$ of randomly generated CKs, we set a series of parameters: $n \in \{10, 20, \ldots, 100\}$, $k \in \{5, 6, \ldots, 15\}$ where $n = ||\Sigma||$ and $k = |R|$. In the experiments, we run each possible setting of the parameters 500 times and measure the average running time of the algorithm in milliseconds, the average number of tuples and null markers in the output Armstrong relation. Figure 2 illustrates that the average running time shows a linear growth with respect to the input size and a fixed schema. Similarly, Figure 3 illustrates that the size of Armstrong relations and the number of null marker occurrences grow slowly with increasing input size. Indeed, with smaller sizes and fewer occurrences of null markers, Armstrong relations become more comprehensible to domain experts. With faster run times, communication between designers and domain experts improves in terms of frequency and efficiency.

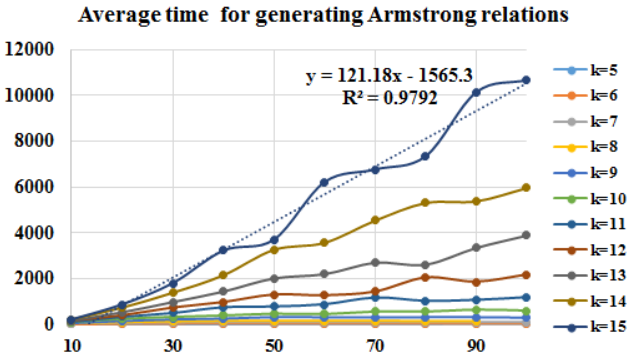



Figure 2: Average Computing Time

# 5 Conclusion and Future Work

We have investigated a new class of keys over incomplete relations, named contextual keys. Contextual keys target the unique identification of those tuples in a relation that are complete on a user-specified set of attributes. This approach ensures that the unique identification is independent of any interpretation of null marker occurrences. In order to unlock the vast usefulness of contextual keys for processing data, we have studied two fundamental problems associated with contextual keys. We have established axiomatic and algorithmic characterizations of the implication problem, enabling us to reason efficiently about contextual keys and to minimize the overhead of enforcing them within a

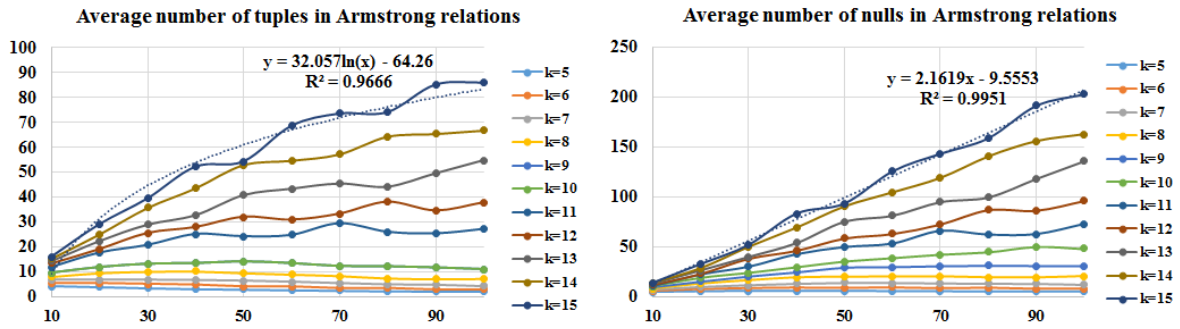



Figure 3: Average sizes and average number of null markers in Armstrong relations

database system. We have further established structural and computational properties of Armstrong relations for contextual keys, enabling us to represent any set of contextual keys in the form of a user-friendly data sample. Our theoretical and experimental analysis shows that Armstrong relations can be computed efficiently and that their size is reasonably small in order to be effective for the acquisition of contextual keys that are meaningful in a given application domain.

In the future, it will be interesting to investigate the discovery problem for contextual keys. The problem is to compute a cover of the set of contextual keys that hold in a given relation. Solutions to this problem will complete our acquisition framework, in which the cover of a set of constraints can be translated back and forth between an abstract set of constraints and an Armstrong relation for this set. Another important avenue of future research will lead to the investigation of other classes of contextual constraints, such as functional dependencies [10, 15], join dependencies [9, 16, 25, 31], or inclusion dependencies [22, 20]. This may have important applications in schema design [21, 32].

# References

[1] Bisbal, J., Grimson, J.: Consistent database sampling as a database prototyping approach. Journal of Software Maintenance and Evolution: Research and Practice 14(6), 447–459 (2002)

[2] Brown, P., Ganesan, J., Köhler, H., Link, S.: Keys with probabilistic intervals. In: Comyn-Wattiau, I., Tanaka, K., Song, I., Yamamoto, S., Saeki, M. (eds.) Conceptual Modeling - 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, Proceedings. LNCS, vol. 9974, pp. 164–179 (2016)

[3] Brown, P., Link, S.: Probabilistic keys. IEEE Trans. Knowl. Data Eng. 29(3), 670–682 (2017)

[4] Codd, E.F.: Missing information (applicable and inapplicable) in relational databases. ACM Sigmod Record 15(4), 53–53 (1986)

[5] Demetrovics, J., Katona, G.O.H., Miklós, D., Seleznjev, O., Thalheim, B.: Asymptotic properties of keys and functional dependencies in random databases. Theor. Comput. Sci. 190(2), 151–166 (1998)

[6] Fagin, R.: Horn clauses and database dependencies. Journal of the ACM (JACM) 29(4), 952–985 (1982)

[7] Gottlob, G., Zicari, R.: Closed world databases opened through null values. In: VLDB. vol. 88, pp. 50–61 (1988)

[8] Grant, J.: Null values in a relational data base. Information Processing Letters 6(5), 156–157 (1977)

[9] Hannula, M., Kontinen, J., Link, S.: On the finite and general implication problems of independence atoms and keys. J. Comput. Syst. Sci. 82(5), 856–877 (2016)

[10] Hartmann, S., Kirchberg, M., Link, S.: Design by example for SQL table definitions with functional dependencies. VLDB J. 21(1), 121–144 (2012)

[11] Hartmann, S., Leck, U., Link, S.: On Codd families of keys over incomplete relations. The Computer Journal 54(7), 1166–1180 (2011)

[12] Hartmann, S., Link, S.: Unlocking keys for XML trees. In: Schwentick, T., Suciu, D. (eds.) Database Theory - ICDT 2007, 11th International Conference, Barcelona, Spain, January 10-12, 2007, Proceedings. LNCS, vol. 4353, pp. 104–118. Springer (2007)

[13] Hartmann, S., Link, S.: Efficient reasoning about a robust XML key fragment. ACM Trans. Database Syst. 34(2), 10:1–10:33 (2009)

[14] Hartmann, S., Link, S.: Expressive, yet tractable XML keys. In: Kersten, M.L., Novikov, B., Teubner, J., Polutin, V., Manegold, S. (eds.) EDBT 2009, 12th International Conference on Extending Database Technology, Saint Petersburg, Russia, March 24-26, 2009, Proceedings. ACM International Conference Proceeding Series, vol. 360, pp. 357–367. ACM (2009)

[15] Hartmann, S., Link, S.: The implication problem of data dependencies over SQL table definitions: Axiomatic, algorithmic and logical characterizations. ACM Trans. Database Syst. 37(2), 13:1–13:40 (2012)

[16] Hartmann, S., Link, S., Schewe, K.: Reasoning about functional and multi-valued dependencies in the presence of lists. In: Seipel, D., Torres, J.M.T. (eds.) Foundations of Information and Knowledge Systems, Third International Symposium, FoIKS 2004, Wilhelminenberg Castle, Austria, February 17-20, 2004, Proceedings. Lecture Notes in Computer Science, vol. 2942, pp. 134–154. Springer (2004)

[17] Khizder, V.L., Weddell, G.E.: Reasoning about uniqueness constraints in object relational databases. IEEE Trans. Knowl. Data Eng. 15(5), 1295–1306 (2003)

[18] Köhler, H., Leck, U., Link, S., Prade, H.: Logical foundations of possibilistic keys. In: Fermé, E., Leite, J. (eds.) Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings. LNCS, vol. 8761, pp. 181–195. Springer (2014)

[19] Köhler, H., Leck, U., Link, S., Zhou, X.: Possible and certain keys for SQL. The VLDB Journal 25(4), 571–596 (2016)

[20] Köhler, H., Link, S.: Inclusion dependencies reloaded. In: Bailey, J., Moffat, A., Aggarwal, C.C., de Rijke, M., Kumar, R., Murdock, V., Sellis, T.K., Yu, J.X. (eds.) Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015. pp. 1361–1370. ACM (2015)

[21] Köhler, H., Link, S.: SQL schema design: Foundations, normal forms, and normalization. In: Özcan, F., Koutrika, G., Madden, S. (eds.) Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016. pp. 267–279. ACM (2016)

[22] Köhler, H., Link, S.: Inclusion dependencies and their interaction with functional dependencies in SQL. J. Comput. Syst. Sci. 85, 104–131 (2017)

[23] Köhler, H., Link, S., Zhou, X.: Possible and certain SQL keys. PVLDB 8(11), 1118–1129 (2015)

[24] Köhler, H., Link, S., Zhou, X.: Discovering meaningful certain keys from incomplete and inconsistent relations. IEEE Data Eng. Bull. 39(2), 21–37 (2016)

[25] Kontinen, J., Link, S., Väänänen, J.A.: Independence in database relations. In: Libkin, L., Kohlenbach, U., de Queiroz, R.J.G.B. (eds.) Logic, Language, Information, and Computation - 20th International Workshop, WoLLIC 2013, Darmstadt, Germany, August 20-23, 2013. Proceedings. LNCS, vol. 8071, pp. 179–193. Springer (2013)

[26] Langeveldt, W.D., Link, S.: Empirical evidence for the usefulness of armstrong relations in the acquisition of meaningful functional dependencies. Information Systems 35(3), 352–374 (2010)

[27] Le, V.B.T., Link, S., Ferrarotti, F.: Empirical evidence for the usefulness of armstrong tables in the acquisition of semantically meaningful SQL constraints. Data Knowl. Eng. 98, 74–103 (2015)

[28] Levene, M., Loizou, G.: Axiomatisation of functional dependencies in incomplete relations. Theoretical Computer Science 206(1), 283–300 (1998)

[29] Levene, M., Loizou, G.: A generalisation of entity and referential integrity in relational databases. RAIRO-Theoretical Informatics and Applications 35(2), 113–127 (2001)

[30] Levene, M., Loizou, G.: A guided tour of relational databases and beyond. Springer Science & Business Media (2012)

[31] Link, S.: Characterisations of multivalued dependency implication over undetermined universes. J. Comput. Syst. Sci. 78(4), 1026–1044 (2012)

[32] Link, S., Prade, H.: Relational database schema design for uncertain data. In: Mukhopadhyay, S., Zhai, C., Bertino, E., Crestani, F., Mostafa, J., Tang, J., Si, L., Zhou, X., Chang, Y., Li, Y., Sondhi, P. (eds.) Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016. pp. 1211–1220. ACM (2016)

[33] Makinouchi, A.: A consideration on normal form of not-necessarily-normalized relation in the relational data model. In: Proceedings of the Third International Conference on Very Large Data Bases, October 6-8, 1977, Tokyo, Japan. pp. 447–453 (1977)

[34] Mannila, H., Raihä, K.: Design by example: An application of Armstrong relations. Journal of computer and system sciences 33(2), 126–141 (1986)

[35] Thalheim, B.: On semantic issues connected with keys in relational databases permitting null values. Elektronische Informationsverarbeitung und Kybernetik 25(1/2), 11–20 (1989)

[36] Thalheim, B.: Dependencies in relational databases. Teubner (1991)

[37] Thalheim, B.: The number of keys in relational and nested relational databases. Discrete Applied Mathematics 40(2), 265–282 (1992)

[38] Zaniolo, C.: Database relations with null values. J. Comput. Syst. Sci. 28(1), 142–166 (1984)