

On Codd Families of Keys over Incomplete Relations

SVEN HARTMANN¹, UWE LECK² AND SEBASTIAN LINK^{3,*}

¹*Department of Informatics, Clausthal University of Technology, Clausthal-Zellerfeld, Germany*

²*Department of Mathematics and Computer Science, University of Wisconsin, Madison, WI, USA*

³*School of Information Management, Victoria University of Wellington, Wellington, New Zealand*

*Corresponding author: Sebastian.Link@vuw.ac.nz

Keys allow a database management system to uniquely identify tuples in a database. Consequently, the class of keys is of great significance for almost all data processing tasks. In the relational model of data, keys have received considerable interest and are well understood. However, for efficient means of data processing most commercial relational database systems deviate from the relational model. For example, tuples may contain only partial information in the sense that they contain so-called null values to represent incomplete information. Codd's principle of entity integrity says that every tuple of every relation must not contain a null value on any attribute of the primary key. Therefore, a key over partial relations enforces both uniqueness and totality of tuples on the attributes of the key. On the basis of these two requirements, we study the resulting class of keys over relations that permit occurrences of Zaniolo's null value 'no-information'. We show that the interaction of this class of keys is different from the interaction of the class of keys over total relations. We establish a finite ground axiomatization, and an algorithm for deciding the associated implication problem in linear time. Further, we characterize Armstrong relations for an arbitrarily given sets of keys; that is, we give a sufficient and necessary condition for a partial relation to satisfy a key precisely when it is implied by a given set of keys. We also establish an algorithm that computes an Armstrong relation for an arbitrarily given set of keys. While the problem of finding an Armstrong relation for a given key set is precisely exponential in general, our algorithm returns an Armstrong relation whose size is at most quadratic in the size of a minimal Armstrong relation. Finally, we settle various questions related to the maximal size of a family of non-redundant key sets. Our results help to bridge the gap between the existing theory of database constraints and database practice.

Keywords: relational database; incomplete relation; key; axiomatization; implication problem; Armstrong relation; extremal problem

Received 11 March 2010; revised 4 June 2010

Handling editor: Leonid Libkin

1. INTRODUCTION

A database system is a software package that manages a collection of persistent information in a shared, reliable, effective and efficient way. Most commercial database systems are still founded on Codd's *relational model of data* [1]. In this model, data is stored in a collection of relations that may vary over time. A relation is a set of tuples over a given time-invariant relation schema. The relation schema itself is a set of attributes that represent the properties that each tuple of each relation over the schema is described by; that is, a tuple maps every attribute of the relation schema to a value from the domain of this attribute. Relations permit the storage of inconsistent data,

i.e. data that violate conditions which every meaningful relation ought to satisfy. Consequently, additional assertions, called data dependencies, are specified by the data administrator in order to restrict the relations to those which are considered meaningful to the application at hand. The class of keys forms probably the most important class of data dependencies. Keys enable the database management system to uniquely identify every tuple in any relation over the relation schema. Consequently, keys express properties that are significant for almost every data processing task [2]. Moreover, 'good' database design is usually associated with the specification of keys only [3]. Formally, a key over a relation schema is an attribute subset

K that constrains instances over the schema to those relations that do not contain any two distinct tuples that agree on all the attributes in K . The majority of research on keys is based on total relations where the value of every tuple in every relation is known, i.e. not null. For total relations there is a single, well-understood, well-accepted and well-studied concept of a key [2, 4]. In practice, however, most relations are only partial.

EXAMPLE 1. Consider a simple database that collects basic information about the weekly schedule of courses. More precisely, we have a relation schema `SCHEDULE` with an attribute set $C_ID, L_Name, Time$ and $Room$. The schema stores the time (including the weekday) and room in which a lecturer (identified by their L_Name) gives a course (identified by their C_ID). An SQL definition of the table may look as follows:

```
CREATE TABLE SCHEDULE (
  C_ID CHAR[5],
  L_Name VARCHAR NOT NULL,
  Time CHAR[12],
  Room VARCHAR,
  PRIMARY KEY (C_ID, Time),
  UNIQUE (L_Name, Time));
```

The table definition specifies additional assertions. According to Codd's principle of entity integrity [5], the primary key forces tuples over `SCHEDULE` to be NOT NULL in the C_ID and $Time$ columns, and to be unique on their $\{C_ID, Time\}$ projections (no two distinct rows must have the same value in both the C_ID column and the $Time$ column). This ensures that every tuple can be uniquely identified. The uniqueness constraint on L_Name and $Time$ forces tuples to be unique on their $\{L_Name, Time\}$ projections. According to the SQL standard this uniqueness constraint can only be violated when there are (at least) two rows in the table where the values of each of the corresponding columns that are part of the constraint are equal and not null. Moreover, the NOT NULL constraint on L_Name excludes the occurrence of a null value in the L_Name column.

Among all keys the so-called minimal keys are of most interest in database practice. Minimal keys do not contain any proper subsets that are also keys, i.e. for every proper subset of a minimal key there is a meaningful relation (a possible future instance of the schema) with two distinct tuples that agree on all the attributes of the subset. Minimal keys enable the database management system to uniquely identify tuples with minimal effort required. Among all minimal keys there is one designated minimal key, the so-called *primary key*. A database management system utilizes the primary key as the primary mechanism to identify tuples in the relation. In Example 1, the attribute subset $\{C_ID, Time\}$ is a primary key for the relation schema `SCHEDULE`. Codd's principle of entity integrity [5] states that every attribute of the primary key must be NOT NULL. The principle addresses the unique identification of tuples in partial relations, which may contain null values. Consequently, we define a key over an underlying relation schema as an attribute

subset that forces all tuples in all relations over the schema to have unique projections on the key attribute subset and to be total on every key attribute. Following Zaniolo [6], Lien [7], Atzeni and Morfuni [8], we adopt the *no information* interpretation of null values, which we denote by `ni`. The following example illustrates that keys behave and interact quite differently over partial relations than they do over total relations.

EXAMPLE 2. The relation schema `SCHEDULE` implies the two minimal keys $\{C_ID, Time\}$ and $\{L_Name, Time\}$. In turn, these two keys together imply the key $\{C_ID, L_Name, Time\}$, but neither $\{C_ID, Time, Room\}$ nor $\{L_Name, Time, Room\}$ are implied keys since the partial relation

C_ID	L_Name	$Time$	$Room$
<i>DB201</i>	<i>Ullman</i>	<i>Mon, 11am</i>	<i>Cotton118</i>
<i>DB201</i>	<i>Ullman</i>	<i>Wed, 03pm</i>	<i>ni</i>

is not total on $Room$.

Example 2 shows that supersets of keys are not necessarily keys. This implies that it does not necessarily suffice to specify all minimal keys explicitly in order to capture all meaningful keys implicitly; that is, we can have non-redundant sets of keys that still contain one another (redundant in the sense that there is some key in the set that is implied by the remaining keys of the set). This situation is different from that in total relations. It is therefore our first objective to study the interaction of keys in detail. This includes problems such as the axiomatization and time complexity of the associated implication problem, but also questions related to the maximal size of non-redundant families of keys. Our results help us to decide which keys must be specified explicitly on a relation schema, which sets of keys are non-redundant, which sets are equivalent, and how complex the specification and maintenance of a relation schema may be.

Our second objective is the study of Armstrong relations in this context. Armstrong relations for a set of constraints are relations that satisfy each constraint implied by the set but no constraint that is not implied by it [9]. Armstrong relations are widely regarded as a helpful tool for design participants to judge, justify, convey or test their understanding of the relation schema [10–12]. We exemplify this situation by the following case.

EXAMPLE 3. Consider again the schema `SCHEDULE`. Suppose that so far in the design process the design participants have identified the following set Σ of keys: $\{C_ID, Time\}$ and $\{C_ID, Time, L_Name\}$. We can represent Σ concisely in the form of the following Armstrong relation:

C_ID	L_Name	$Time$	$Room$
<i>DB201</i>	<i>Ullman</i>	<i>Mon, 11am</i>	<i>ni</i>
<i>DB201</i>	<i>Ullman</i>	<i>Tue, 03pm</i>	<i>Cotton 118</i>
<i>CS100</i>	<i>Ullman</i>	<i>Tue, 03pm</i>	<i>MacKenzie 220</i>

Using the Armstrong relation, the design participants may notice that *Ullman* teaches not only *DB201* on *Tue* at *03pm*, but *CS100* as well. Moreover, they notice that *Ullman* is not only in *Room*

Cotton 118 on Tue at 03pm, but also in Room *MacKenzie 220*. After some deliberation, the designers decide to specify the key $\{L_Name, Time\}$ as a response. However, the set consisting of the three keys

$$\{C_ID, Time\}, \{L_Name, Time\}, \{C_ID, Time, L_Name\}$$

is redundant. In fact, $\{C_ID, Time, L_Name\}$ is implied by the two keys $\{C_ID, Time\}$ and $\{L_Name, Time\}$. Assuming that we can compute Armstrong relations and assuming that we can reason about keys efficiently, we have effective assistance in identifying meaningful keys and specifying non-redundant sets of keys.

Over total relations, properties of Armstrong databases have been well studied [9]. For partial relations, however, not even the existence of Armstrong relations has been investigated, except by Levene and Loizou [13] for a certain class of functional dependencies. Hence, there is a research gap in the study of Armstrong relations which we begin to address in this paper. We establish results and algorithms that can be implemented in design aids in the future, as was the case for functional dependencies over total relations [10]. The following example provides an illustration of the potential impact of our results on the efficiency of data processing tasks.

EXAMPLE 4. Continuing Example 3, our design team has finally decided to specify the key set Σ consisting of $\{C_ID, Time\}$ and $\{L_Name, Time\}$. A type of query that the database system must process for the underlying application domain is the following:

```
SELECT DISTINCT C_ID, L_Name, Time
FROM SCHEDULE
WHERE Room = "EA204" AND C_ID NOT NULL
AND L_Name NOT NULL AND Time NOT NULL .
```

However, since Σ implies the key $\{C_ID, L_Name, Time\}$, this query can be optimized as follows:

```
SELECT C_ID, L_Name, Time
FROM SCHEDULE
WHERE Room = "EA204" .
```

As duplicate elimination often requires an expensive sort of the query result [14], the results we develop in this article can help to identify automatically unnecessary DISTINCT clauses.

Contributions and organization. We comment on related work in Section 2, and summarize the notions of the underlying data model in Section 3. In Section 4, we introduce our class of keys over partial database relations. This class is a proper generalization of the class of keys over total relations [4], and is motivated by Codd's principle of entity integrity [5]. We show that these keys interact differently over partial relations than keys over total relations. Subsequently, we capture the implication of keys by means of a finite ground axiomatization, and by means of an algorithm that decides the implication

problem in time linear in the size of the input keys. Extending the notions of anti-keys and agree sets, we characterize Armstrong relations for our class of keys in Section 5. In Section 6, we establish an algorithm that computes an Armstrong relation with a number of tuples that is at most quadratic in the size of a minimal Armstrong relation. We also show that the time complexity of finding such Armstrong relations is precisely exponential in the number of attributes. In Section 7, we study extremal problems: we characterize all maximal non-redundant families of keys, even if the size of the given keys has some fixed upper bound.

Our contributions extend well-known results from total to partial relations. Hence, the resulting theory is closer to instances that occur in real database systems. Moreover, the theory still possesses the properties that have previously been established for total relations. The results also point to multiple directions of future work which we outline in Section 8.

2. RELATED WORK

Data dependencies have been studied thoroughly in various data models, and we refer the reader to more detailed surveys [2, 4, 15, 16].

We begin with a summary of the work over total relations. The concept of a key is almost as old as the relational model of data itself [1, 17]. A total relation over a relation schema R satisfies a key K if every pair of distinct tuples in the relation disagrees on some attribute of K . The concept of a key is well accepted and well understood for the relational model of data. In particular, the implication of keys can be axiomatized by two rules: the key axiom states that the full set of attributes of a relation schema is a key, and the superkey rule states that every superset of a key is also a key. In general, axiomatizations can be applied by database designers to enumerate all implied data dependencies. In practice, such an enumeration is often desirable, for example, to validate the correct specification of explicit knowledge, to design and fine-tune databases or to optimize queries. In particular, the completeness of the inference rules ensures that all opportunities of utilizing implicit knowledge for these purposes have been exploited. Furthermore, an analysis of the completeness argument can provide invaluable hints for finding algorithms that efficiently decide the associated implication problem. For keys over total relations, for example, the completeness of the two inference rules above implies that a key K is implied by a key set Σ over relation schema R if and only if $K = R$ or there is some key $K' \in \Sigma$ such that K is a superset of K' . Such decision algorithms complement the enumeration algorithm by a further reasoning capability that can make efficient, but only partial, decisions about implicit knowledge. These decisions are only partial in the sense that the input to this algorithm must also contain a candidate for an implied dependency. In contrast, the enumeration algorithm simply lists all implied dependencies. The reason for the prominence of the

implication problem is manifold. An algorithm for testing the implication of dependencies enables us to test whether two given sets of dependencies are equivalent or whether a given set of dependencies is redundant. A solution to these problems is a big step towards automated database schema design [18, 19], which some researchers see as the ultimate goal in dependency theory [20]. Moreover, such an algorithm can be used in relational normalization theory and practice involving many normal form proposals [17, 19–24], requirements engineering and schema validation [12], data mining [25], and in database security [26], view maintenance [27] and query optimization [28–30]. More recently, the implication problem has received a lot of attention in other data models as well [8, 13, 31–49]. New application areas involve consistency enforcement [50], data cleaning [51], data transformations [52], consistent query answering [53], data exchange [54–56] and data integration [57, 58].

Armstrong relations constitute an invaluable tool for the validation of semantic knowledge, and a user-friendly representation of integrity constraints. Armstrong relations have been deeply studied for keys [59–62] over total relations. They have also been analysed for various other classes of data dependencies [9, 12, 13, 63–71]. An excellent survey on Armstrong databases is [9]. Recently, the concept of *informative Armstrong databases* was introduced [68]. These are small subsets of an existing database, satisfying exactly the same data dependencies. Note that design aids [10, 12, 72] use Armstrong databases to help judge, justify, convey or test the database designer’s understanding of a given relation schema. There is empirical evidence for the usefulness of Armstrong relations in the acquisition of meaningful semantic constraints [11].

Extremal problems for families of non-redundant sets of keys have only been investigated in the absence of null values. In that context, the body of research is rather substantial and we only name a few [60, 61, 73–77]. Since database relations that occur in real database systems usually contain occurrences of null values, the study of extremal problems is particularly motivated in this context. Our results should be seen as a first step into that direction.

We now comment on some of the work concerning data dependencies in the presence of nulls. In the literature, many kinds of null values have been proposed; for example, ‘missing’ or ‘value unknown at present’ [78, 79], ‘non-existence’ [80], ‘inapplicable’ [78], ‘no information’ [6] and ‘open’ [81]. The most primitive is the ‘no information’ interpretation that can be used to model every kind of missing or incomplete information, and its semantics is certainly simple and well understood [8]. An occurrence of the no-information null value ni as a value for a tuple in a column A means that no information is available for that tuple on A ; that is, one does not even know whether some value exists (and is unknown) or whether no value exists. SQL uses this null value interpretation to accommodate all possibilities for modelling incomplete information.

Finally, Thalheim [82] discusses an alternative to Codd’s principle of entity integrity. It is argued that in certain situations

the principle may be unnecessarily strict. As an alternative, the notion of a *key set* is introduced. 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. We acknowledge that this is an interesting avenue to pursue, but the focus in this paper are the implications arising from Codd’s principle.

3. PRELIMINARIES

We use this section to summarize the basic notions required for our treatment of keys over partial database relations. Our development will follow the extension of Codd’s relational model of data [1] to encompass incomplete information by the ‘no information’ null value introduced by Zaniolo [6].

3.1. Total and partial relations

Let $\mathfrak{A} = \{A_1, A_2, \dots\}$ be a (countably) infinite set of distinct symbols, called attributes. A *relation schema* is a finite, non-empty subset R of \mathfrak{A} whose attributes represent column names of a relation. Each attribute A of a relation schema R is associated with an infinite domain $\text{dom}(A)$ which represents the set of possible values that can occur in the column named A . In order to encompass incomplete information, it is assumed that every attribute may have a null value, denoted by $\text{ni} \in \text{dom}(A)$. The intention of the null value ni is to mean ‘no information’.

If X and Y are sets of attributes, then we may write XY for $X \cup Y$. If $X = \{A_1, \dots, A_m\}$, then we may write $A_1 \cdots A_m$ for X . In particular, we may write simply A to represent the singleton $\{A\}$. A *tuple* over R (R -tuple or simply tuple, if R is understood) is a function $t : R \rightarrow \bigcup_{A \in R} \text{dom}(A)$ with $t(A) \in \text{dom}(A)$ for all $A \in R$. For $X \subseteq R$ let $t[X]$ denote the restriction of the tuple t over R to X , and let $\text{dom}(X) = \prod_{A \in X} \text{dom}(A)$ denote the Cartesian product of the domains of attributes in X . A (partial) *relation* r over R is a finite set of tuples over R . Let t_1 and t_2 be two tuples over R . It is said that t_1 *subsumes* t_2 if, for every attribute $A \in R$, $t_1[A] = t_2[A]$ or $t_2[A] = \text{ni}$ holds. In consistency with previous work [6–8, 83] on this subject, the following restriction will be imposed on the relations in a database: no relation in the database shall contain two tuples t_1 and t_2 such that t_1 subsumes t_2 . When no null value is present, this restriction amounts to saying that no two tuples are identical, an explicit requirement for database relations.

For a tuple $t \in R$ and a set $X \subseteq R$, t is said to be X -total if, for all $A \in X$, $t(A) \neq \text{ni}$. Similarly, a relation r over R is said to be X -total, if every tuple t of r is X -total. A relation r over R is said to be a *total relation* if it is R -total; that is, if, for all tuples $t \in r$ and all attributes $A \in R$, we have $t(A) \neq \text{ni}$.

3.2. Implication and inference

For a set Σ of constraints over some relation schema R , we say that a (partial) relation r over R *satisfies* Σ , denoted by $\models_r \Sigma$, if r satisfies every element of Σ . If, for some $\sigma \in \Sigma$, the relation

r does not satisfy σ , we sometimes say that r violates σ (in which case r also violates Σ) and write $\not\models_r \sigma$ ($\not\models_r \Sigma$).

In general, one may consider different classes \mathcal{C} of constraints over a single relation schema R . An example of such a class \mathcal{C} are keys, but one may also consider classes such as functional dependencies [63].

For the design of a relational database schema dependencies are normally specified as semantic constraints on the relations which are intended to be instances of the schema. During the design process or the lifetime of a database one usually needs to determine further dependencies which are logically implied by the given ones. In line with the literature of database constraints, we restrict our attention to the implication of constraints in some fixed class \mathcal{C} .

Let R be a relation schema and let $\Sigma \cup \{\varphi\}$ be a set of constraints over R in the class \mathcal{C} . We say that Σ (*finitely*) *implies* φ , denoted by $\Sigma \models_{(f)} \varphi$, if and only if every (finite) relation r over R that satisfies Σ also satisfies φ . The (*finite*) *implication problem for \mathcal{C}* is to decide, given any relation schema R and any set $\Sigma \cup \{\varphi\}$ of constraints in \mathcal{C} over R , whether $\Sigma \models_{(f)} \varphi$. If Σ does not (finitely) imply φ , we may also write $\Sigma \not\models_{(f)} \varphi$. Note that key sets $\Sigma \cup \{\varphi\}$ over any relation schema R are always finite. Moreover, finite and unrestricted implication problems coincide for the class of keys we consider. Hence, we commonly speak of the *implication problem* for the class of keys.

For a set Σ of constraints in \mathcal{C} over a relation schema R , let $\Sigma^* = \{\varphi \in \mathcal{C} \text{ over } R \mid \Sigma \models \varphi\}$ be its *semantic closure*, i.e. the set of all constraints in \mathcal{C} over R implied by Σ . The notion of syntactical inference ($\vdash_{\mathfrak{R}}$) with respect to a set \mathfrak{R} of inference rules can be defined as usual [2, pp. 164–168]; that is, a finite sequence $\gamma = [\sigma_1, \dots, \sigma_l]$ of constraints is called an *inference from Σ by \mathfrak{R}* if every σ_i is an element of Σ or is obtained by applying one of the rules of \mathfrak{R} to appropriate elements of $\{\sigma_1, \dots, \sigma_{i-1}\}$. We say that the inference γ infers σ_l , i.e. the last element of the sequence γ , and write $\Sigma \vdash_{\mathfrak{R}} \sigma_l$. For a finite set Σ of constraints in \mathcal{C} , let $\Sigma_{\mathfrak{R}}^+ = \{\varphi \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$ be its *syntactic closure* under inferences by \mathfrak{R} . A set \mathfrak{R} of inference rules is said to be *sound (complete)* for the implication of constraints in \mathcal{C} if, for every relation schema R and for every set Σ of constraints in \mathcal{C} over R , we have $\Sigma_{\mathfrak{R}}^+ \subseteq \Sigma^*$ ($\Sigma^* \subseteq \Sigma_{\mathfrak{R}}^+$). The set \mathfrak{R} is said to be an *axiomatization* for the implication of constraints in \mathcal{C} if \mathfrak{R} is both sound and complete for the implication of constraints in \mathcal{C} . Finally, an axiomatization \mathfrak{R} is said to be *finite* if \mathfrak{R} is finite.

3.3. Armstrong relations

Let Σ be a set of constraints in class \mathcal{C} over some relation schema R . A (partial) relation r over R is said to be an *Armstrong relation for Σ* , if, for all constraints φ in \mathcal{C} over R , it is true that r satisfies φ if and only if φ is implied by Σ . Hence, r is a perfect representation of the constraint set Σ in the sense that it satisfies all the constraints in Σ (and its implications), but violates all the constraints not implied by Σ .

A class \mathcal{C} of constraints is said to *enjoy Armstrong relations* if and only if, for every relation schema R and for every set Σ of constraints in \mathcal{C} over R , there is some (partial) relation r over R that is an Armstrong relation for Σ .

4. DESIGNATED TOOLS FOR REASONING

In this section, we introduce the class of *keys* in the context of partial relations. The definition of keys is derived from Codd's principle of entity integrity. Subsequently, we establish a finite axiomatization, and a linear-time algorithm to decide the associated implication problem.

4.1. Keys over partial relations

The issue of missing information in relational databases has received considerable attention in the database literature. It was Codd who formulated the principle of entity integrity: for the primary key K of any relation schema R , partial relations over R must be K -total. The principle is derived from the desire to identify every object that is stored in the database. For the purpose of this article, we are interested in the class of keys that meet this principle, i.e. keys that enforce uniqueness and totality.

DEFINITION 4.1. *Let R be some relation schema. A key over R is a subset $K \subseteq R$. A relation r over the relation schema R is said to satisfy the key K over R , denoted by $\models_r K$, if and only if r is K -total and for all $t, t' \in r$ the following holds: if $t \neq t'$, then there is some $A \in K$ such that $t(A) \neq t'(A)$.*

Note that Definition 4.1 is a proper generalization of the definition of keys in the context of total relations in which every pair of distinct tuples is required to disagree on some attribute of the key. Indeed, a total relation satisfies Definition 4.1 if and only if it satisfies the standard definition of a key.

Note that a relation r satisfies the key \emptyset over R precisely when r contains at most one tuple. We call the key \emptyset the *empty key*. A key is said to be *standard* if it is not the empty key.

Finally, note that Definition 4.1 allows us to introduce further standard terminology used in database theory and practice.

DEFINITION 4.2. *Let R be some relation schema and let Σ be a set of keys over R . The key K is said to be a candidate key for R with respect to Σ if and only if the following two properties are satisfied: (i) K is implied by Σ , and (ii) for all K' implied by Σ it follows that K' is not a proper subset of K . The primary key for R with respect to Σ is a designated candidate key for R with respect to Σ .*

EXAMPLE 5. The partial relation

C_ID	L_Name	Room	Time
DB201	Ullman	Cotton118	Mon, 11am
DB201	Ullman	ni	Wed, 03pm
COMP101	Beeri	MacKenzie255	Wed, 03pm

TABLE 1. Axiomatizations of keys over total relations, and over relations.

$\frac{}{\overline{R}}$ <p>(<i>R</i>-axiom)</p> $\frac{X}{XY}$ <p>(superkey)</p> <p><i>Key axiomatization over total relations</i></p>	$\frac{X; YZ}{XY}$ <p>(key extension)</p> <p><i>Key axiomatization over relations</i></p>
--	---

does not satisfy the key $\{C_ID, Room, Time\}$ since it is not $\{C_ID, Room, Time\}$ -total. However, the partial relation does satisfy the key $\{C_ID, Time\}$. In fact, $\{C_ID, Time\}$ appears to be a good candidate for the primary key of SCHEDULE. In particular, the relation above is a meaningful relation with distinct tuples that agree on C_ID , and distinct tuples that agree on time.

4.2. Axiomatization

Table 1 shows the well-known axiomatization for the implication of keys over total relations. For partial relations, however, none of the inference rules in this axiomatization is sound.

PROPOSITION 4.1. *The R -axiom and superkey rule are both not sound for the implication of keys over partial R -relations.*

Proof. Example 5 suffices to see this: the partial relation is not total, and hence, the R -axiom is not sound; and the partial relation satisfies the key $X = \{C_ID, Time\}$, but it violates the key XY where $Y = \{Room\}$. \square

Since relations that occur in commercial database systems are usually partial, and since keys interact differently over partial relations than they do over total relations, the problem of establishing an axiomatization for the class of keys over partial relations is of practical interest. We will show that the set \mathcal{K} consisting of the *key extension* rule in Table 1 is a finite axiomatization for the class of keys over partial relations.

We begin with some simple observations that we shall utilize quite frequently in the remainder of the article.

PROPOSITION 4.2. *Let Σ be a set of keys over the relation schema R . For every positive integer n we have: if $K_1, \dots, K_n \in \Sigma_{\mathcal{K}}^+$, then $K_1 \cup \dots \cup K_n \in \Sigma_{\mathcal{K}}^+$.*

Proof. This follows by a simple induction over n . For $n = 1$, there is nothing to show. If $K_1, \dots, K_{n+1} \in \Sigma_{\mathcal{K}}^+$, then the induction hypothesis tells us that $K_1 \cup \dots \cup K_n \in \Sigma_{\mathcal{K}}^+$. A single application of the key extension rule to K_{n+1} and $K_1 \cup \dots \cup K_n$ shows that $K_1 \cup \dots \cup K_{n+1} \in \Sigma_{\mathcal{K}}^+$. \square

DEFINITION 4.3. *For a set Σ of keys over a relation schema R , let $total(\Sigma) := \{A \in R \mid \exists Y \in \Sigma (A \in Y)\}$ denote the total attributes of Σ , i.e. those attributes of R that are elements of some key in Σ .*

COROLLARY 4.1. *Let Σ be a set of keys over the relation schema R . If $\Sigma \neq \emptyset$, then $\Sigma \vdash_{\mathcal{K}} total(\Sigma)$.*

Proof. This follows immediately from Proposition 4.2 since $total(\Sigma)$ is the union of keys from Σ . \square

Note that Corollary 4.1 does not hold when $\Sigma = \emptyset$: the empty key is not trivial, i.e. it does not follow from the empty set of keys.

PROPOSITION 4.3. *Let Σ be a set of keys over the relation schema R and let $K \in \Sigma$. For all $K' \subseteq total(\Sigma)$, such that $K \subseteq K'$, it follows that $\Sigma \vdash_{\mathcal{K}} K'$.*

Proof. Since $\Sigma \neq \emptyset$, we conclude that $K' \cup total(\Sigma) = total(\Sigma) \in \Sigma_{\mathcal{K}}^+$ by Corollary 4.1. An application of the key extension rule to K and $K' \cup total(\Sigma)$ results in $K' = K K' \in \Sigma_{\mathcal{K}}^+$. \square

We can now establish our first main result.

THEOREM 4.1. *The set \mathcal{K} forms a finite ground axiomatization for the implication of keys over relations.*

Proof. Soundness. Let R denote some relation schema, and r some partial relation over R that violates the key XY over R . Then there are two distinct tuples $t, t' \in r$ such that $t[XY] = t'[XY]$, or r is not XY -total. If r violates the key X , then there is nothing to show. Therefore, it remains to consider the case where r satisfies the key X over R . In this case, $t[X] \neq t'[X]$ and it therefore follows that r is not Y -total. Consequently, r violates the key YZ . This shows the soundness of \mathcal{K} for the implication of keys over partial relations.

Completeness. Let R denote some relation schema, $\Sigma \cup \{K\}$ a set of keys over R such that $K \notin \Sigma_{\mathcal{K}}^+$. The completeness of \mathcal{K} will follow, if we show that $K \notin \Sigma^*$. Therefore, we establish some partial relation r over R that satisfies Σ and violates K .

We distinguish between four cases.

Case 1. Suppose $\emptyset \in \Sigma$. Then let r denote the relation

$$\frac{total(\Sigma) \mid R - total(\Sigma)}{0 \dots 0 \mid ni \dots ni}$$

over R . Let X denote a key over R that is in Σ . It follows from the definition of $total(\Sigma)$ that $X \subseteq total(\Sigma)$. Consequently, r satisfies X , and since X was arbitrary, r satisfies Σ .

Since $\Sigma \neq \emptyset$, we conclude by Corollary 4.1 that $\text{total}(\Sigma) \in \Sigma_{\mathcal{K}}^+$. Assume that $K \subseteq \text{total}(\Sigma)$. Since $\emptyset \in \Sigma$, Proposition 4.3 would imply that $K \in \Sigma_{\mathcal{K}}^+$. This however, is a contradiction to the fact that $K \notin \Sigma_{\mathcal{K}}^+$. We conclude that $K \cap (R - \text{total}(\Sigma)) \neq \emptyset$. Therefore, by definition of r , it follows that r violates K .

Case 2. Suppose $R - (K - \text{total}(\Sigma)) = \emptyset$. Since $\text{total}(\Sigma) \subseteq R$ it follows that $K = R$ and $\text{total}(\Sigma) = \emptyset$. If $\Sigma = \{\emptyset\}$, then we are in Case 1. Otherwise, $\Sigma = \emptyset$. Let A denote some attribute of R . Then the relation

A	$R - A$
ni	$0 \dots 0$

violates the key $K = R$ since it is not R -total; that is, r satisfies $\Sigma = \emptyset$ and violates $K = R$.

Case 3. Let $K - \text{total}(\Sigma) \neq \emptyset$, and let $R - (K - \text{total}(\Sigma)) \neq \emptyset$. If $\emptyset \in \Sigma$, then we are in Case 1. Hence, let $\emptyset \notin \Sigma$. Let r denote the partial relation

$K - \text{total}(\Sigma)$	$R - (K - \text{total}(\Sigma))$
ni \dots ni	$1 \dots 1$
$0 \dots 0$	$0 \dots 0$

over R . Note that r is subsumption-free since $R - (K - \text{total}(\Sigma)) \neq \emptyset$. Since $K - \text{total}(\Sigma) \neq \emptyset$, it follows that r is not K -total. Consequently, r violates K . It remains to show that r satisfies Σ . Let X denote a key over R that is in Σ . Since $X \subseteq \text{total}(\Sigma)$ and $X \neq \emptyset$, it follows that $X \cap (K - \text{total}(\Sigma)) = \emptyset$. Consequently, r satisfies X . We have shown that r satisfies Σ and violates K .

Case 4. Let $K - \text{total}(\Sigma) = \emptyset$, i.e. $K \subseteq \text{total}(\Sigma)$. If $\emptyset \in \Sigma$, then we are in Case 1. Hence, let $\emptyset \notin \Sigma$. Let r denote the relation

K	$R - K$
$0 \dots 0$	$0 \dots 0$
$0 \dots 0$	$1 \dots 1$

over R . Suppose that $R - K = \emptyset$. Then it is easy to see that $K = R = \text{total}(\Sigma)$. Since $R \neq \emptyset$ we conclude that $\Sigma \neq \emptyset$. Hence, $K = \text{total}(\Sigma) \in \Sigma_{\mathcal{K}}^+$ by Corollary 4.1. This, however, is a contradiction to the fact that $K \notin \Sigma_{\mathcal{K}}^+$. Consequently, $R - K \neq \emptyset$, and r consists of two distinct tuples.

If $K = \emptyset$, then r violates K since r contains two distinct tuples. If $K \neq \emptyset$, then r violates K since the two distinct tuples of r match on all attributes of K by construction of r . It remains to show that r satisfies Σ . Assume that r violates some key X in Σ . It follows from the construction of r that $X \subseteq K$ holds. Since $X \in \Sigma$ and $K \subseteq \text{total}(\Sigma)$, Proposition 4.3 shows that $K \in \Sigma_{\mathcal{K}}^+$, too. This, however, contradicts the fact that $K \notin \Sigma_{\mathcal{K}}^+$. Consequently, r satisfies X . We have shown that r satisfies Σ and violates K . \square

EXAMPLE 6. Consider the set Σ consisting of the two keys $\{C_ID, Time\}$ and $\{Room, Time\}$. The closure $\Sigma_{\mathcal{K}}^+$ of Σ under inferences using \mathcal{K} consists of the keys in Σ , and the key $\{C_ID, Room, Time\}$.

4.3. Implication problem

As it has been often the case in database theory for other classes of integrity constraints, the axiomatization of keys provides us with sufficient clues towards establishing an algorithm that decides the associated implication problem. In fact, the soundness and completeness of the key extension rule indicates that a key K is implied by a set Σ of keys if and only if K is a superset of some key in Σ and every attribute of K is also an attribute of some key in Σ .

THEOREM 4.2. *Let $\Sigma \cup \{K\}$ denote a set of keys over the relation schema R . Then $\Sigma \models K$ if and only if $K \subseteq \text{total}(\Sigma)$ and there is some $K' \in \Sigma$ such that $K' \subseteq K$.*

Proof. Sufficiency. If $K \subseteq \text{total}(\Sigma)$ and $K' \subseteq K$ for some $K' \in \Sigma$, then Proposition 4.3 shows that $K \in \Sigma_{\mathcal{K}}^+$. The soundness of \mathcal{K} means that K is implied by Σ .

Necessity. Suppose that, for all $K' \in \Sigma$, we have that $K' \cap (R - K) \neq \emptyset$. Then it is easy to see that the relation

K	$R - K$
$0 \dots 0$	$0 \dots 0$
$0 \dots 0$	$1 \dots 1$

over R satisfies Σ and violates K . Consequently, K is not implied by Σ .

Suppose now that $K \not\subseteq \text{total}(\Sigma)$, i.e. $K - \text{total}(\Sigma) \neq \emptyset$. Then Cases 1–3 in the proof of Theorem 4.1 show that there is some relation that satisfies Σ and violates K . Consequently, K is not implied by Σ . \square

The following algorithm is based on the characterization of key implication in Theorem 4.2.

Algorithm 4.1 (Implication of keys).

Input: (Σ, K) with a key set $\Sigma \cup \{K\}$ over some relation schema R

Output: YES, if $\Sigma \models K$, and NO, otherwise

Method:

- (A1) $\text{total}(\Sigma) := \emptyset$;
- (A2) $\text{Key} := \text{FALSE}$;
- (A3) if $\emptyset \in \Sigma$, then $\text{Key} := \text{TRUE}$;
- (A4) for all $K' \in \Sigma - \{\emptyset\}$
 - if $K' \cap K \neq \emptyset$, then
 - $\text{total}(\Sigma) := \text{total}(\Sigma) \cup \{K'\}$;
 - if $K' \subseteq K$, then $\text{Key} := \text{TRUE}$;
- endfor
- (A5) if $K \subseteq \text{total}(\Sigma)$ and Key , then
 - return YES else return NO

EXAMPLE 7. Consider the set Σ consisting of the two keys $\{C_ID, Time\}$ and $\{Room, Time\}$. Further, let K_1 denote the key $\{C_ID, L_Name\}$ and K_2 denote the key $\{C_ID, Room, Time\}$. On input (Σ, K_1) , Algorithm 4.1 will return NO, and on input (Σ, K_2) , Algorithm 4.1 will return YES.

THEOREM 4.3. *Algorithm 4.1 decides the implication problem $\Sigma \models K$ for the class of keys in $\mathcal{O}(|\Sigma \cup \{K\}|)$ time.*

Proof. The soundness of Algorithm 4.1 follows from Theorem 4.2. The upper time bound follows straight from step (A4) in which every occurrence of a key in Σ is only considered once. \square

5. DEPICTING ARMSTRONG RELATIONS

We now establish necessary and sufficient conditions for a relation to be Armstrong for an arbitrarily given set of keys. This extends results on the characterization of Armstrong relations for keys over total relations. We begin with the special case where the empty key is part of the given set of keys.

LEMMA 5.1. *Let R be some relation schema and let Σ be a set of keys over R such that $\emptyset \in \Sigma$. A non-empty relation r over R is an Armstrong relation for Σ if and only if r consists of a single tuple t such that, for all $A \in R$, we have $t[A] \neq \text{ni}$ if and only if $A \in \text{total}(\Sigma)$.*

Proof. Sufficiency. Let r be defined as in the lemma. For all keys K over R it is true that r satisfies K if and only if $K \subseteq \text{total}(\Sigma)$. According to Theorem 4.2, and since $\emptyset \in \Sigma$, it holds that $K \subseteq \text{total}(\Sigma)$ if and only if $K \in \Sigma^*$. Therefore, r is an Armstrong relation for Σ .

Necessity. Let r be a non-empty relation over R , and an Armstrong relation for Σ . Since r satisfies $\emptyset \in \Sigma$, it follows that r contains precisely one tuple t . Since the key $\text{total}(\Sigma)$ is implied by Σ , r must be total on any attribute $\text{total}(\Sigma)$. According to Theorem 4.2, no attribute $A \in R - \text{total}(\Sigma)$ is a key implied by Σ . Consequently, r must violate A ; that is, r must be partial on A . We have just shown that, for every $A \in R$, it is true that $t(A) \neq \text{ni}$ if and only if $A \in \text{total}(\Sigma)$. \square

Consequently, Lemma 5.1 states that a total non-empty relation r is Armstrong for a key set Σ that contains the empty key \emptyset if and only if r is a singleton. For the purpose of characterizing Armstrong relations in the general case, we need to introduce some more terminology.

DEFINITION 5.1. *Let Σ be a set of keys over relation schema R . The set*

$$\Sigma^{-1} := \{X \subseteq \text{total}(\Sigma) \mid \Sigma \not\models X \wedge \forall A \in (\text{total}(\Sigma) - X)(\Sigma \models XA)\}$$

is the set of anti-keys with respect to Σ .

Next we require a generalization for the notion of an agree set. Recall [64, 73] that the agree set of a total relation consists of all attribute subsets on which two distinct tuples of the relation agree.

DEFINITION 5.2. *Let R be some relation schema. For two tuples t, t' over R define*

$$\begin{aligned} \text{ag}(t, t') := & \{(X, Y) \mid \forall A \in R \\ & ((t[A] = t'[A] \wedge t[A] \neq \text{ni}) \leftrightarrow A \in X) \wedge \\ & ((t[A] = t'[A]) \leftrightarrow A \in Y)\} \end{aligned}$$

as the agree set of t and t' . For a relation r over R let

$$\text{ag}(r) := \{\text{ag}(t, t') \mid t, t' \in r \wedge t \neq t'\}$$

be the agree set of r . Furthermore, let

$$\text{ag}^s(r) := \{X \mid (X, Y) \in \text{ag}(r)\}$$

denote the strong agree set of r .

REMARK 1. Note that in the case of a total relation r all elements of $\text{ag}(r)$ have the form (X, X) . Therefore, the definition of agree sets in total relations requires only one of the two components; cf. [12, 64, 73].

Recall that a total non-empty relation is Armstrong for a set Σ of keys if and only if $\Sigma^{-1} \subseteq \text{ag}(r)$ and no key in Σ is contained in any agree set of r . The following result generalizes this characterization to partial relations. For a relation r over R let $\text{total}(r) := \{A \in R \mid \forall t \in r(t[A] \neq \text{ni})\}$.

THEOREM 5.1. *Let R be some relation schema and let Σ be a key set over R . A non-empty relation r over R is an Armstrong relation for Σ if and only if the following three conditions are satisfied:*

- (1) $\text{total}(\Sigma) = \text{total}(r)$;
- (2) $\forall X \in \Sigma^{-1} \exists Y \in \text{ag}^s(r)(X \subseteq Y)$;
- (3) $\forall X \in \text{ag}^s(r) \forall K \in \Sigma(K \not\subseteq X)$.

Proof. If $\emptyset \in \Sigma$, then Lemma 5.1 shows the theorem in this case: the one tuple relation satisfies conditions (1), (2) and (3). For the remainder of the proof, we can therefore assume that $\emptyset \notin \Sigma$, and, thereby, $\emptyset \notin \Sigma^*$.

Sufficiency. Let r satisfy the conditions (1), (2) and (3). We show that r is an Armstrong relation for Σ .

First we show that r satisfies Σ . Let $K \in \Sigma$, and assume that r violates K . Then there are some distinct $t, t' \in r$ such that $t[K] = t'[K]$ and where t is K -total. Consequently, $K \subseteq \text{ag}^s(t, t')$. This, however, is a contradiction to condition (3). We conclude that our assumption was wrong and that r satisfies K .

We show next that r violates every key $K \notin \Sigma^*$. Theorem 4.2 implies that either $K \not\subseteq \text{total}(\Sigma) = \text{total}(r)$ or $K \subseteq \text{total}(\Sigma) = \text{total}(r)$ and that, for all $K' \in \Sigma$, we have $K' \not\subseteq K$. In the first case, r violates K . In the second case, there is some $X \in \Sigma^{-1}$ such that $K \subseteq X$. Condition (2) implies that there is some $Y \in \text{ag}^s(r)$ such that $X \subseteq Y$. Consequently, $K \subseteq \text{ag}^s(t, t')$ for some distinct $t, t' \in r$. We conclude that r violates K . This shows that r is an Armstrong relation for Σ .

Necessity. Let r be an Armstrong relation for Σ . We show that r satisfies conditions (1), (2) and (3).

Since r satisfies Σ , it follows that r satisfies the key $\text{total}(\Sigma)$. Hence, r is $\text{total}(\Sigma)$ -total; that is, $\text{total}(\Sigma) \subseteq \text{total}(r)$. Assume there is some attribute $A \in \text{total}(r) - \text{total}(\Sigma)$. According to Theorem 4.2, we conclude that the key $\text{total}(\Sigma)A$ is not implied by Σ . The only possibility for r to violate $\text{total}(\Sigma)A$ and satisfy $\text{total}(\Sigma)$ is for r to be not A -total. This contradicts our assumption that $A \in \text{total}(r)$. Consequently, $\text{total}(r) \subseteq \text{total}(\Sigma)$. This shows that r satisfies condition (1).

Let $X \in \Sigma^{-1}$. In particular, $\Sigma \not\models X$ and $X \subseteq \text{total}(\Sigma)$. Since r is an Armstrong relation, we know that r violates X and that $X \subseteq \text{total}(r)$ (since we have just shown that condition (1) holds). We conclude that there are some distinct $t, t' \in r$ such that $t[X] = t'[X]$ and t is X -total. Consequently, $X \subseteq \text{ag}^s(t, t')$ for some distinct $t, t' \in r$. We have just shown that $X \subseteq Y$ for some $Y \in \text{ag}^s(r)$, i.e. condition (2) holds.

For condition (3), assume that there is some $K \in \Sigma$ and some $X \in \text{ag}^s(r)$ such that $K \subseteq X$. That is means that r would have to violate $K \in \Sigma$, which contradicts the fact that r is an Armstrong relation for Σ . Consequently, condition (3) holds as well. \square

EXAMPLE 8. Consider the relation schema `SCHEDULE`, and assume that Σ consists of the two keys $\{C_ID, Time\}$, $\{L_Name, Time\}$. Consequently, Σ^{-1} consists of the two anti-keys $\{C_ID, L_Name\}$ and $\{Time\}$. The partial relation

C_ID	L_Name	Room	Time
DB201	Ullman	Cotton118	Mon, 11am
DB201	Ullman	Cotton118	Wed, 03pm
DB320	Widom	ni	Wed, 03pm

is an Armstrong relation for Σ . In fact, it is an easy exercise to verify that r satisfies conditions (1), (2) and (3) of Theorem 5.1.

6. COMPUTATION AND COMPLEXITY

We now capitalize on Theorem 5.1 and show how to compute an Armstrong relation for an arbitrarily given set Σ of keys. We will also show that the problem of finding an Armstrong relation for a given set of keys is precisely exponential in the number of total attributes.

6.1. The computation

Before we can present the computation, we adapt a common method [25, 36, 84] for computing the set Σ^{-1} of anti-keys with respect to Σ . This method generates the hyper-graph where the vertex set is $\text{total}(\Sigma)$ and where the hyper-edges are the elements of Σ . The anti-keys are now exactly the complements of the minimal transversals of the hyper-graph with respect to $\text{total}(\Sigma)$. The following algorithm is an adaptation of Demetrovics' construction for candidate key systems over total relations [59].

Algorithm 6.1 (Armstrong relation for keys).

Input: (R, Σ) with a key set Σ over R

Output: Armstrong relation r over R for Σ

Method: let $c_{A,0}, c_{A,1}, \dots \in \text{dom}(A)$ be distinct

(A0) let $\text{total}(\Sigma) := \bigcup \{X \mid X \in \Sigma\}$;

(A1) $r := \{t_0\}$ where

$$t_0(A) := \begin{cases} c_{A,0}, & \text{if } A \in \text{total}(\Sigma) \\ \text{ni}, & \text{else} \end{cases};$$

(A2) if $\emptyset \in \Sigma$, then return r ;

(A3) compute Σ^{-1} ;

(A4) $i := 0$;

(A5) for all $Y \in \Sigma^{-1}$

$i := i + 1$; $r := r \cup \{t_i\}$, where

$$t_i(A) := \begin{cases} c_{A,0}, & \text{if } A \in Y \\ c_{A,i}, & \text{if } A \in \text{total}(\Sigma) - Y \\ \text{ni}, & \text{else} \end{cases};$$

endfor

(A6) return r ;

THEOREM 6.1. Algorithm 6.1, on input (R, Σ) , computes an Armstrong relation for Σ .

Proof. The theorem follows directly from Lemma 5.1 and Theorem 5.1. \square

The following example illustrates Algorithm 6.1; cf. Example 8.

EXAMPLE 9. Consider again the relation schema

`SCHEDULE` = $\{C_ID, L_Name, Time, Room\}$.

Suppose Σ consists of the two keys $\{C_ID, Time\}$, and $\{L_Name, Time\}$. Then the anti-keys with respect to Σ are $\{Time\}$ and $\{C_ID, L_Name\}$. Algorithm 6.1 computes the partial relation

C_ID	L_Name	Room	Time
DB201	Ullman	ni	Mon, 11:00am
DB420	Widom	ni	Mon, 11:00am
DB201	Ullman	ni	Wed, 01:00pm

which is an Armstrong relation for Σ .

COROLLARY 6.1. The class of keys enjoys Armstrong relations.

6.2. Complexity results

We will show now that the user-friendly representation of a key set in the form of an Armstrong relation comes, in general, at a price. In fact, the number of tuples in a minimal Armstrong relation can be exponential in the number of total attributes. Because of this general exponentiality result, we cannot design an algorithm for generating Armstrong relations in polynomial time in the worst case. The following theorem can be proved similarly to the case of total relations [85].

THEOREM 6.2. *Let Σ be a set of keys over some relation schema R and let r be an Armstrong relation for Σ . Then $|\Sigma^{-1}| \leq |ag(r)| \leq \binom{|r|}{2}$.*

Proof. The second condition of Theorem 5.1 says that, for all $X \in \Sigma^{-1}$, there is some $Y \in ag^s(r)$ such that $X \subseteq Y$. If we add the third condition of Theorem 5.1, then we derive that, for all $Y \in ag^s(r)$, there is at most one $X \in \Sigma^{-1}$ such that $X \subseteq Y$. This shows that

$$|\Sigma^{-1}| \leq |ag^s(r)|.$$

Moreover, we have that $|ag^s(r)| \leq |ag(r)|$. Finally, $|ag(r)| \leq \binom{|r|}{2}$ since every distinct pair of distinct tuples in r has precisely one agree set. \square

We define what we mean by *precisely exponential*. Firstly, it means that there is an algorithm for obtaining an Armstrong relation, given the set Σ of keys, where the running time of the algorithm is exponential in the number of total attributes. Secondly, it also means that there is a set Σ of keys in which the number of tuples in each minimal Armstrong relation for Σ is exponential—thus, an exponential amount of time is required in this case simply to write down the relation.

THEOREM 6.3. *The complexity of finding an Armstrong relation for an arbitrarily given set of keys is precisely exponential in the number of total attributes.*

Proof. The time complexity of Algorithm 6.1 is proportional to the time complexity of computing the set Σ^{-1} of anti-keys with respect to Σ . One method is to generate the hyper-graph with vertex set $total(\Sigma)$ and where the hyper-edges are the elements of Σ . The anti-keys are now exactly the complements of the minimal transversals of the hyper-graph with respect to $total(\Sigma)$ [84]. Therefore, Algorithm 6.1 runs in time exponential in the number of total attributes.

It remains to show that there is a set Σ of keys for which the number of tuples in each Armstrong relation for Σ is exponential in the number of total attributes. According to Theorem 6.2 it suffices to find a set Σ of keys such that Σ^{-1} is exponential in the number of total attributes. Such a set is given by

$$\Sigma = \bigcup_{1 \leq i \leq n} \{A_{2i-1}, A_{2i}\}.$$

The set Σ^{-1} consists of those sets that contain precisely one attribute of each element of Σ . Therefore, Σ^{-1} contains precisely 2^n elements with $2n$ total attributes given. \square

6.3. The size of minimal Armstrong relations

Despite the general worst-case exponential complexity in the number of total attributes, Algorithm 6.1 is a fairly simple algorithm for generating Armstrong relations that is quite conservative in its use of time. In this section, we derive a result that allows us to make the term ‘quite conservative’ more precise.

Let the size of an Armstrong relation be defined as the number of tuples that it contains. In practice, the most appealing Armstrong relation for a key set Σ should be of minimal size. The reason is that a small number of tuples is easier to comprehend for humans. Furthermore, if the removal of any tuple from an Armstrong relation for a key set Σ results in a relation that is still Armstrong for Σ , then the tuple did not add any new information about the representation of the key set. For these reasons, it is a practical question to ask how many tuples a minimal Armstrong relation for a given key set requires.

An Armstrong relation r for Σ is said to be *minimal* if there is no Armstrong relation r' for Σ such that $|r'| < |r|$; that is, for a minimal Armstrong relation for Σ there is no Armstrong relation for Σ with a smaller number of tuples. The following theorem is the counterpart of a result over total relations [85].

THEOREM 6.4. *Let Σ be a set of keys over some relation schema R and let r be a minimal Armstrong relation for Σ . Then $\sqrt{1 + 8 \cdot |\Sigma^{-1}|}/2 \leq |r| \leq |\Sigma^{-1}| + 1$.*

Proof. The upper bound follows immediately from Theorem 6.1.

The lower bound follows from Theorem 6.2. Indeed, it follows that $|\Sigma^{-1}| \leq \binom{|r|}{2}$; that is, $|\Sigma^{-1}| \leq |r| \cdot (|r| - 1)/2$. Consequently, $\sqrt{1 + 8 \cdot |\Sigma^{-1}|}/2 \leq |r|$. \square

We conclude that Algorithm 6.1 always computes an Armstrong relation of reasonably small size.

COROLLARY 6.2. *On input (R, Σ) , Algorithm 6.1 computes an Armstrong relation for Σ whose size is at most quadratic in the size of a minimal Armstrong relation for Σ .*

7. EXTREMAL PROBLEMS

From the results in the last section, we can identify those partial relations that obey precisely the keys that are implied by a set of keys. A natural question to ask is about the maximal number of keys that may need to be specified explicitly. It is a significant question to ask since the answer has a direct impact on the complexity of the design and maintenance of a database. If the cardinality of non-redundant sets of constraints is large, then maintenance becomes infeasible. A further natural question to ask is which families of constraints attain these maximal cardinalities. This question is also significant since an answer will tell us which families cause a high complexity for maintaining the consistency of a database. In this section, we settle these problems for the class of keys. We present solutions to even more general versions of these problems, where the number of attributes in any key of such a family is bounded by an arbitrary fixed number. In practice, such a bound may be chosen by a database designer in order to avoid high complexities in the maintenance of the database.

7.1. Families of non-redundant keys

A set Σ of integrity constraints is said to be *non-redundant* if there is no element $\sigma \in \Sigma$ such that $\Sigma - \{\sigma\}$ implies σ [4]. Non-redundant sets are important in database practice as they represent a necessary requirement on what integrity constraints must be enforced. The simplest example, in the context of total relations, is given by a set of minimal keys since such sets consist of elements that are pairwise incomparable with respect to set inclusion. In the context of partial relations, however, minimal keys do not suffice in general: a key might be contained in another key not implied by it. Therefore, non-redundant families of keys have a different structure.

Throughout, let n and k be integers with $1 \leq k \leq n$ and

$$\begin{aligned} [n] &:= \{1, 2, \dots, n\}, \\ 2^{[n]} &:= \{F : F \subseteq [n]\}, \\ \binom{[n]}{\ell} &:= \{F \in 2^{[n]} : |F| = \ell\}. \end{aligned}$$

We study families $\mathcal{F} \subseteq 2^{[n]}$ satisfying the following implication:

$$\text{There is no subset } \{X, Y\} \subseteq \mathcal{F} \text{ with } X \subset Y \subseteq \bigcup_{F \in \mathcal{F} \setminus \{Y\}} F. \quad (1)$$

Note that \mathcal{F} is allowed to contain the empty set.

Our objective is to maximize $|\mathcal{F}|$ over all $\mathcal{F} \subseteq 2^{[n]}$ satisfying (1) and such that $|F| \leq k$ for all $F \in \mathcal{F}$. In the sequel, we shall solve this problem and determine all extremal families \mathcal{F} .

Theorem 4.2 provides us with a sufficient and necessary condition for families of keys to be non-redundant.

THEOREM 7.1. $\mathcal{F} \subseteq 2^{[n]}$ is non-redundant if and only if \mathcal{F} satisfies (1).

Proof. \mathcal{F} is redundant if and only if there is some key $X \in \mathcal{F}$ such that $\mathcal{F} - \{X\}$ implies X . According to Theorem 4.2 this means that there is some key $Y \in \mathcal{F} - \{X\}$ such that $Y \subset X \subseteq \bigcup_{F \in \mathcal{F} - \{X\}} F$. However, this is just the negation of condition (1). \square

7.2. Exchange operation

LEMMA 7.1. Let $\mathcal{F} \subseteq 2^{[n]}$ satisfy (1), and assume that there are $X, Y \in \mathcal{F}$ such that $\emptyset \subsetneq X \subsetneq Y$. Then $\sigma_{XY}(\mathcal{F}) := (\mathcal{F} \setminus \{Y\}) \cup \{Y \setminus X\}$ satisfies (1) and $|\sigma_{XY}(\mathcal{F})| = |\mathcal{F}|$.

Proof. First note that $Y \setminus X \notin \mathcal{F}$ as \mathcal{F} satisfies (1) and $Y \setminus X \subset Y = X \cup (Y \setminus X)$. Hence, $|\sigma_{XY}(\mathcal{F})| = |\mathcal{F}|$.

Assume that $\sigma_{XY}(\mathcal{F})$ does not satisfy (1). Then

$$X' \subset Y' \subseteq \bigcup_{F \in \sigma_{XY}(\mathcal{F}) \setminus \{Y'\}} F$$

for some $\{X', Y'\} \subseteq \sigma_{XY}(\mathcal{F})$.

If $Y \setminus X \notin \{X', Y'\}$, then $\{X', Y'\} \subseteq \mathcal{F}$ and

$$X' \subset Y' \subseteq \bigcup_{F \in \sigma_{XY}(\mathcal{F}) \setminus \{Y'\}} F \subseteq \bigcup_{F \in \mathcal{F} \setminus \{Y'\}} F,$$

contradicting (1).

If $Y \setminus X \in \{X', Y'\}$, then $Y \setminus X \subseteq Y'$, and we have

$$\begin{aligned} Y \setminus X &\subseteq \bigcup_{F \in \sigma_{XY}(\mathcal{F}) \setminus \{Y'\}} F \\ &\subseteq \bigcup_{F \in \sigma_{XY}(\mathcal{F}) \setminus \{Y \setminus X\}} F \\ &= \bigcup_{F \in \mathcal{F} \setminus \{Y\}} F. \end{aligned}$$

As $X \in \mathcal{F} \setminus \{Y\}$, this implies

$$X \subset Y = (Y \setminus X) \cup X \subseteq \left(\bigcup_{F \in \mathcal{F} \setminus \{Y\}} F \right) \cup X = \bigcup_{F \in \mathcal{F} \setminus \{Y\}} F,$$

which is again a contradiction to (1). \square

COROLLARY 7.1. Let $\mathcal{F} \subseteq 2^{[n]}$ satisfy (1). By a finite number of applications of operations σ_{XY} like in Lemma 7.1, \mathcal{F} can be transformed into some $\mathcal{F}' \subseteq 2^{[n]}$ such that $|\mathcal{F}| = |\mathcal{F}'|$ and $\mathcal{F}' \setminus \{\emptyset\}$ is an anti-chain.

Proof. Beginning with \mathcal{F} , keep applying operations σ_{XY} as long as there are X, Y with $\emptyset \subsetneq X \subsetneq Y$ in the resulting family. This process terminates after a finite number of steps as in each step the volume of the family (i.e. the sum of the cardinalities of all sets in the family) decreases. Now the claim follows by Lemma 7.1. \square

7.3. Solution of the problem

THEOREM 7.2. Let n and k be integers, with $1 \leq k \leq n$, and let $\mathcal{F} \subseteq 2^{[n]}$ such that \mathcal{F} satisfies (1) and $|F| \leq k$ for all $F \in \mathcal{F}$.

(i) If $n \leq 3$ or $k = 1$, then $|\mathcal{F}| \leq n + 1$, where equality holds if and only if

$$\mathcal{F} = \binom{[n]}{1} \cup \{\emptyset\}.$$

(ii) If $2 \leq k \leq n/2$, then $|\mathcal{F}| \leq \binom{n}{k}$, where equality holds if and only if

$$\mathcal{F} = \binom{[n]}{k}.$$

(iii) If $2 \leq n/2 < k$, then $|\mathcal{F}| \leq \binom{n}{\lfloor n/2 \rfloor}$, where equality holds if and only if

$$\mathcal{F} = \binom{[n]}{\lceil (n-1)/2 \rceil} \text{ or } \mathcal{F} = \binom{[n]}{\lfloor (n+1)/2 \rfloor}.$$

Proof. (i) If $k = 1$, then there is nothing to prove. For $2 \leq k \leq n \leq 3$, it is an easy exercise to verify that $|\mathcal{F}|$ cannot exceed $n + 1$, and that the only family of size $n + 1$ that satisfies (1) is $\binom{[n]}{1} \cup \{\emptyset\}$.

(ii) Assume that $2 \leq k \leq n/2$.

First note that if $\emptyset \in \mathcal{F}$, then by (1) and as $\emptyset \subset F$ for all $F \in \mathcal{F} \setminus \{\emptyset\}$ every set in $\mathcal{F} \setminus \{\emptyset\}$ must contain some element of $[n]$ that is not contained in any other set in \mathcal{F} . Hence, $|\mathcal{F}| \leq n + 1 < \binom{n}{k}$ in this case.

If \mathcal{F} is an anti-chain, then $|\mathcal{F}| \leq \binom{n}{k}$ with equality only for $\mathcal{F} = \binom{[n]}{k}$ as an immediate consequence of the LYM-inequality (see [86, p. 17] for instance).

Assume that $\emptyset \notin \mathcal{F}$ and that \mathcal{F} is not an anti-chain. It remains to show that $|\mathcal{F}| < \binom{n}{k}$ in this case. By Corollary 7.1, \mathcal{F} can be transformed into an anti-chain \mathcal{A} of the same size by successively applying operations σ_{XY} . By the definition of σ_{XY} , we have $|A| \leq k$ for all $A \in \mathcal{A}$. If $\mathcal{A} \neq \binom{[n]}{k}$, then the claim follows. On the other hand, $\mathcal{A} = \binom{[n]}{k}$ implies that the volume of \mathcal{A} is larger than the volume of \mathcal{F} , which is impossible.

(iii) Assume that $2 \leq n/2 < k$.

If $\emptyset \in \mathcal{F}$, then analogously to the proof of (ii), we obtain $|\mathcal{F}| \leq n + 1 < \binom{n}{\lfloor n/2 \rfloor}$.

If \mathcal{F} is an anti-chain, then by Sperner's theorem [87], $|\mathcal{F}| \leq \binom{n}{\lfloor n/2 \rfloor}$ with equality if and only if $\mathcal{F} = \binom{[n]}{\lfloor (n-1)/2 \rfloor}$ or $\mathcal{F} = \binom{[n]}{\lfloor (n+1)/2 \rfloor}$.

Assume that $\emptyset \notin \mathcal{F}$ and that \mathcal{F} is not an anti-chain. Again, by Corollary 7.1, \mathcal{F} can be transformed into an anti-chain \mathcal{A} of the same size by successively applying operations σ_{XY} . This implies $|\mathcal{F}| \leq \binom{n}{\lfloor n/2 \rfloor}$ with equality only if $\mathcal{A} = \binom{[n]}{\lfloor (n-1)/2 \rfloor}$ or $\mathcal{A} = \binom{[n]}{\lfloor (n+1)/2 \rfloor}$. Assume that equality holds. Considering the last step when transforming \mathcal{F} into \mathcal{A} , by Lemma 7.1 and Corollary 7.1, there is a family $\mathcal{F}' \subseteq 2^{[n]}$ which satisfies (1) and contains two sets X, Y such that $\emptyset \subsetneq X \subsetneq Y$ and $\sigma_{XY}(\mathcal{F}') = \mathcal{A}$. But this is impossible as $Y \subseteq \bigcup_{A \in \mathcal{A} \setminus \{Y \setminus X\}} A = [n]$ and Y is a superset of at least one set in $\mathcal{A} \setminus \{Y \setminus X\}$. \square

EXAMPLE 10. Consider again the relation schema *SCHEDULE* with the $n = 4$ attributes: *C_ID*, *L_Name*, *Time* and *Room*. If $k = 1$, then the maximal family \mathcal{F}_1 of non-redundant keys of size at most 1 has the five elements: \emptyset , $\{C_ID\}$, $\{L_Name\}$, $\{Time\}$ and $\{Room\}$. A relation over *SCHEDULE* that is Armstrong for \mathcal{F}_1 is as follows:

C_ID	L_Name	Room	Time
DB201	Ullman	Cotton118	Mon, 11am

If $k = 2$, then the maximal family \mathcal{F}_2 of non-redundant keys of size at most 2 contains all the six two-element subsets of *SCHEDULE*.

8. CONCLUSION AND FUTURE WORK

We have investigated a class of keys over partial database relations. Following Codd's principle of entity integrity, we

have defined keys such that every meaningful relation is unique and total on the key attributes. We have seen that the interaction of keys over partial relations is different from that over total relations. On the basis of this observation, we have established an axiomatization for the implication of keys, and an algorithm that decides the associated implication problem in linear time. Furthermore, we have characterized Armstrong relations for the class of keys, and established an algorithm that computes an Armstrong relation for an arbitrarily given set of keys as input. While the complexity of finding an Armstrong relation is precisely exponential in the size of the given key set, our algorithm computes an Armstrong relation where the number of tuples is the number of anti-keys with respect to the given key set. We have shown that this number is at most quadratic in the size of a minimal Armstrong relation. In general, Armstrong relations for keys are invaluable design aids that can help designers, domain experts and end users to understand, convey and test the specification of the target SQL table [11]. Finally, we have identified all the families of keys that attain a maximal cardinality of non-redundant key sets, even under the restriction where all keys in the given set have an upper bound on their maximal number of attributes. This provides a data administrator with invaluable information for the complexity required to maintain the database.

We hope that the results of this article can stimulate further research that bridges the gap between the existing theory of data dependencies and database practice. In particular, if a database is known to satisfy a set of dependencies, then this information can result in a better design of the database schema and improve the performance of common data processing tasks such as updates and queries [2]; cf. Example 4. In particular, the satisfaction of key dependencies can result in a decrease of the indexing space [2]. Over total relations, keys form a special case of functional dependencies. Our class of keys is subsumed by Atzeni and Morfuni's class of functional dependencies over partial relations when studied together with a null-free subschema (the set of attributes declared NOT NULL) [8]. The results in our article may provide valuable hints to gain insights into the existence and properties of Armstrong relations for Atzeni and Morfuni's class, and to extend the existing design theory for schemata that permit only total relations [3, 12, 20–22, 64, 88] to schemata that permit instances of real database systems; cf. [89].

Fagin has shown that Armstrong relations exist for so-called *implicational dependencies* [90]. While such general classes of data dependencies cannot generally be expected to be understood nor utilized by database designers in practice, it would be a worthwhile endeavour to identify broad classes of dependencies that enjoy Armstrong relations in the presence of null values, too. For example, the classes of functional and multivalued dependencies have been studied in the context of the no-information interpretation [7, 8, 42, 83].

Other directions include the problem of dependency inference [25] or data cleaning [51], but also the investigation of extremal problems [86], e.g. for the class of functional

dependencies [75], in the context of partial relations. Finally, all these problems should be considered for other interpretations of null values [78–81].

FUNDING

This research was supported by the Marsden fund council from Government funding, administered by the Royal Society of New Zealand. The first author was supported by a research grant of the Alfried Krupp von Bohlen and Halbach foundation, administered by the German Scholars organization.

REFERENCES

- [1] Codd, E.F. (1970) A relational model of data for large shared data banks. *Commun. ACM*, **13**, 377–387.
- [2] Abiteboul, S., Hull, R. and Vianu, V. (1995) *Foundations of Databases*. Addison-Wesley, Reading, MA.
- [3] Fagin, R. (1981) A normal form for relational databases that is based on domains and keys. *ACM Trans. Database Syst.*, **6**, 387–415.
- [4] Maier, D. (1983) *The Theory of Relational Databases*. Computer Science Press, Rockville, MD.
- [5] Codd, E.F. (1986) Missing information (applicable and inapplicable) in relational databases. *SIGMOD Rec.*, **15**, 53–78.
- [6] Zaniolo, C. (1984) Database relations with null values. *J. Comput. Syst. Sci.*, **28**, 142–166.
- [7] Lien, E. (1982) On the equivalence of database models. *J. ACM*, **29**, 333–362.
- [8] Atzeni, P. and Morfuni, N. (1986) Functional dependencies and constraints on null values in database relations. *Inf. Control*, **70**, 1–31.
- [9] Fagin, R. (1982) Armstrong Databases. Technical Report RJ3440(40926), IBM Research Laboratory, San Jose, CA, USA.
- [10] De Marchi, F., Lopes, S., Petit, J.-M. and Toumani, F. (2003) Analysis of existing databases at the logical level: the DBA companion project. *SIGMOD Rec.*, **32**, 47–52.
- [11] Langeveldt, W.-D. and Link, S. (2010) Empirical evidence for the usefulness of Armstrong relations in the acquisition of meaningful functional dependencies. *Inf. Systems*, **35**, 352–374.
- [12] Mannila, H. and Rähkä, K.-J. (1986) Design by example: an application of Armstrong relations. *J. Comput. Syst. Sci.*, **33**, 126–141.
- [13] Levene, M. and Loizou, G. (1998) Axiomatisation of functional dependencies in incomplete relations. *Theor. Comput. Sci.*, **206**, 283–300.
- [14] Paulley, G. and Larson, P.-A. (1994) Exploiting Uniqueness in Query Optimization. *Proc. 10th Int. Conf. Data Engineering (ICDE)*, Houston, USA, February 14–18, pp. 68–79. IEEE Computer Society.
- [15] Fagin, R. and Vardi, M. (1984) The Theory of Data Dependencies—an overview. *Proc. 11th Colloquium on Automata, Languages and Programming (ICALP)*, Antwerp, Belgium, July 16–20, Lecture Notes in Computer Science 172, pp. 1–22. Springer.
- [16] Thalheim, B. (1991) *Dependencies in Relational Databases*. Teubner, Leipzig.
- [17] Codd, E.F. (1972) Further Normalization of the Database Relational Model. *Proc. 1971 Courant Computer Science Symposia Series 6: Data Base Systems*, New York, USA, May 24–25, pp. 33–64. Prentice Hall.
- [18] Beeri, C., Mendelzon, A.O., Sagiv, Y. and Ullman, J.D. (1981) Equivalence of relational database schemes. *SIAM J. Comput.*, **10**, 352–370.
- [19] Bernstein, P. (1976) Synthesizing third normal form relations from functional dependencies. *ACM Trans. Database Syst.*, **1**, 277–298.
- [20] Beeri, C., Bernstein, P.A. and Goodman, N. (1978) A Sophisticate’s Introduction to Database Normalization Theory. *Proc. 4th Int. Conf. Very Large Databases (VLDB)*, West Berlin, Germany, September 13–15, pp. 113–124. IEEE-CS.
- [21] Beeri, C. and Bernstein, P. (1979) Computational problems related to the design of normal form relational schemas. *ACM Trans. Database Syst.*, **4**, 30–59.
- [22] Biskup, J., Dayal, U. and Bernstein, P. (1979) Synthesizing Independent Database Schemas. *Proc. ACM SIGMOD Int. Conf. Management of Data (SIGMOD)*, Boston, USA, May 30–June 1, pp. 143–151. ACM.
- [23] Maier, D. (1980) Minimum covers in relational database model. *J. ACM*, **27**, 664–674.
- [24] Lucchesi, C. and Osborn, S. (1978) Candidate keys for relations. *J. Comput. Syst. Sci.*, **17**, 270–279.
- [25] Mannila, H. and Rähkä, K.-J. (1994) Algorithms for inferring functional dependencies from relations. *Data Knowl. Eng.*, **12**, 83–99.
- [26] Biskup, J., Embley, D. and Lochner, J. (2008) Reducing inference control to access control for normalized database schemas. *Inf. Process Lett.*, **106**, 8–12.
- [27] Klug, A. and Price, R. (1982) Determining view dependencies using tableaux. *ACM Trans. Database Syst.*, **7**, 361–380.
- [28] Cheng, Q., Gryz, J., Koo, F., Leung, C., Liu, L., Qian, X. and Schiefer, K. (1999) Implementation of Two Semantic Query Optimization Techniques in DB2 Universal Database. *Proc. 25th Int. Conf. Very Large Databases (VLDB)*, Edinburgh, Scotland, September 7–10, pp. 687–698. Morgan Kaufmann.
- [29] Deutsch, A., Ludäscher, B. and Nash, A. (2007) Rewriting queries using views with access patterns under integrity constraints. *Theor. Comput. Sci.*, **371**, 200–226.
- [30] Deutsch, A., Popa, L. and Tannen, V. (2006) Query reformulation with constraints. *SIGMOD Rec.*, **35**, 65–73.
- [31] Arenas, M., Fan, W. and Libkin, L. (2008) On the complexity of verifying consistency of XML specifications. *SIAM J. Comput.*, **38**, 841–880.
- [32] Arenas, M. and Libkin, L. (2004) A normal form for XML documents. *ACM Trans. Database Syst.*, **29**, 195–232.
- [33] Bojanczyk, M., Muscholl, A., Schwentick, T. and Segoufin, L. (2009) Two-variable logic on data trees and XML reasoning. *J. ACM*, **56**, Article 13, 48p.
- [34] Buneman, P., Davidson, S., Fan, W., Hara, C. and Tan, W. (2003) Reasoning about keys for XML. *Inf. Syst.*, **28**, 1037–1063.
- [35] Fischer, P.C., Saxton, L.V., Thomas, S.J. and Van Gucht, D. (1985) Interactions between dependencies and nested relational structures. *J. Comput. Syst. Sci.*, **31**, 343–354.

- [36] Gottlob, G., Pichler, R. and Wei, F. (2010) Tractable database design and datalog abduction through bounded treewidth. *Inf. Syst.*, **35**, 278–298.
- [37] Hartmann, S. and Link, S. (2007) Unlocking Keys for XML Trees. *Proc. 11th Int. Conf. Database Theory (ICDT)*, Barcelona, Spain, January 10–12, Lecture Notes in Computer Science 4353, pp. 104–118. Springer.
- [38] Hartmann, S. and Link, S. (2008) Characterising nested database dependencies by fragments of propositional logic. *Ann. Pure Appl. Log.*, **152**, 84–106.
- [39] Hartmann, S., Köhler, H., Link, S., Trinh, T. and Wang, J. (2008) On the Notion of an XML Key. *Revised Selected Papers from the 3rd Int. Workshop on Semantics in Data and Knowledge Bases (SDKB)*, Nantes, France, March 29, Lecture Notes in Computer Science 4925, pp. 103–112. Springer.
- [40] Hartmann, S. and Link, S. (2009) Efficient reasoning about a robust XML key fragment. *ACM Trans. Database Syst.*, **34**, Article 10, 33p.
- [41] Hartmann, S. and Link, S. (2009) Expressive, Yet Tractable XML Keys. *Proc. 12th Int. Conf. Extending Database Technology (EDBT)*, Saint Petersburg, Russia, March 24–26, ACM International Conference Proceeding Series 360, pp. 357–367. ACM.
- [42] Hartmann, S. and Link, S. (2010) When Data Dependencies over SQL Tables Meet the Logics of Paradox and S-3. *Proc. 29th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS)*, Indianapolis, USA, June 6–11, pp. 317–326. ACM.
- [43] Jensen, C., Snodgrass, R. and Soo, M. (1996) Extending existing dependency theory to temporal databases. *IEEE Trans. Knowl. Data Eng.*, **8**, 563–582.
- [44] Kolahi, S. (2007) Dependency-preserving normalization of relational and XML data. *J. Comput. Syst. Sci.*, **73**, 636–647.
- [45] Sözat, M. and Yazici, A. (2001) A complete axiomatization for fuzzy functional and multivalued dependencies in fuzzy database relations. *ACM Fuzzy Sets Syst.*, **117**, 161–181.
- [46] Tari, Z., Stokes, J. and Spaccapietra, S. (1997) Object normal forms and dependency constraints for object-oriented schemata. *ACM Trans. Database Syst.*, **22**, 513–569.
- [47] Vincent, M., Liu, J. and Liu, C. (2004) Strong functional dependencies and their application to normal forms in XML. *ACM Trans. Database Syst.*, **29**, 445–462.
- [48] Weddell, G. (1992) Reasoning about functional dependencies generalized for semantic data models. *ACM Trans. Database Syst.*, **17**, 32–64.
- [49] Wijzen, J. (1999) Temporal FDs on complex objects. *ACM Trans. Database Syst.*, **24**, 127–176.
- [50] Link, S. (2001) Consistency Enforcement in Databases. *Revised Papers from the 2nd Int. Workshop on Semantics in Databases*, Dagstuhl Castle, Germany, January 7–12, Lecture Notes in Computer Science 2582, pp. 139–158. Springer.
- [51] Fan, W., Geerts, F., Jia, X. and Kementsisidis, A. (2008) Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst.*, **33**, Article 6, 48p.
- [52] Davidson, S., Fan, W. and Hara, C. (2007) Propagating XML constraints to relations. *J. Comput. Syst. Sci.*, **73**, 316–361.
- [53] Chomicki, J. (2007) Consistent Query Answering: Five Easy Pieces. *Proceedings of the 11th Int. Conf. Database Theory (ICDT)*, Barcelona, Spain, January 10–12, Lecture Notes in Computer Science 4353, pp. 1–17. Springer.
- [54] Fagin, R., Kolaitis, P., Popa, L. and Tan, W. (2009) Reverse Data Exchange: Coping with Nulls. *Proc. 28th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS)*, Providence, USA, June 19–July 1, pp. 23–32. ACM.
- [55] Libkin, L. (2006) Data Exchange and Incomplete Information. *Proc. 25th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems (PODS)*, Chicago, USA, June 26–28, pp. 60–69. ACM.
- [56] Miklau, G. and Suciu, D. (2007) A formal analysis of information disclosure in data exchange. *J. Comput. Syst. Sci.*, **73**, 507–534.
- [57] Cali, A., Calvanese, D., De Giacomo, G. and Lenzerini, M. (2004) Data integration under integrity constraints. *Inf. Syst.*, **29**, 147–163.
- [58] Miller, R., Hernandez, M., Haas, L., Yan, L.-L., Ho, C., Fagin, R. and Popa, L. (2001) The Clío project: managing heterogeneity. *SIGMOD Rec.*, **30**, 78–83.
- [59] Demetrovics, J. (1980) On the equivalence of candidate keys with Sperner systems. *Acta Cybern.*, **4**, 247–252.
- [60] Demetrovics, J. and Gyepesi, G. (1983) A note on minimal matrix representation of closure operations. *Combinatorica*, **2**, 177–179.
- [61] Katona, G. and Tichler, K. (2006) Some Contributions to the Minimum Representation Problem of Key Systems. *Proc. 4th Int. Symp. Foundations of Information and Knowledge Systems (FoIKS)*, Budapest, Hungary, February 14–17, Lecture Notes in Computer Science 3861, pp. 240–257. Springer.
- [62] Sali, A. (2004) Minimal Keys in Higher-Order Datamodels. *Proc. 3rd Int. Symp. Foundations of Information and Knowledge Systems (FoIKS)*, Wilhelminenburg Castle, Austria, February 17–20, Lecture Notes in Computer Science 2942, pp. 242–251. Springer.
- [63] Armstrong, W.W. (1974) Dependency structures of database relationships. *Inf. Process.*, **74**, 580–583.
- [64] Beerli, C., Dowd, M., Fagin, R. and Statman, R. (1984) On the structure of Armstrong relations for functional dependencies. *J. ACM*, **31**, 30–46.
- [65] Demetrovics, J. and Thi, V. (1995) Armstrong relations, functional dependencies and strong dependencies. *Comput. Artif. Intell.*, **14**.
- [66] Fagin, R. and Vardi, M. (1983) Armstrong databases for functional and inclusion dependencies. *Inf. Process. Lett.*, **16**, 13–19.
- [67] Gottlob, G. and Libkin, L. (1990) Investigation on Armstrong relations, dependency inference, and excluded functional dependencies. *Acta Cybern.*, **9**, 385–402.
- [68] De Marchi, F. and Petit, J.-M. (2007) Semantic sampling of existing databases through informative Armstrong databases. *Inf. Syst.*, **32**, 446–457.
- [69] Sali, A. and Schewe, K.-D. (2008) Keys and Armstrong databases in trees with restructuring. *Acta Cybern.*, **18**, 529–556.
- [70] Sali, A. and Székely, L. (2008) On the Existence of Armstrong Instances with Bounded Domains. *Proc. 5th Int. Symp. Foundations of Information and Knowledge Systems (FoIKS)*, Pisa, Italy, February 11–15, Lecture Notes in Computer Science 4932, pp. 151–157. Springer.

- [71] Vincent, M. and Srinivasan, B. (1993) Armstrong Relations for Functional and Multivalued Dependencies in Relational Databases. *Proc. 4th Australian Database Conference (ADC)*, Brisbane, Australia, February 1–2, pp. 317–328. World Scientific.
- [72] Silva, A. and Melkanoff, M. (1979) A Method for Helping Discover the Dependencies of a Relation. *Proc. Workshop on Formal Bases for Data Bases—Advances in Data Base Theory*, Toulouse, France, December 12–14, pp. 115–133. Plenum Press.
- [73] Demetrovics, J. (1978) On the number of candidate keys. *Inf. Process. Lett.*, **7**, 266–269.
- [74] Demetrovics, J., Katona, G., Miklos, D., Seleznev, O. and Thalheim, B. (1998) Asymptotic properties of keys and functional dependencies in random databases. *Theor. Comput. Sci.*, **190**, 151–166.
- [75] Demetrovics, J., Katona, G., Miklos, D. and Thalheim, B. (2006) On the Number of Independent Functional Dependencies. *Proc. 4th Int. Symp. Foundations of Information and Knowledge Bases (FoIKS)*, Budapest, Hungary, February 14–17, Lecture Notes in Computer Science 3861, pp. 83–91. Springer.
- [76] Katona, G. (1992) Combinatorial and Algebraic Results for Database Relations. *Proc. 4th Int. Conf. Database Theory (ICDT)*, Berlin, Germany, October 14–16, Lecture Notes in Computer Science 646, pp. 1–20. Springer.
- [77] Tichler, K. (2004) Extremal theorems for databases. *Ann. Math. Artif. Intell.*, **40**, 165–182.
- [78] Grant, J. (1977) Null values in a relational data base. *Inf. Process. Lett.*, **6**, 156–157.
- [79] Grahne, G. (1984) Dependency Satisfaction in Databases with Incomplete Information. *Proc. 10th Int. Conf. Very Large Databases (VLDB)*, Singapore, August 27–31, pp. 37–45. IEEE Computer Society.
- [80] Makinouchi, A. (1977) A Consideration on Normal Form of Not-Necessarily-Normalized Relation in the Relational Data Model. *Proc. 3rd Int. Conf. Very Large Databases (VLDB)*, Tokyo, Japan, October 6–8, pp. 447–453. IEEE Computer Society.
- [81] Gottlob, G. and Zicari, R. (1988) Closed World Databases Opened Through Null Values. *Proc. 14th Int. Conf. Very Large Databases (VLDB)*, Los Angeles, USA, August 29–September 1, pp. 50–61. IEEE Computer Society.
- [82] Thalheim, B. (1989) On semantic issues connected with keys in relational databases permitting null values. *Elektron. Inf.verarb. Kybern.*, **25**, 11–20.
- [83] Link, S. (2008) On the implication of multivalued dependencies in partial database relations. *Int. J. Found. Comput. Sci.*, **19**, 691–715.
- [84] Demetrovics, J. and Thi, V. (1987) Keys, antikeys and prime attributes. *Annales Univ. Sci. Budapest, Sect. Comp.*, **8**, 35–52.
- [85] Demetrovics, J. and Katona, O. (1981) Extremal Combinatorial Problems in Relational Data Base. *Proc. Fundamentals of Computing Theory (FCT)*, Szeged, Hungary, August 24–28, Lecture Notes in Computer Science 117, pp. 110–119. Springer.
- [86] Engel, K. (1997) *Sperner Theory*. Cambridge University Press, Cambridge, UK.
- [87] Sperner, E. (1928) Ein Satz über Untermengen einer endlichen Menge. *Math Z.*, **27**, 544–548.
- [88] Levene, M. and Vincent, M. (2000) Justification for inclusion dependency normal form. *IEEE Trans. Knowl. Data Eng.*, **12**, 281–291.
- [89] Levene, M. and Loizou, G. (1999) Database design for incomplete relations. *ACM Trans. Database Syst.*, **24**, 80–125.
- [90] Fagin, R. (1982) Horn clauses and database dependencies. *J. ACM*, **29**, 952–985.