CDMTCS
Research
Report
Series

SQL Schema Design: Foundations, Normal Forms, and Normalization

Henning Köhler
Massey University,
Palmerston North, New Zealand

Sebastian Link
University of Auckland,
Auckland, New Zealand

CDMTCS-496
April 2016

Centre for Discrete Mathematics and
Theoretical Computer Science

SQL Schema Design: Foundations, Normal Forms, and Normalization

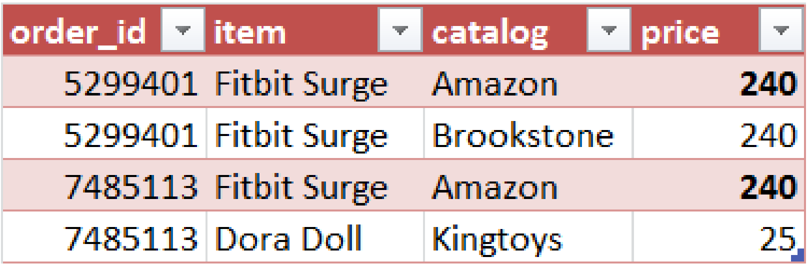
Henning Köhler
Massey University, New Zealand
h.koehler@massey.ac.nz

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

April 10, 2016

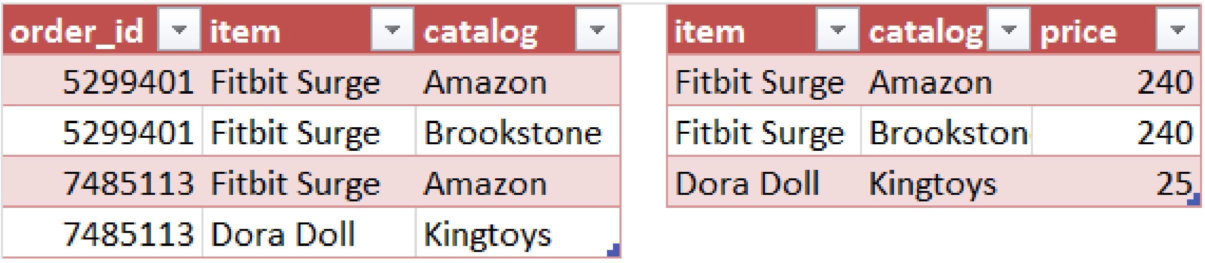
Abstract

Normalization helps us find a database schema at design time that can process the most frequent updates efficiently at run time. Unfortunately, relational normalization only works for idealized database instances in which duplicates and null markers are not present. On one hand, these features occur frequently in real-world data compliant with the industry standard SQL, and especially in modern application domains. On the other hand, the features impose challenges that have made it impossible so far to extend the existing forty year old normalization framework to SQL. We introduce a new class of functional dependencies and show that they provide the right notion for SQL schema design. Axiomatic and linear-time algorithmic characterizations of the associated implication problem are established. These foundations enable us to propose a Boyce-Codd normal form for SQL. Indeed, we justify the normal form by showing that it permits precisely those SQL instances which are free from data redundancy. Unlike the relational case, there are SQL schemata that cannot be converted into Boyce-Codd normal form. Nevertheless, for an expressive sub-class of our functional dependencies we establish a normalization algorithm that always produces a schema in Value-Redundancy free normal form. This normal form permits precisely those instances which are free from any redundant data value occurrences other than the null marker. Experiments show that our functional dependencies occur frequently in real-world data and that they are effective in eliminating redundant values from these data sets without loss of information.



order_id	item	catalog	price
5299401	Fitbit Surge	Amazon	**240**
5299401	Fitbit Surge	Brookstone	240
7485113	Fitbit Surge	Amazon	**240**
7485113	Dora Doll	Kingtoys	25

Figure 1: The relation `purchase`



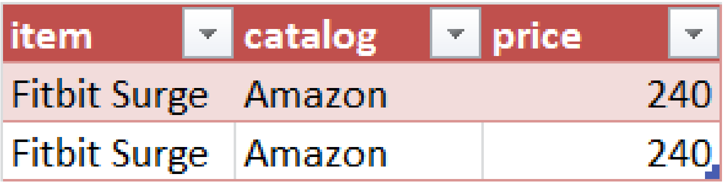
order_id	item	catalog
5299401	Fitbit Surge	Amazon
5299401	Fitbit Surge	Brookstone
7485113	Fitbit Surge	Amazon
7485113	Dora Doll	Kingtoys

item	catalog	price
Fitbit Surge	Amazon	240
Fitbit Surge	Brookston	240
Dora Doll	Kingtoys	25

Figure 2: Lossless decomposition of `purchase`

1 Introduction

Background of idealized instances. The relational normalization framework enables us to derive a database schema at design time that permits exactly those relations as instances that are free from any redundant data value occurrences at run time. This distinguished feature makes it possible to process updates efficiently. De-normalization refers to the process in which different relation schemata are joined in order to process frequent queries more efficiently. The original techniques for this (de-)normalization framework are based on the fundamental notions of functional dependency, data redundancy, and Boyce-Codd normal form. As a simple running example consider the relation schema PURCHASE with four attributes *o(rder_id)*, *i(tem)*, *c(atalog)*, and *p(rice)*, which store information about items that have been ordered from a catalog for a price. Any reasonable instance is expected to satisfy the business rule that the same item from the same catalog cannot have different prices, which we can express as the functional dependency (FD) $\mathit{item}, \mathit{catalog} \to \mathit{price}$. As an example, the relation `purchase` in Figure 1 satisfies this FD. The example also shows that $\{\mathit{item}, \mathit{catalog}\}$ does not form a key over PURCHASE, because there are items from a catalog that occur in different orders. This example illustrates the exact situation in which a schema permits instances in which redundant data values occur. Here, each bold occurrence of **240** in the *price* column is redundant in the sense that *any* change of one 240-value to a different value would result in a relation that violates the FD $\mathit{item}, \mathit{catalog} \to \mathit{price}$. The Boyce-Codd normal form (BCNF) condition provides a syntactic characterization of those schemata which permit exactly those instances in which no redundant data values occur. In fact, a given relation schema T is in BCNF with respect to a given set Σ of FDs if and only if for every non-trivial FD $X \to Y$ in Σ it is the case that $X \to T$ is implied by Σ. Our

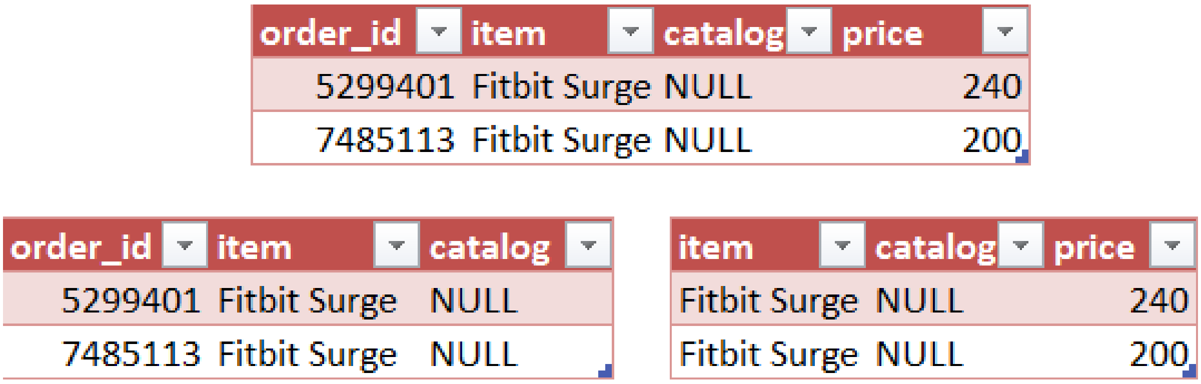


item	catalog	price
Fitbit Surge	Amazon	240
Fitbit Surge	Amazon	240

Figure 3: Instance satisfying all FDs and no keys

Purchase example schema violates BCNF, and our relation purchase is an example of an instance in which redundant data values occur. Redundant data value occurrences can incur prohibitively expensive costs under updates, as all occurrences of a redundant data value must be modified consistently. In relational databases it is always possible to obtain a lossless BCNF decomposition of a given schema. That is, if a given schema is not in BCNF, then we can always convert it into a schema that is. The foundation for this technique is the result that a T-relation I that satisfies a given FD $X \to Y$ is the lossless join of its projections on XY and $X(T - XY)$, that is, $I = I[XY] \bowtie I[X(T - XY)]$. In our example, we can decompose purchase into its two projections purchase$[oic]$ and purchase$[icp]$ as shown in Figure 2, without loss of information. The remarkable result of this decomposition is an exact representation of the original relation in which no redundant data values occur anymore. This magic is possible as the FD $X \to Y$ over T has become a key on the projected schema XY, which ensures that Y-values that occurred redundantly in I are now only stored once in the projected relation $I[XY]$. In our example, the projected relation purchase$[icp]$ over $\{item, catalog, price\}$ stores the two redundant occurrences of the price 240 from purchase only once. Note that the occurrence of 240 in the second tuple of purchase$[icp]$ does not result from a redundant occurrence of 240 in the purchase relation. In addition, purchase$[icp]$ satisfies the key $\{item, catalog\}$ over $\{item, catalog, price\}$. These are the original fundamental ideas for the framework of relational normalization on which we will focus. Further developments of the relational normalization framework, such as other normal forms and classes of data dependencies, are outside the scope of this article. In particular, we defer the treatment of Third normal form and dependency-preservation to future work. We remark that dependency-preserving BCNF decompositions can always be obtained by attribute splitting [30].

Challenges of real-world instances. While Codd's relational model of data provides a sound foundation for relational database systems, the de-facto industry standard SQL exhibits features that have made an extension of the relational normalization theory to SQL-compliant data challenging. For example, SQL permits occurrences of a null marker in database instances, which makes it possible in practice to store and process incomplete information. In our example, catalog information may not exist for some ordered item or be unknown at the time of the order. Both interpretations of incomplete information are represented uniformly by the SQL null marker, here denoted by NULL or $\perp$, thereby consciously trading a loss of information for a gain in the efficiency of data management. The handling of null markers in query answering has been a long-standing problem in database research that has attracted continued interest since the beginnings, with remarkable insights and results produced over the years. While schema design has



order_id	item	catalog	price
5299401	Fitbit Surge	NULL	240
7485113	Fitbit Surge	NULL	200

order_id	item	catalog
5299401	Fitbit Surge	NULL
7485113	Fitbit Surge	NULL

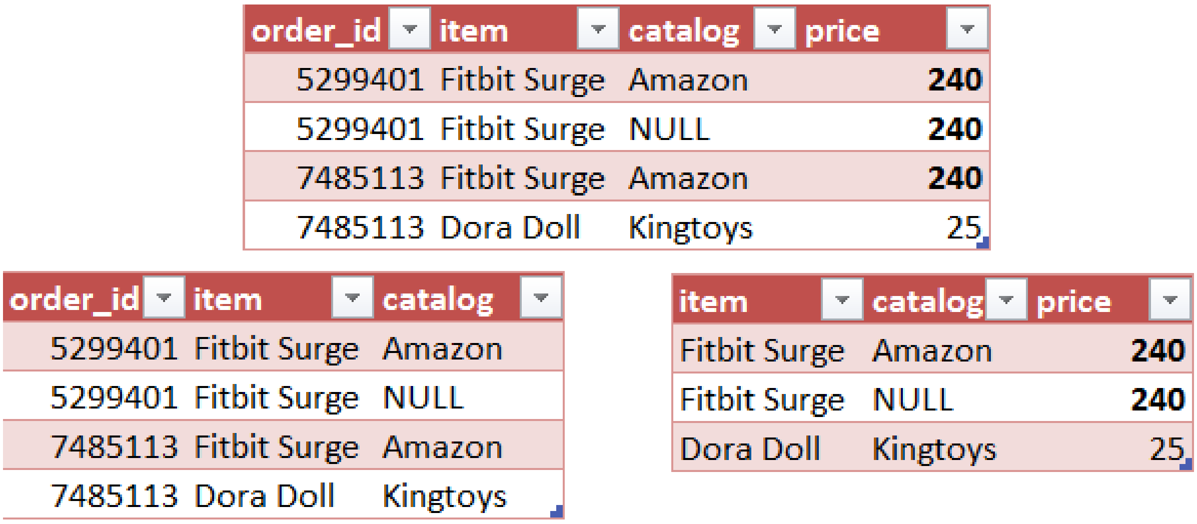
item	catalog	price
Fitbit Surge	NULL	240
Fitbit Surge	NULL	200

Figure 4: Lossy decomposition based on the pFD $\mathit{item}, \mathit{catalog} \; {}_{s}\!\!\rightarrow \mathit{price}$

attracted some research, even the basics as mentioned above have not been adequately addressed so far. An additional challenge is caused by another distinguished feature of SQL. While the relational model is set-based and duplicates must not occur, SQL does permit duplicate tuples. The main reason is the importance of duplicates for aggregation queries, and another reason is that duplicate removal becomes prohibitively expensive in some situations. While duplicates offer opportunities for extending query languages, they incur additional challenges for schema design. We will now illustrate the challenges on our running example.

The first challenge is that the presence of duplicate tuples means that keys cannot be expressed by functional dependencies anymore. When duplicate tuples cannot be present, a relation I over relation schema T satisfies the key kX if and only if I satisfies the FD $X \rightarrow T$. When duplicate tuples may be present, I still satisfies the FD $X \rightarrow T$ whenever it satisfies the key kX, but not vice versa. That is, I may satisfy the FD $X \rightarrow T$ but violate the key kX. A simple and powerful illustration is the instance in Figure 3 which satisfies every FD but violates every key over $\{\mathit{item}, \mathit{catalog}, \mathit{price}\}$. The challenge means that FDs must be studied together with keys, because FDs can incur the presence of data redundancy while keys guarantee their absence. For example, the schema $\{\mathit{item}, \mathit{catalog}, \mathit{price}\}$ with the FD $\mathit{item}, \mathit{catalog} \rightarrow \mathit{price}$ satisfies the traditional BCNF condition, but admits instances (with duplicate tuples) in which redundant data values occur, e.g., the instance above. As SQL permits the specification of attributes as `NOT NULL`, the actual challenge is to study the combined class of keys, FDs, and `NOT NULL` constraints.

The second challenge concerns the definition of the semantics of keys and functional dependencies in instances with duplicate and partial information. The challenge here is to define the semantics in such a way that i) many constraints that hold in an application domain can be expressed, ii) occurrences of null markers are handled appropriately by the dependencies, iii) the dependencies can be reasoned about efficiently, and iv) the dependencies are useful for designing SQL schemata. While the literature on functional dependencies in incomplete databases is rather rich, the only class of functional dependencies that has had a notable impact on schema design is that proposed by Lien [28]. We refer to these as possible FDs (p-FDs) and denote them by $X \; {}_{s}\!\!\rightarrow Y$. An instance



order_id	item	catalog	price
5299401	Fitbit Surge	Amazon	**240**
5299401	Fitbit Surge	NULL	**240**
7485113	Fitbit Surge	Amazon	**240**
7485113	Dora Doll	Kingtoys	25

order_id	item	catalog
5299401	Fitbit Surge	Amazon
5299401	Fitbit Surge	NULL
7485113	Fitbit Surge	Amazon
7485113	Dora Doll	Kingtoys

item	catalog	price
Fitbit Surge	Amazon	**240**
Fitbit Surge	NULL	**240**
Dora Doll	Kingtoys	25

Figure 5: Lossless decomposition and redundant values in bold based on the cFD $\mathit{item}, \mathit{catalog}\ {}_{w}\!\!\rightarrow \mathit{price}$

I satisfies the p-FD $X\ {}_{s}\!\!\rightarrow Y$ if and only if all tuples $t, t' \in I$ have matching values on all the attributes in Y whenever they have strongly similar values on all the attributes in X, that is, when $t[X] \sim_s t'[X] \Rightarrow t[Y] = t'[Y]$. Here, $t[X] \sim_s t'[X]$ means that t and t' have the same matching non-null values on all the attributes in X, that is, $\perp \neq t(A) = t'(A) \neq \perp$ for all $A \in X$. The impact of this definition on schema design can be explained as follows. The X-total part of an instance I over schema T that satisfies the p-FD $X\ {}_{s}\!\!\rightarrow Y$ is the lossless join of the X-total projection $I_X[XY]$ and the X-total projection $I_X[X(T - XY)]$. The usefulness of p-FDs for schema design is therefore limited as a) their lossless join property does not extend to tuples in which null markers occur in X, as the example in Figure 4 illustrates. Indeed, while the given instance I satisfies the p-FD $\mathit{item}, \mathit{catalog}\ {}_{s}\!\!\rightarrow \mathit{price}$, it is not the lossless join of $I[\mathit{oic}]$ and $I[\mathit{icp}]$. In fact, the usefulness of p-FDs for schema design is further limited as b) lossless decompositions do exist for instances with null marker occurrences, and c) p-FDs cannot capture many redundant data value occurrences. We illustrate these deficiencies on the example in Figure 5. Here, the given instance I satisfies a stronger notion of an FD, which we call a *certain FD* or c-FD for short. We say that two tuples t, t' are weakly similar on attribute A if $t(A) = \perp$ or $t(A) = t'(A)$ hold. Indeed, the first and second, as well as the second and third tuples in I are weakly similar on *item*,*catalog* and have matching values on *price*. That is, they satisfy the c-FD $\mathit{item}, \mathit{catalog}\ {}_{w}\!\!\rightarrow \mathit{price}$. This also makes the occurrence of the value **240** in the second tuple redundant. Note the difference: The instance also satisfies the p-FD $\mathit{item}, \mathit{catalog}\ {}_{s}\!\!\rightarrow \mathit{price}$, but the occurrence of the value **240** in the second tuple is not redundant with respect to this p-FD. This illustrates point c) above. To see point b), instance I in Figure 5 is indeed the lossless join of the two projections $I[\mathit{oic}]$ and $I[\mathit{icp}]$, where the join condition is based on matching values (and not weak similarity) on common attributes.

While the new notion of certain FDs is the right notion for SQL schema design, there are further challenges that must be dealt with. While the decomposition of the instance

I in Figure 5 is lossless, it has not eliminated all data redundancy from the projected instance $I[icp]$. Indeed, each occurrence of the value **240** is redundant, as changing one of them to a different value will violate the c-FD *item*, *catalog* $_w\!\rightarrow$ *price*. The reason is that the projection of I onto *icp* did not imply that *item*, *catalog* is a key on $I[icp]$, in contrast to relations. This showcases again the necessity of studying keys and FDs together, but also the need to handle the semantics of keys in instances with null marker occurrences. As it turns out, the notions of possible and certain keys from [21] are very useful in our context. Indeed, a p-key $p\,\langle X\rangle$ (c-key $c\,\langle X\rangle$, respectively) holds on I if there are no two different tuples in I that are strongly (weakly, respectively) similar on all the attributes in X. In instance $I[cip]$ of Figure 5, for example, the first two tuples are weakly, but not strongly similar on *catalog*, which means that $I[cip]$ satisfies the p-key $p\,\langle item, catalog\rangle$ but not the c-key $c\,\langle item, catalog\rangle$. Indeed, the redundant data values **240** occur in $I[cip]$ because this instance does not satisfy the c-key $c\,\langle item, catalog\rangle$.

The major challenges are therefore to identify syntactic conditions that characterize SQL schemata that permit precisely those instances in which no redundant data values occur, and to identify when and how a given SQL schema can be transformed into one that satisfies these conditions.

Contributions. Main contributions of our research are:

1. We introduce the new class of c-FDs as a natural complement to Lien's class of p-FDs. C-FDs can express constraints not expressible by previously studied data dependencies. We show that c-FDs are the right notion for SQL schema design.

2. We establish axiomatic and linear-time algorithmic characterizations for the implication problem of the combined class of p-FDs, c-FDs, p-keys, c-keys, and `NOT NULL` constraints. This combination covers the main features of SQL instances in which duplicate and partial information readily occur. Duplicates make it necessary to include keys as these are no longer subsumed by FDs. Null markers make it necessary to investigate possible and certain variants of these constraints, and also to include `NOT NULL` constraints which largely determine the interaction of these constraints. Our results subsume and unify previous results by Lien on p-FDs [28], by Atzeni/Morfuni on p-FDs and `NOT NULL` constraints [5], and by Köhler/Link/Zhou on p- and c-keys [21], under our combined class of constraints, without time penalties to reason about them. These foundations enable us to propose a normal form for well-designed SQL schemata, and to justify it semantically by the absence of data redundancy.

3. Indeed, we propose a normal form in terms of our constraints and show that it syntactically characterizes SQL schemata that permit precisely those instances in which no data redundancy occurs. Our normal form reduces to the well-known Boyce-Codd normal form in the idealized special case where no duplicate tuples and null markers occur, that is, when all attributes are `NOT NULL` and some key holds on the schema. Our normal form condition is invariant under different representations of the given constraints, and can be verified in time quadratic in the input, thanks to our efficient algorithms to reason about the constraints.

4. In contrast to the idealized relational model, there are cases of SQL schemata that cannot be transformed into ones that satisfy our normal form condition. Nevertheless, we identify total FDs as an expressive sub-class of c-FDs, and combine them with c-keys, to establish a normal form condition that syntactically characterizes SQL schemata that permit precisely those instances in which no redundant data values occur except for redundant null markers. We further establish an algorithm that transforms any given SQL schema with total FDs and c-keys into one that satisfies this normal form condition. Our algorithm reduces to the classical BCNF decomposition algorithm in the idealized special case where all attributes are `NOT NULL` and some key holds on the schema. Thanks to our algorithms to reason about the constraints, our normalization algorithm is as efficient as the classical BCNF decomposition algorithm.

5. We have applied a data mining algorithm to discover c- and p-FDs in publicly available data sets. These experiments show that both c- and p-FDs frequently occur in practice, that c-FDs are often total, and that c-FDs are effective in eliminating data redundancy from real-world data sets, without loss of information.

Organization. Preliminaries are given in Section 2. Related work is discussed in Section 3. Schema design foundations are treated in Section 4. Normal forms are proposed and justified in Section 5. Section 6 introduces SQL schema normalization. Experimental results are presented in Section 7. We conclude in Section 8.

2 Possible and Certain FDs

Our data model and constraints are introduced in this section. Essentially, we will follow the *no information NULL* approach by [5, 17, 18, 21, 28], but extend it by introducing a new type of functional dependency which we call certain. The remainder of the article will establish the significance of certain functional dependencies for SQL schema design.

We begin with basic terminology. Let $\mathfrak{A} = \{A_1, A_2, \ldots\}$ be a (countably) infinite set of distinct symbols, called *attributes*. Attributes represent column names of tables. A *table schema* is a finite non-empty subset T of $\mathfrak{A}$. Each attribute A of a table schema T is associated with an infinite domain $dom(A)$ which represents the possible values that can occur in column A. In order to encompass incomplete information the domain of each attribute contains the null marker, denoted by $\perp$. The interpretation of $\perp$ is to mean "no information" [5, 17, 18, 21, 28]. We stress that the null marker is not a domain value. In fact, it is a purely syntactic convenience that we include the null marker in the domain of each attribute as a distinguished element.

For attribute sets X and Y we may write XY for their set union $X \cup Y$. If $X = \{A_1, \ldots, A_m\}$, then we may write $A_1 \cdots A_m$ for X. In particular, we may write A to represent the singleton $\{A\}$. A *tuple* over T is a function $t : T \to \bigcup_{A \in T} dom(A)$ with $t(A) \in dom(A)$ for all $A \in X$. For $X \subseteq T$ let $t[X]$ denote the restriction of the tuple t over T to X. We say that a tuple t is X*-total* if $t[A] \neq \perp$ for all $A \in X$. A tuple t over T is said to be a *total tuple* if it is T-total. A *table* I over T is a finite multiset of tuples over T. A table I over T is a *total table* if every tuple $t \in I$ is total. Let t, t' be tuples over T. We define *weak/strong similarity* of t, t' on $X \subseteq T$ as follows:

$$\begin{aligned} t[X] \sim_w t'[X] \quad &:\Leftrightarrow \quad \forall A \in X. \\ &\qquad (t[A] = t'[A] \vee t[A] = \perp \vee t'[A] = \perp) \\ t[X] \sim_s t'[X] \quad &:\Leftrightarrow \quad \forall A \in X. \\ &\qquad (t[A] = t'[A] \neq \perp) \end{aligned}$$

Hence, strong similarity of t, t' on X means for each attribute A of X that the two values $t[A]$ and $t'[A]$ are both total and identical, and weak similarity of t, t' on X means for each attribute A of X that the two values $t[A]$ and $t'[A]$ are identical or at least one of them is $\perp$. Weak and strong similarity coincide for tuples that are X-total. In such "classical" cases we denote similarity by $t[X] \sim t'[X]$. We will use the phrase t, t' *agree* interchangeably for t, t' *are similar.*

A *null-free subschema* (NFS) over table schema T is an expression T_S where $T_S \subseteq T$. The NFS T_S over T is satisfied by a table I over T iff I is T_S-total. In SQL, we can declare attributes to be `NOT NULL`, and the set of these attributes forms an NFS over the table schema. We sometimes refer to the pair (T, T_S) as table schema.

We say that $X \subseteq T$ is a *key* for the total table I over T, denoted by $I \vdash X$, if there are no two different tuples $t, t' \in I$ that agree on X. The following notions of possible and certain keys were introduced in [21]. Given a table I on T, we say that $X \subseteq T$ is a *possible/certain key* for I, denoted by $p\langle X\rangle$ and $c\langle X\rangle$ respectively, if no two tuples in I with distinct tuple identities are strongly (weakly) similar on X. In this paper, we introduce the following notions of possible and certain functional dependencies.

Definition 1 (Possible/Certain FD)
Let $X, Y \subseteq T$. We call an expression of the form $X \ {}_s\!\!\rightarrow Y$ a possible functional dependency *(p-FD), and an expression of the form $X \ {}_w\!\!\rightarrow Y$ a* certain functional dependency *(c-FD) on T. A possible (certain) FD $X \ {}_s\!\!\rightarrow Y$ ($X \ {}_w\!\!\rightarrow Y$) holds for a table I over T, if for every pair of tuples $t, t' \in I$ strong (weak) agreement on X implies equality on Y.*

$$I \vdash X \ {}_s\!\!\rightarrow Y :\Leftrightarrow \forall t, t' \in I.\ t[X] \sim_s t'[X] \Rightarrow t[Y] = t'[Y]$$
$$I \vdash X \ {}_w\!\!\rightarrow Y :\Leftrightarrow \forall t, t' \in I.\ t[X] \sim_w t'[X] \Rightarrow t[Y] = t'[Y]$$

Intuitively, possible FDs hold if it is possible to satisfy the FD classically, that is, there is some replacement of $\perp$ occurrences in columns of the LHS by some domain values that will satisfy the FD classically; and certain FDs hold if it is certain to satisfy the FD classically, that is, every replacement of $\perp$ occurrences in columns of the LHS by some domain values will satisfy the FD classically. We do not impose weak or strong similarity for the RHS, in contrast to [24]. Intuitively, the LHS identifies a sub-entity (possibly or certainly), and the (possible or certain) FD expresses that matching sub-entities must have the same RHS value - this may be known or unknown (i.e., $\perp$), but if we know it in one case it should not be stored as unknown in the other. While p-FDs were introduced by Lien [28], and further investigated by Atzeni/Morfuni [5], and Hartmann/Link [18], c-FDs are new to the best of our knowledge.

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

One may wonder at this point whether our definition of c-FDs is practical, that is, whether it allows us to express "desirable" constraints that cannot be expressed with other notions of FDs. For this, consider Example 1.

Example 1 *Consider the following relation where name and appointment are* `NOT NULL`*:*

n(ame)	*d(ob)*	*a(ppointment)*
John Smith	*19/05/1969*	*DB Admin*
John Smith	*01/04/1971*	*Finance Manager*
John Smith	$\bot$	*Programmer*
James Brown	$\bot$	*Programmer*

A reasonable constraint would be that every employee should be uniquely identifiable by their name and date of birth. This is violated in the table above: While rows 1, 2 and 4 clearly refer to distinct employees, it is not clear which of the two John Smiths the third row refers to, if either.

Our constraint can be expressed as the c-FD $\mathit{nd}\ {}_{w}\!\!\rightarrow d$*. To satisfy it, we would have to (e.g.) assign a date of birth to the third row, identifying the employee as one of the two previous John Smiths (or as a distinct third John Smith). None of the previously studied notions of FD can express this constraint, as further explained in Section 3. While the c-key* $c\,\langle \mathit{nd}\rangle$ *comes close, it would prevent an employee from having two appointments, which may not be desirable.*

The example shows that non-trivial c-FDs of the form $X\ {}_{w}\!\!\rightarrow X$ can be practical. Going further, we expect that one will be hard-pressed to find an example where a c-FD of the form $X\ {}_{w}\!\!\rightarrow Y$ is sensible, but $X\ {}_{w}\!\!\rightarrow XY$ is not. In Section 6 we will see that c-FDs, and particularly those of the form $X\ {}_{w}\!\!\rightarrow XY$, enable SQL schema decomposition.

3 Related Work

Schema design is a core research topic for many data models: Relational [26, 31, 34], conceptual [37], nested [23, 34], object-oriented [9, 36], XML [3, 41], and RDF [20, 22]. Our main question asks how to generalize relational schema design to SQL, where duplicate and partial information readily occur, in particular in modern applications. The fact that only little research has been reported in over forty years is already witness to the challenges that this question poses.

As main enabling results we consider the lossless decomposition property of relations that satisfy FDs [35], the Boyce-Codd normal form (BCNF) [10, 19] and its semantic justification [7, 15, 40], the BCNF-decomposition [38], Armstrong's axioms [4] and linear-time algorithms to decide FD implication [6, 13] as tools that facilitate decompositions [8].

We review the most significant extensions of classical FDs in data models with partial information. The two most prolific interpretations of the null marker are "value unknown at present" [11] and "no information" [28]. First we discuss the former interpretation, which has a possible world semantics as its foundation but cannot express non-existing data. In [39] a three-valued model of FD satisfaction is explored. Here, all possible worlds over an instance I are considered, and an FD either holds, does not hold, or may hold on I, iff it holds for all, none or some (but not all) possible worlds, respectively. In [24] the notions of weak and strong FDs are introduced. A weak FD holds on some possible world, while a strong FD holds on every possible world. The authors show that the combined class of weak and strong FDs does not enjoy any finite axiomatization. In [25] the same authors study several database design problems for partial relations. Here, a set of FDs is satisfied by such a partial relation, if there is some possible world in which all FDs of the given set are satisfied. Under such assumption, the issue of decomposing partial relations is avoided altogether. This, however, is the main focus in practice and of our current work. Secondly, we discuss proposals related to the "no information" approach, which encompasses unknown as well as non-existing data [28] and has therefore been adopted by SQL. Lien defined FDs in this context to hold if strong similarity on the LHS of an FD implies equality on its RHS [28]. Our p-FDs thus correspond to Lien's notion of an FD. P-FDs enjoy a partial decomposition theorem in the sense that the X-total projection of a partial relation I over schema T that satisfies the p-FD $X \to Y$ is the lossless join of the X-total projections of $I[XY]$ and $I[X(T-XY)]$. The work in [16] proposed an SQL normal form with respect to p-FDs, but lossless decompositions are only achievable with respect to p-FDs $X \ {}_{s}{\to}\ Y$ where all attributes in X are `NOT NULL`. What we show in our work is that c-FDs enjoy a full decomposition theorem. Atzeni/Morfuni characterized the implication problem for p-FDs together with `NOT NULL` constraints [5], and Hartmann/Link extended this work to include "possible" multivalued dependencies [17, 18]. Our notions of p-FDs and c-FDs are built on the notions of p-keys and c-keys recently introduced by Köhler/Link/Zhou [21]. Since duplicate information is present, p-FDs and c-FDs cannot express p-keys or c-keys. Hence it is necessary to consider all four notions together to accommodate partial and duplicate information that occur in SQL data.

Our main novelty are certain FDs (Definition 1) which have not been studied beforehand. Our results show that c-FDs are the right notion for SQL schema design. Previously studied notions of FDs are not suitable for decompositions. Indeed, c-FDs allow us to generalize the main enabling results from relational to SQL schema design. This is impossible to achieve with the other notions of FDs, as they cannot express many constraints that can be expressed by c-FDs. A brief comparison of the different notions is given by the following example.

Example 2 *Consider the following relation:*

e(mployee)	*d(ept)*	*m(anager)*	*s(alary)*
Turing	*CS*	*von Neumann*	$\perp$
Turing	$\perp$	*Gödel*	$\perp$

Functional dependencies on this relation are satisfied (T) or violated (F) by the different *definitions as follows:*

	[39]	*[24]*	*[24]*	*[28]*	*here*
		weak	*strong*	*possible*	*certain*
$e \rightarrow d$	*unk*	*T*	*F*	*F*	*F*
$e \rightarrow m$	*F*	*F*	*F*	*F*	*F*
$e \rightarrow s$	*unk*	*T*	*F*	*T*	*T*
$d \rightarrow d$	*T*	*T*	*T*	*T*	*F*
$d \rightarrow m$	*unk*	*T*	*F*	*T*	*F*
$m \rightarrow e$	*T*	*T*	*T*	*T*	*T*
$m \rightarrow d$	*unk*	*T*	*T*	*T*	*T*

Note that the c-FD $d \ {}_{w}\!\!\rightarrow d$ *does* not *hold.*

In summary, the novel notion of a certain functional dependency allows us to develop the first schema design approach to SQL that generalizes relational schema design. C-FDs ensure that the problems studied in this paper are substantially different from those in the research literature.

4 Schema Design Foundations

The same way classical relation schema design is founded on the ability to efficiently reason about FDs, we will see that SQL schema design is founded on the ability to efficiently reason about the combined class of p-keys, c-keys, p-FDs, c-FDs and `NOT NULL` constraints. This chapter establishes these reasoning tools. Many other data management tasks, including data profiling, transaction processing, and query optimization, benefit from the ability to decide the implication problem of semantic constraints.

In our context, the implication problem can be defined as follows. Let (T, T_S) denote the schema under consideration. For a set $\Sigma \cup \{\varphi\}$ of constraints over (T, T_S) we say that Σ *implies* φ, denoted $\Sigma \models \varphi$, iff every table over (T, T_S) that satisfies Σ also satisfies φ. The *implication problem* for a class $\mathcal{C}$ of constraints is to decide, for arbitrary (T, T_S) and $\Sigma \cup \{\varphi\}$ in $\mathcal{C}$, whether Σ implies φ.

We will first establish axiomatic and algorithmic characterizations of the implication problem for the class of p- and c-FDs in the presence of `NOT NULL` constraints, and then show how to add p- and c-keys.

4.1 Possible and Certain FDs

Axioms facilitate our understanding and help us reason about constraints. In our case, the axioms enable us to establish a semantic justification for our normal form proposal of well-designed SQL schemata in Section 5. The definitions of *sound* and *complete* sets of axioms are standard [34]. Table 1 shows the set $\mathfrak{F}$ of inference rules which forms a sound and complete axiomatization for p-FDs, c-FDs and `NOT NULL` constraints. Here, $X \rightarrow Y$ is to mean either $X \ {}_{s}\!\!\rightarrow Y$ or $X \ {}_{w}\!\!\rightarrow Y$, substituted uniformly for each rule. That is, either all unspecified FDs $X \rightarrow Y$ in a rule are certain, or all are possible.

Reflexivity (R):	$\dfrac{}{X \ {}_{s}\!\!\rightarrow X}$
L-Augmentation (A):	$\dfrac{X \rightarrow Y}{XZ \rightarrow Y}$
Strengthening (S):	$\dfrac{X \ {}_{s}\!\!\rightarrow Y}{X \ {}_{w}\!\!\rightarrow Y} \ X \subseteq T_S$
Union (U):	$\dfrac{X \rightarrow Y \quad X \rightarrow Z}{X \rightarrow YZ}$
Decomposition (D):	$\dfrac{X \rightarrow YZ}{X \rightarrow Y}$
Pseudo-Transitivity (T):	$\dfrac{X \rightarrow Y \quad XY \ {}_{w}\!\!\rightarrow Z}{X \rightarrow Z}$
Null-Transitivity (NT):	$\dfrac{X \ {}_{s}\!\!\rightarrow Y \quad XY \ {}_{s}\!\!\rightarrow Z}{X \ {}_{s}\!\!\rightarrow Z} \ Y \subseteq T_S$

Table 1: Axiomatization $\mathfrak{F}$ of p/c-FDs and `NOT NULL`s

Theorem 1 *The set $\mathfrak{F}$ of inference rules from Table 1 are sound and complete for the implication of certain and possible functional dependencies and `NOT NULL` constraints.*

Given PURCHASE $= T = oicp$ with $T_S = ocp$, let Σ consist of the p-FD $oi \ {}_{s}\!\!\rightarrow c$ and the c-FD $ic \ {}_{w}\!\!\rightarrow p$. Applying L-augmentation to $ic \ {}_{w}\!\!\rightarrow p$ results in $oic \ {}_{w}\!\!\rightarrow p$, and applying pseudo-transitivity to $oi \ {}_{s}\!\!\rightarrow c$ and $oic \ {}_{w}\!\!\rightarrow p$ results in $oi \ {}_{s}\!\!\rightarrow p$. As $\mathfrak{F}$ is sound, we conclude that Σ implies $oi \ {}_{s}\!\!\rightarrow p$. For deciding if Σ implies the c-FD $oi \ {}_{w}\!\!\rightarrow p$, we could check if $oi \ {}_{w}\!\!\rightarrow p$ can be inferred from Σ using $\mathfrak{F}$. This is inefficient and does not make good use of the given input $oi \ {}_{w}\!\!\rightarrow p$.

The notion of an attribute closure was instrumental in deriving a linear-time decision algorithm for traditional FDs [6]. Here, we introduce the notions of p- and c-closures, and show that they serve the same purpose in the SQL context.

Definition 2 (closure)
Let (T, T_S, Σ) be a schema with Σ containing c-FDs and p-FDs. The p-closure (c-closure) *of $X \subseteq T$ is the set of all $A \in T$ such that $X \ {}_{s}\!\!\rightarrow A$ ($X \ {}_{w}\!\!\rightarrow A$) is implied by Σ:*

$$X^{*p}_{\Sigma,T_S} := \{A \in T \mid \Sigma \models X \ {}_{s}\!\!\rightarrow A\}$$
$$X^{*c}_{\Sigma,T_S} := \{A \in T \mid \Sigma \models X \ {}_{w}\!\!\rightarrow A\}$$

We simply write X^{*p}, X^{*c} when Σ, T_S is understood.

Classically, an FD $X \rightarrow Y$ is implied by an FD set Σ iff Y is a subset of the attribute closure of X. We establish corresponding properties for p- and c-closures. Hence, the implication problem reduces to their computations.

Theorem 2 *Let (T, T_S, Σ) be a schema with Σ containing p-FDs and c-FDs. Then Σ implies $X \ {}_{s}\!\!\rightarrow Y$ if and only if $Y \subseteq X^{*p}$, and Σ implies $X \ {}_{w}\!\!\rightarrow Y$ if and only if $Y \subseteq X^{*c}$.*

Algorithm 1 p-Closure

```
Input: $\Sigma, T_S, X$
Output: $X^{*p}$
$\mathcal{C} := X$
repeat
    $\mathcal{C}_{\text{old}} := \mathcal{C}$
    for all $Y \;_{w}\!\!\rightarrow Z \in \Sigma$ with $Y \subseteq \mathcal{C}$ do
        $\mathcal{C} := \mathcal{C} \cup Z$
    for all $Y \;_{s}\!\!\rightarrow Z \in \Sigma$ with $Y \subseteq (\mathcal{C} \cap T_S) \cup X$ do
        $\mathcal{C} := \mathcal{C} \cup Z$
until $\mathcal{C} = \mathcal{C}_{\text{old}}$
return $\mathcal{C}$
```

Algorithm 2 c-Closure

```
Input: $\Sigma, T_S, X$
Output: $X^{*c}$
$\mathcal{C} := X \cap T_S$
repeat
    $\mathcal{C}_{\text{old}} := \mathcal{C}$
    for all $Y \;_{w}\!\!\rightarrow Z \in \Sigma$ with $Y \subseteq \mathcal{C} \cup X$ do
        $\mathcal{C} := \mathcal{C} \cup Z$
    for all $Y \;_{s}\!\!\rightarrow Z \in \Sigma$ with $Y \subseteq \mathcal{C} \cap T_S$ do
        $\mathcal{C} := \mathcal{C} \cup Z$
until $\mathcal{C} = \mathcal{C}_{\text{old}}$
return $\mathcal{C}$
```

In contrast to the relational model, neither $(\cdot)^{*c}$ nor $(\cdot)^{*p}$ is an actual closure operator: X^{*c} need not contain X, and $(X^{*p})^{*p} = X^{*p}$ does not hold in general. Nevertheless we will keep referring to them as closures for the sake of consistency with existing terminology. Indeed, the following properties are important in establishing the soundness of algorithms for the computation of p- and c-closures.

Lemma 1 *The following statements holds: i) If $X \subseteq Y$ then $X^{*p} \subseteq Y^{*p}$ and $X^{*c} \subseteq Y^{*c}$, ii) $X, X^{*c} \subseteq X^{*p}$, and iii) $(X^{*c})^{*c} \subseteq X^{*c}$ and $(X^{*p})^{*c} \subseteq X^{*p}$.*

Algorithms 1 and 2 compute the p- and c-closures of a given attribute set for a given set of p- and c-FDs. Both algorithms are provably correct and operate in quadratic time. In both cases, linear time complexity is achieved by applying the optimization techniques of [6].

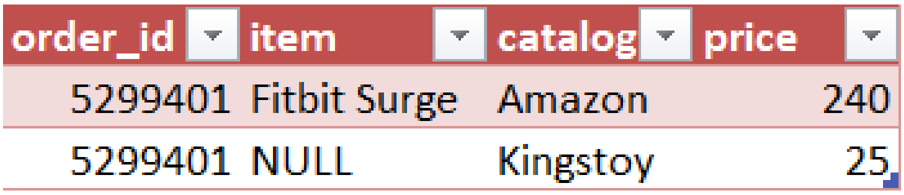
Theorem 3 *Algorithms 1 and 2 are correct, and the implication problem for the combined class of p-FDs, c-FDs, and `NOT NULL` constraints can be decided in linear time.*

Recall our schema PURCHASE $= oicp$ with $T_S = ocp$ and where Σ consists of the p-FD $oi \;_{s}\!\!\rightarrow c$ and the c-FD $ic \;_{w}\!\!\rightarrow p$. We saw that $oi \;_{s}\!\!\rightarrow p$ is implied by Σ, which is

key-Augmentation (kA):	$\dfrac{(p/c)\,\langle X\rangle}{(p/c)\,\langle XY\rangle}$
key-Strengthening (kS):	$\dfrac{p\,\langle X\rangle}{c\,\langle X\rangle}\; X\subseteq T_S$
key-Weakening (kW):	$\dfrac{c\,\langle X\rangle}{p\,\langle X\rangle}$

Table 2: The axiomatization $\mathfrak{K}$ for p/c-keys

confirmed by the p-closure $oi^{*p} = oicp$ of oi for Σ, and checking that $p \in oi^{*p}$. To see if $oi \ {}_w\!\!\rightarrow p$ is implied by Σ, we compute the c-closure $oi^{*c} = o$ of oi for Σ and realize that $p \notin oi^{*c}$. Hence, Σ does not imply $oi \ {}_w\!\!\rightarrow p$. This is confirmed by the following instance over (T, T_S, Σ) which violates $oi \ {}_w\!\!\rightarrow p$.



order_id	item	catalog	price
5299401	Fitbit Surge	Amazon	240
5299401	NULL	Kingstoy	25

4.2 Interaction with Keys

As FDs cannot express keys over instances with duplicate tuples, and keys are fundamental in schema design, we extend our results to include p-keys and c-keys. For this purpose, we recall the axiomatization $\mathfrak{K}$ from [21] for p-keys, c-keys, and `NOT NULL` constraints, as shown in Table 2. Here, $(p/c)\,\langle X\rangle$ means either $p\,\langle X\rangle$ or $c\,\langle X\rangle$, substituted uniformly in the key-Augmentation rule.

We now target an axiomatization for p-keys, c-keys, p-FDs, c-FDs, and `NOT NULL` constraints together. We use the rules in $\mathfrak{F}$ for p-FDs and c-FDs, $\mathfrak{K}$ for p-keys and c-keys, and the rules in $\mathfrak{FK}$ from Table 3 that capture the interaction of p/c-keys and p/c-FDs. Here, $X \rightarrow Y$ is to mean either $X \ {}_s\!\!\rightarrow Y$ or $X \ {}_w\!\!\rightarrow Y$, while $(p/c)\,\langle X\rangle$ is to mean either $p\,\langle X\rangle$ or $c\,\langle X\rangle$, substituted uniformly for each rule. That is, either all unspecified keys and FDs are certain, or all are possible.

Theorem 4 *The rules from sets $\mathfrak{F}$ in Table 1, $\mathfrak{K}$ from Table 2, and $\mathfrak{FK}$ from Table 3 together are sound and complete for the implication of p-keys, c-keys, p-FDs, c-FDs, and `NOT NULL` constraints.*

Given Purchase $= oicp$ with $T_S = ocp$, let Σ consist of the p-FD $oi \ {}_s\!\!\rightarrow c$ and the p-key $p\,\langle oic\rangle$. An application of key-Null-transitivity to both elements of Σ is possible as $c \in T_S$, and results in $p\,\langle oi\rangle$, which is thus implied by Σ. The question remains how to decide implication efficiently, e.g., how to decide if Σ implies $oi \ {}_w\!\!\rightarrow p$.

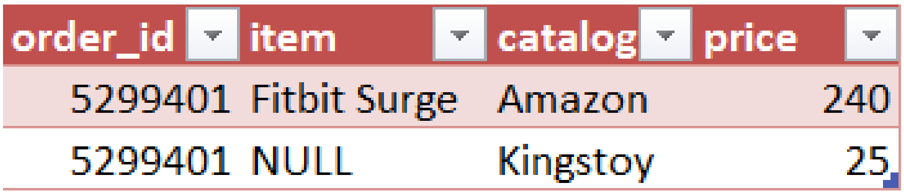
Theorem 4 can be shown by reducing the implication problem to the class of p/c-keys and the class of p/c-FDs alone. The reduction is embodied in the following definition.

key-FD-Weakening (kfW):	$\dfrac{(p/c)\,\langle X\rangle}{X \to Y}$
key-Transitivity (kT):	$\dfrac{X \to Y \quad c\,\langle XY\rangle}{(p/c)\,\langle X\rangle}$
key-Null-Trans. (kNT):	$\dfrac{X \;{}_s\!\!\to Y \quad p\,\langle XY\rangle}{p\,\langle X\rangle}\; Y \subseteq T_S$

Table 3: Set $\mathfrak{FK}$ of inference rules for the interaction of p/c-keys, p/c-FDs and `NOT NULL` constraints

Definition 3 (FD-projection/key-projection)
Let Σ be a set of p-keys, c-keys, p-FDs, and c-FDs over T. The FD-projection *of Σ, denoted by $\Sigma|_{FD}$, is obtained by replacing each key X with an FD $X \to T$: $\Sigma|_{FD} := \{X \to Y \in \Sigma\} \cup \{X \;{}_s\!\!\to T \mid p\,\langle X\rangle \in \Sigma\} \cup \{X \;{}_w\!\!\to T \mid c\,\langle X\rangle \in \Sigma\}$. The* key-projection *$\Sigma|_{key}$ of Σ contains all keys in Σ: $\Sigma|_{key} := \{(p/c)\,\langle X\rangle \in \Sigma\}$.*

FD-projection reduces the implication of an FD by a set of keys and FDs to the implication of an FD by a set of FDs alone: For a p- or c-FD $X \to Y$ over (T, T_S) and a set Σ of p-keys, c-keys, p-FDs, and c-FDs over (T, T_S), Σ implies $X \to Y$ iff $\Sigma|_{\rm FD}$ implies $X \to Y$. Given PURCHASE $= oicp$ with $T_S = ocp$ and $\Sigma = \{oi \;{}_s\!\!\to c, p\,\langle oic\rangle\}$, we have $\Sigma|_{\rm FD} = \{oi \;{}_s\!\!\to c, oic \;{}_s\!\!\to p\}$. Now, Σ implies $oi \;{}_w\!\!\to p$ iff $\Sigma|_{\rm FD}$ implies $oi \;{}_w\!\!\to p$. However, the latter is true iff $p \in (oi)^{*c} = o$. Consequently, Σ does not imply $oi \;{}_w\!\!\to p$, as the following instance witnesses:



order_id	item	catalog	price
5299401	Fitbit Surge	Amazon	240
5299401	NULL	Kingstoy	25

Key-projection reduces the implication of a key by a set of keys and FDs to the implication of a key by a set of keys alone: (i) Σ implies $p\,\langle X\rangle$ iff $\Sigma|_{\rm key}$ implies $c\left\langle X^{*p}_{\Sigma|_{\rm FD}}\right\rangle$ or $p\left\langle X(X^{*p}_{\Sigma|_{\rm FD}} \cap T_S)\right\rangle$, and (ii) Σ implies $c\,\langle X\rangle$ iff $\Sigma|_{\rm key}$ implies $c\left\langle XX^{*c}_{\Sigma|_{\rm FD}}\right\rangle$. Given PURCHASE $= oicp$ with $T_S = ocp$ and $\Sigma = \{oi \;{}_s\!\!\to c, p\,\langle oic\rangle\}$, we have $\Sigma|_{\rm key} = \{p\,\langle oic\rangle\}$. By (i), Σ implies $p\,\langle oi\rangle$ if $\Sigma|_{\rm key}$ implies $p\left\langle oi(oi^{*p}_{\Sigma|_{\rm FD}} \cap ocp)\right\rangle = p\,\langle oic\rangle$. Consequently, Σ does imply $p\,\langle oi\rangle$. Our reduction and our closure algorithms entail the following result.

Theorem 5 *The implication problem for the combined class of p-keys, c-keys, p-FDs, c-FDs, and `NOT NULL` constraints can be decided in linear time of the input.*

5 Normal Forms

Generalizing from the relational model, we stipulate the absence of data redundancy from any database instance as a semantic criterion for an SQL schema to be well-designed. Using our axiomatization and linear-time decidability of the implication problem from Section 4 we establish an effective and efficient syntactic characterization for well-designed SQL schemata in the form of a Boyce-Codd normal form (BCNF) proposal, based on p-keys, c-keys, p-FDs, c-FDs, and `NOT NULL` constraints.

5.1 Normal Form Definitions

We first define the notion of data redundancy, and say that a schema is in Redundancy-free normal form if it only permits instances in which no data redundancy occurs. Our notion of data redundancy follows Vincent's seminal approach [40], which is independent from the type of constraint. Informally, an occurrence of a value in an instance is redundant if *every* change to a different value results in the violation of some given constraint.

Definition 4 (redundancy)
Let (T, T_S, Σ) be a schema, I an instance over it, and p_0 a position *(row and column) in I. A p_0-*value substitution *of I is an instance I' over (T, T_S, Σ) which differs from I precisely in the value at position p_0. We say that the value at position p_0 is* redundant *in I if I possesses no p_0-value substitution. We call I* redundancy-free *if it contains no redundant positions. We say that the schema (T, T_S, Σ) is in* Redundancy-free normal form *(RFNF) if and only if all instances over the schema are redundancy-free.*

Definition 4 stipulates what we want from a schema but does not allow us to identify schemata that meet the conditions of RFNF since there can be infinitely many instances over the schema. However, finding some instance with some redundant position tells us that the given schema is not in RFNF. For example, the schema $\textsc{Purchase} = oicp$ with $\textsc{Purchase}_S = oip$ and $\Sigma = \{ic \ {}_w\!\!\rightarrow p\}$ is not in RFNF: the top instance in Figure 5 contains redundant positions (marked bold). In the relational model, BCNF characterizes syntactically those schemata that are in RFNF [40]. The BCNF condition for schema (T, Σ) says that for every non-trivial FD $X \to Y$ that can be inferred from Σ by the Armstrong axioms, the FD $X \to T$ can also be inferred from Σ by the Armstrong axioms. Using our axiomatization for p-keys, c-keys, p-FDs, and c-FDs, we can extend the BCNF condition from relational to SQL databases. Naturally, it requires the left-hand sides of all inferable and non-trivial p- and c-FDs to be inferable p- and c-keys, respectively.

Definition 5 (BCNF)
Let (T, T_S, Σ) be a schema, with Σ consisting of p-keys, c-keys, p-FDs, and c-FDs. We say that (T, T_S, Σ) is in Boyce-Codd Normal Form (BCNF) *if and only if (i) for all non-trivial p-FDs $X \ {}_s\!\!\rightarrow Y \in \Sigma^+$, $p\langle X\rangle \in \Sigma^+$, and (ii) for all non-trivial c-FDs $X \ {}_w\!\!\rightarrow Y \in \Sigma^+$, $c\langle X\rangle \in \Sigma^+$.*

Definition 5 is purely syntactic using our axiomatization for p-keys, c-keys, p-FDs, c-FDs, and `NOT NULL` constraints. Another important aspect of Definition 5 is its invariance under different representations of the given constraints. Indeed, two schemata are *equivalent* iff they have the same set of instances. For equivalent schemata (T, T_S, Σ_1) and (T, T_S, Σ_2), (T, T_S, Σ_1) is in BCNF iff (T, T_S, Σ_2) is in BCNF. In fact, (T, T_S, Σ_1)

and (T, T_S, Σ_2) are equivalent iff the syntactic closures Σ_1^+ and Σ_2^+ are the same, i.e., the sets of constraints that can be inferred from Σ by a sound and complete axiomatization. So, Definition 5 ensures invariance under different constraint representations, but means we cannot directly decide if a given schema is in BCNF. Fortunately, we can show the following result.

Theorem 6 *The schema (T, T_S, Σ) is in BCNF if and only if (i) for all non-trivial $X\ {}_{s}\!\!\rightarrow Y \in \Sigma$, $p\,\langle X\rangle \in \Sigma^+$, and (ii) for all non-trivial $X\ {}_{w}\!\!\rightarrow Y \in \Sigma$, $c\,\langle X\rangle \in \Sigma^+$.*

Theorem 6 shows that input p-FDs and c-FDs suffice to decide if a given schema is in BCNF. The linear-time decidability of the underlying implication problem gives us a quadratic upper time bound to decide the BCNF condition.

Theorem 7 *The problem whether a given schema is in BCNF can be decided in time quadratic in the input.*

The schema $\textsc{Purchase} = oicp$ with $\textsc{Purchase}_S = oip$ and $\Sigma = \{ic\ {}_{w}\!\!\rightarrow p\}$ is not in BCNF as the c-FD $ic\ {}_{w}\!\!\rightarrow p$ is non-trivial, and the c-key $c\,\langle ic\rangle$ is not implied by Σ. Given $\textsc{Purchase}_S = \emptyset$ and $\Sigma = \{oic\ {}_{w}\!\!\rightarrow p, c\,\langle oicp\rangle\}$ instead the schema is indeed in BCNF: the c-key $c\,\langle oic\rangle$ is implied by Σ because $p \in (oic)^{*c}_{\Sigma|_{\mathrm{FD}}}$.

Theorem 7 ensures that we can decide efficiently if a given schema is in BCNF. However, deciding if the projection of a given schema (T, T_S, Σ) onto a given attribute set $X \subseteq T$ is in BCNF is co-NP complete. Here, the projection of a given schema (T, T_S, Σ) onto a given attribute set $X \subseteq T$ is defined as the schema $(X, X \cap T_S, \Sigma[X])$ where $\Sigma[X] = \{Y \rightarrow Z \in \Sigma^+ \mid YZ \subseteq X\} \cup \{{}_{(p/c)}\,\langle Y\rangle \in \Sigma^+ \mid Y \subseteq X\}$.

Theorem 8 *The problem of deciding whether the projection of a given schema onto a given attribute set is in BCNF (RFNF) is co-NP complete.*

5.2 Semantic Justification

We now pinpoint what our BCNF proposal achieves: It captures schemata that permit exactly those instances that are free from data redundancy.

Theorem 9 *A schema (T, T_S, Σ) is in RFNF if and only if (T, T_S, Σ) is in BCNF.*

It is a corollary of Theorem 7 and Theorem 9 that we can decide efficiently whether a given schema is in RFNF.

Theorem 10 *The problem whether a given schema is in RFNF can be decided in time quadratic in the input.*

$\textsc{Purchase} = oicp$ with $\textsc{Purchase}_S = oip$ and $\Sigma = \{ic\ {}_{w}\!\!\rightarrow p\}$ is not in RFNF, as it is not in BCNF. However, $\textsc{Purchase}$ with $\textsc{Purchase}_S = \emptyset$ and $\Sigma = \{oic\ {}_{w}\!\!\rightarrow p, c\,\langle oicp\rangle\}$ is in RFNF, since it is in BCNF as observed earlier.

The proof of Theorem 9 requires us to construct instances with data redundancy whenever the BCNF condition is violated. This is achieved by the following lemma.

Lemma 2 (construction lemma) *Consider a schema (T, T_S, Σ) and table $I = \{t_0, t_1\}$ over (T, T_S).*

i) Let $\Sigma \nvDash p\,\langle X \rangle$ and t_0, t_1 constructed as

$$t_i[A] = \begin{cases} 0 & \textit{if } A \in X^{*p} \textit{ and } A \in X \cup T_S \\ \perp & \textit{if } A \in X^{*p} \textit{ and } A \notin X \cup T_S \\ i & \textit{if } A \notin X^{*p} \end{cases}$$

Then Σ holds on I.

ii) Let $\Sigma \nvDash c\,\langle X \rangle$ and t_0, t_1 constructed as

$$t_i[A] = \begin{cases} 0 & \textit{if } A \in XX^{*c} \textit{ and } A \in T_S \\ \perp & \textit{if } A \in XX^{*c} \textit{ and } A \notin T_S \\ i & \textit{if } A \notin XX^{*c} \end{cases}$$

Then Σ holds on I.

6 SQL Schema Normalization

In practice we cannot expect that our first attempt at designing a suitable SQL schema meets the BCNF condition. Hoping to achieve BCNF with a new attempt is doomed for failure and a waste of resources. Instead, it is desirable to transform the current schema design into one that meets the normal form condition. This process is well-known as normalization, whose primary goal is the elimination of data redundancy. In relational databases, one can always transform a given schema into BCNF, without loss of information. The two enabling properties for this lossless BCNF decomposition process are that i) T-relations which exhibit an FD $X \to Y$ are the lossless join of their projections on XY and $X(T - XY)$, and ii) if X was not a key over T, then X will be a key over the schema XY. We now show that normalization of SQL schemata is more involved. First we show that c-FDs enable us to generalize the decomposition theorem from relations to SQL instances, using multiset/set-projections. Unlike the idealized special case of relations, decompositions do not always ensure that the resulting projected schema is in BCNF. For this purpose, we identify *total* FDs as an expressive sub-class of c-FDs that ensure the elimination of redundant data value occurrences different from the null marker. As the null marker is not a data value, we re-define RFNF to exclude null markers from redundant data value occurrences and term this normal form *Value redundancy-free normal form* (VRNF). We then propose *SQL-BCNF* as a normal form that characterizes VRNF syntactically. SQL-BCNF is invariant under different representations of constraint sets, and can be decided in quadratic time. Finally, we establish an algorithm that transforms a given schema into one in VRNF (SQL-BCNF).

6.1 Lossless Decompositions

In relational databases, FDs play a vital role in schema decomposition. The central decomposition theorem states that, given an FD $X \to Y$ over table schema T, we can decompose T into $X(T-XY)$ and XY without losing information, i.e., every instance I that satisfies $X \to Y$ also satisfies $I = I[X(T-XY)] \bowtie I[XY]$. Here, $I[X]$ denotes the set projection of I onto X. SQL instances can be multi-sets and we thus require different notions of projection.

Definition 6 (multi-set notations)
We denote multi-sets using the notation $\{\{a, a, b\}\}$. *Let* I *be a table over* T, *and* $X \subseteq T$. *We denote the* multi-set projection *of* I *onto* X *by* $I[[X]] := \{\{t[X] \mid t \in I\}\}$. *As usual, set projection is denoted by* $I[X]$.

Accordingly, sub-schemata of a decomposition may contain set as well as multi-set projections.

Definition 7 (decomposition)
Let T *be a schema and* $\mathcal{D} = \{[T_i], \ldots, [[T_j]], \ldots\}$ *consist of a set of sub-schemata with* $\bigcup \mathcal{D} = T$, *where the different notations indicate set and multi-set projection, respectively. Then* $\mathcal{D}$ *is a* decomposition *of* T.

Similar to the relational model, we can introduce lossless decompositions of instances and schemata. Intuitively here, a decomposition is lossless when it recovers the original instance by joining the projected components of its decomposition. We mean by join an *equality join*, that is we only require equality of data values on common attributes and not strong similarity.

Definition 8 (lossless)
We call a decomposition of an instance I lossless *if a join of its components results in* I. *We call a schema decomposition lossless if it induces a lossless decomposition for all instances. We call a schema decomposition a BCNF decomposition if all its components are in BCNF.*

Lien studied only p-FDs and showed that lossless decomposition can only be done for p-FDs $X\ {}_{s}\!\!\to Y$ where $X \subseteq T_S$ [28]. Hence, we will focus on c-FDs. Theorem 11 shows that c-FDs $X\ {}_{w}\!\!\to Y$ over T enable lossless schema decompositions into $[[X(T-XY)]]$ and $[XY]$. It subsumes the classical decomposition theorem for relations.

Theorem 11 *Let* (T, T_S, Σ) *be a schema with* $\Sigma \models X\ {}_{w}\!\!\to Y$, *and* I *an instance over* (T, T_S, Σ). *Then the following holds:* $I = I[[X(T-XY)]] \bowtie I[XY]$.

For example, consider PURCHASE with c-FD $ic\ {}_{w}\!\!\to p$ and the top instance `purchase` from Figure 5. The decomposition of `purchase` into `purchase`[[oic]] and `purchase`[icp], as shown at the bottom of Figure 5, is lossless.

Theorem 11 shows the significance of c-FDs for achieving lossless SQL schema decompositions. However, the goal of schema design is to eliminate data redundancy. In our example, `purchase`[icp], shown at the bottom right of Figure 5, still contains redundant data value occurrences. The reason is that `purchase`[icp] does not satisfy the c-key $c\,\langle ic\rangle$. In fact, $c\,\langle ic\rangle$ does not hold on `purchase`[icp] because the c-FD $ic \;{}_{w}\!\!\rightarrow icp$ does not hold on `purchase`, even though $ic \;{}_{w}\!\!\rightarrow p$ does hold. Intuitively, the next result shows that c-FDs of the form $X \;{}_{w}\!\!\rightarrow XY$ eliminate data redundancy on the $[XY]$ component of a lossless decomposition.

Theorem 12 *Let (T, T_S, Σ) be a schema with $\Sigma \models X \;{}_{w}\!\!\rightarrow Y, X \cap Y = \emptyset$, and I an instance over (T, T_S, Σ). If $\Sigma \models X \;{}_{w}\!\!\rightarrow XY$, then $c\,\langle X\rangle$ holds on $I[XY]$.*

Theorem 12 establishes the significance of c-FDs $X \;{}_{w}\!\!\rightarrow XY$ for SQL schema design. Accordingly, we name them.

Definition 9 (total FD)
We call a certain FD of the form $X \;{}_{w}\!\!\rightarrow XY$ total.

We will focus on schema decompositions based on total FDs and c-keys. As this includes p-FDs $X \;{}_{s}\!\!\rightarrow Y$ where $X \subseteq T_S$, our decomposition approach generalizes that for p-FDs [28]. Still, our initial goal of obtaining lossless BCNF decomposition is elusive, even if we consider only total FDs.

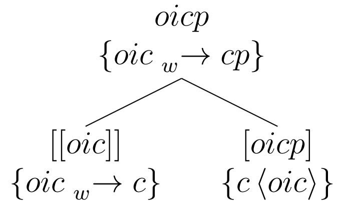
Theorem 13 *There are schemata for which no lossless BCNF decomposition exists, even if only total FDs are given.*

The following example provides a proof for Theorem 13.

Example 3 *Consider the following variation of our running example:*

$$(oicp,\ oip,\ \{oic \;{}_{w}\!\!\rightarrow cp\}).$$

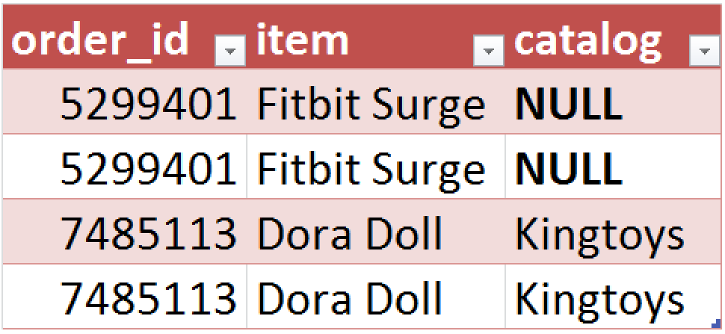
Here, the same item may be ordered from different catalogues at different prices (e.g. promotions with purchase limits), as long as all catalogues are known. Applying Theorem 11, we get


$oicp$
$\{oic \;{}_{w}\!\!\rightarrow cp\}$
$[[oic]]$
$\{oic \;{}_{w}\!\!\rightarrow c\}$
$[oicp]$
$\{c\,\langle oic\rangle\}$

Here the sub-schema oic contains a non-trivial c-FD but no keys, and cannot be decomposed further.

6.2 Eliminating Value Redundancy

Theorem 13 showed that decomposing SQL schemata into BCNF without loss of information is elusive. While this makes it impossible to always eliminate all data redundancy, it is useful to investigate the type of data redundancy that resists elimination. For this purpose, consider the following instance over $[oic]$ from Example 3:



order_id	item	catalog
5299401	Fitbit Surge	**NULL**
5299401	Fitbit Surge	**NULL**
7485113	Dora Doll	Kingtoys
7485113	Dora Doll	Kingtoys

Here, the only redundant positions are those in which the null marker NULL ($\perp$) occurs. Substituting one occurrence of $\perp$ by any domain value will result in an instance that violates $oic \;{}_{w}\!\!\rightarrow c$. However, neither position of `Kingtoys` is redundant: After substituting one position by `Amazon` the given c-FD $oic \;{}_{w}\!\!\rightarrow c$ is still satisfied.

While it would be nice to also avoid redundant $\perp$-positions, it may not be appropriate to speak of redundant data occurrences in this case. In fact, $\perp$ is a marker not to be confused with a domain value. Furthermore, the amount of information duplicated by redundant $\perp$-positions is intuitively less than for any actual domain value. This insight provides a strong motivation to revise our original semantic normal form proposal of RFNF to the following.

Definition 10 (VRNF)
We say that a position in an instance I is value redundant *if it is not $\perp$ and redundant. We call I* free from value redundancy *if it contains no positions that are value redundant. We say that the schema (T, T_S, Σ) is in* Value redundancy-free normal form *(VRNF) if and only if all instances over the schema are free from value redundancy.*

We want to efficiently decide whether a given schema is in VRNF. Based on our findings on lossless decompositions we will focus on c-keys and c-FDs, and propose a syntactic normal form that we show to be equivalent to VRNF. For this purpose, we require one more definition that helps us distinguish between trivial FDs in the relational model (where an FD of the form $X \rightarrow Y$ with $Y \subseteq X$ is satisfied by all relations) from trivial c-FDs.

Definition 11 (internal/external FD)
We call a functional dependency $X \rightarrow Y$ internal *if $Y \subseteq X$. Otherwise we call it* external.

Indeed, an internal c-FD $X \;{}_{w}\!\!\rightarrow Y$ is only trivial if $Y \subseteq T_S$. For example, on $(T = oicp, T_S = oip)$ the internal c-FD $oic \;{}_{w}\!\!\rightarrow c$ is not trivial since $c \notin T_S$. We now propose *SQL-BCNF* as another syntactic normal form, requiring the LHS of each inferable external c-FD to be an inferable c-key.

Definition 12 (SQL-BCNF)
Let (T, T_S, Σ) be a schema, with Σ consisting of c-keys and c-FDs. We say that (T, T_S, Σ) is in SQL-BCNF *if and only if for every* external *c-FD $X \ {}_w\!\!\rightarrow Y \in \Sigma^+$, $c\,\langle X\rangle \in \Sigma^+$.*

As for our BCNF, the SQL-BCNF condition is invariant under equivalent constraint representations, and using our linear-time decidability of the implication problem for c-keys and c-FDs, we can show that it is decidable in quadratic time in the input if a given schema is in SQL-BCNF.

Theorem 14 *A schema (T, T_S, Σ) is in SQL-BCNF if and only if for every* external *c-FD $X \ {}_w\!\!\rightarrow Y \in \Sigma$, $c\,\langle X\rangle \in \Sigma^+$. It can therefore be decided in time quadratic in the input whether a given schema is in SQL-BCNF.*

The schema $(T = oicp, T_S = oip, \Sigma = \{oic \ {}_w\!\!\rightarrow cp\})$ of Example 3 is not in SQL-BCNF because the c-FD $oic \ {}_w\!\!\rightarrow cp$ is external but the c-key $c\,\langle oic\rangle$ is not implied by Σ. However, both schemata $(T_1 = oic, T_{1,S} = oi, \Sigma_1 = \{oic \ {}_w\!\!\rightarrow c\})$ and $(T_2 = oicp, T_{2,S} = oip, \Sigma_2 = \{c\,\langle oic\rangle\})$ are in SQL-BCNF.

Our semantic justification for proposing SQL-BCNF is its ability to permit exactly those instances that are free from redundant data value occurrences.

Theorem 15 *A schema (T, T_S, Σ) is in VRNF if and only if it is in SQL-BCNF.*

For examples, the given schema $(T = oicp, T_S = oip, \Sigma = \{oic \ {}_w\!\!\rightarrow cp\})$ is not in VRNF, but both schemata $(T_1 = oic, T_{1,S} = oi, \Sigma_1 = \{oic \ {}_w\!\!\rightarrow c\})$ and $(T_2 = oicp, T_{2,S} = oip, \Sigma_2 = \{c\,\langle oic\rangle\})$ are in VRNF.

6.3 Lossless VRNF decompositions

We are now ready to compute lossless VRNF decompositions, based on any given set of total FDs and certain keys. Algorithm 3 generalizes the classical BCNF decomposition algorithm from relational databases, which is given by the special case where $T_S = T$ and Σ implies some p- or c-key. Already the classical algorithm runs in exponential time in the input, due to computing the set of constraints that hold on a sub-schema. Algorithms that produce lossless BCNF decompositions in polynomial time [38] decompose projected schemata that are already in BCNF, thereby making the de-normalization effort more difficult.

Algorithm 3 always returns a lossless VRNF decomposition of the given input in time exponential in the input.

Theorem 16 *Algorithm 3 is correct and terminates in exponential time in the input.*

The correctness of Algorithm 3 is ensured as the totality of (LHS-minimal) FDs on sub-schemata is preserved during decomposition. That is, for every LHS-minimal FD $X \ {}_w\!\!\rightarrow Y$ implied by a set Σ of total FDs and certain keys, the total FD $X \ {}_w\!\!\rightarrow XY$ is also implied by Σ.

```
Algorithm 3 VRNF decomposition
Input: Schema $(T, T_S, \Sigma)$ where $\Sigma$ consists of certain keys and total FDs.
Output: Lossless VRNF decomposition $\mathcal{D}$.
Initialize $\mathcal{D} := \{\ [[T]]\ \}$
while $\mathcal{D}$ is not in VRNF do
    pick $[[T_i]] \in \mathcal{D}$ or $[T_i] \in \mathcal{D}$ not in VRNF, remove from $\mathcal{D}$
    pick external $X\ {}_{w}\!\!\rightarrow XY$ on $T_i$ implied by $\Sigma$ where $\Sigma \not\models c\,\langle X\rangle$
    if $[[T_i]]$ picked then
        add $[[X(T_i - XY)]]$ and $[XY]$ to $\mathcal{D}$
    else
        add $[X(T_i - XY)]$ and $[XY]$ to $\mathcal{D}$
return $\mathcal{D}$
```

Continuing Example 3, on input $(T = oicp, T_S = oip, \Sigma = \{oic\ {}_{w}\!\!\rightarrow cp\})$, Algorithm 3 would return the lossless VRNF decomposition: $(T_1 = oic, T_{1,S} = oi, \Sigma_1 = \{oic\ {}_{w}\!\!\rightarrow c\})$ and $(T_2 = oicp, T_{2,S} = oip, \Sigma_2 = \{c\,\langle oic\rangle\})$.

Deciding if the projection of a given schema onto a given attribute set is in VRNF (SQL-BCNF) is co-NP complete.

Theorem 17 *The problem of deciding whether the projection of a given schema onto a given attribute set is in SQL-BCNF (VRNF) is co-NP complete.*

7 Experiments

We conducted experiments to get some indication of how frequent certain FDs are, and how often they can be used for schema decomposition. We also use a public data set to illustrate our techniques and their benefits, and comment on the overheads of discovering certain FDs from data sets. Experiments were run in Ubuntu 14.04 LTS on an AMD FX(tm)-8350 Eight-Core Processor with 15.6 GB memory.

Quantitative Insights. We mined a total of 130 tables from the following publicly available data sets:

- GO-termdb (Gene Ontology)
  `www.geneontology.org/`
- IPI (International Protein Index)
  `www.ebi.ac.uk/IPI`
- LMRP (Local Medical Review Policy)
  `www.cms.gov/medicare-coverage-database/`
- PFAM (protein families)
  `pfam.sanger.ac.uk/`
- RFAM (RNA families)
  `rfam.sanger.ac.uk/`
- Naumann (benchmarks for FD mining)
  `https://hpi.de/naumann/projects/repeatability/data-profiling/fd.html`

- UCI (Machine Learning Repository)
  `https://archive.ics.uci.edu/ml/datasets.html`

We classify FDs into possible FDs (p-FDs), certain FDs (c-FDs) and not-null FDs (nn-FDs) containing no NULL columns in the LHS of the FD. Furthermore we record the number of certain FDs which are total (t-FDs), and the number of total FDs which (1) contain RHS attributes that do not occur on the LHS, and (2) are not certain keys. These criteria ensure that the FD can be utilized for VRNF decomposition, and we denote them as λ-*FDs*. Our results are given below. We record all non-trivial FDs with minimal LHSs, and only once per LHS (multiple FDs of the same type with identical LHS are combined and reported once).

nn-FDs	p-FDs	c-FDs	t-FDs	λ-FDs
847	557	419	205	83

Two factors are likely to have a significant impact on the figures above. First, constraints may only hold accidentally, especially when the tables examined are small. Second, constraints that should sensibly hold may well be violated due to lack of enforcement. We thus consider our results qualitative rather than quantitative. Still, they indicate that c-FDs do appear frequently in practice, and are helpful in decomposing schemas further to eliminate redundancy.

We report the relative size of tables projected onto λ-FDs, as this suggests the amount of data redundancy avoided by decomposing into VRNF. By eliminating data redundancy we minimize the overhead to maintain data consistency, which is the primary goal of normalization. Our results for the 83 λ-FDs (and the 620 nn-FDs whose LHSs are not keys) are visualized in Figure 6.

We observe a large gap in the distribution for λ-FDs, with no relative projection sizes between 52% and 78%. Closer manual examination of the FDs and tables involved reveals that for the majority of λ-FDs with projection size of 78% and over, the LHSs should really be certain keys, but are not due to dirty data (e.g. identical contact details being stored multiple times). This leaves 27 λ-FDs which could sensibly be used for decomposition to eliminate a significant number of redundant data values, plus an unknown number that *should* hold but are violated by dirty data. Similar observations hold for nn-FDs with no clear gap in their distribution but a large number of LHSs that should be keys.

Qualitative Insights. First, we illustrate the occurrence of a certain functional dependency in a real-world data set and its use for decomposing the data set without loss of information. For this purpose, we use the `contact_draft_lookup` table in the **LMRP** database. The actual table contains 14 columns and 124 rows. Figure 7 shows a snippet I consisting of only 5 columns and 14 rows.

The table `contact_draft_lookup` satisfies the λ-FD

$$\sigma: \ \textit{first\_name,last\_name,city} \ {}_w\!\rightarrow \\ \textit{first\_name,last\_name,city,state\_id} \ .$$

In fact, the `contact_draft_lookup` even satisfies the λ-FDs

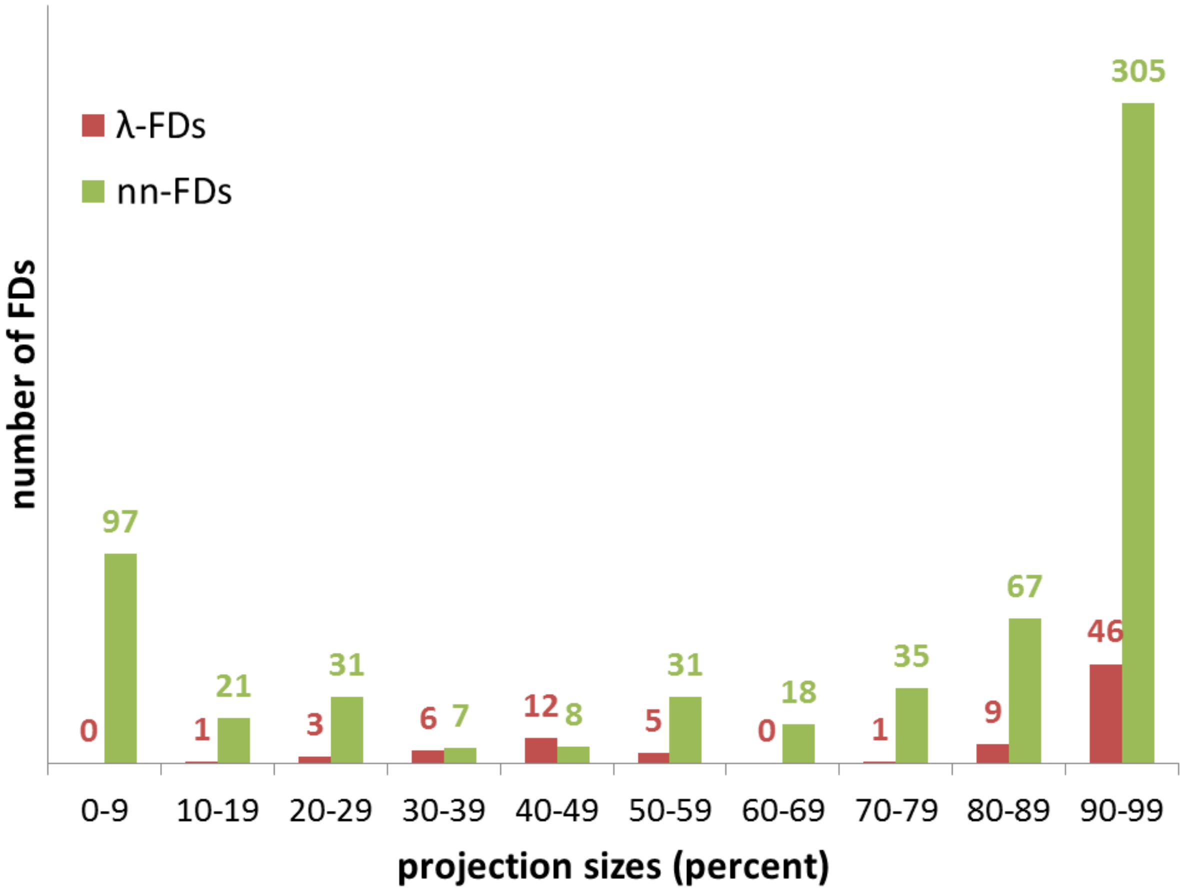


Figure 6: Relative sizes of projections on λ-FDs

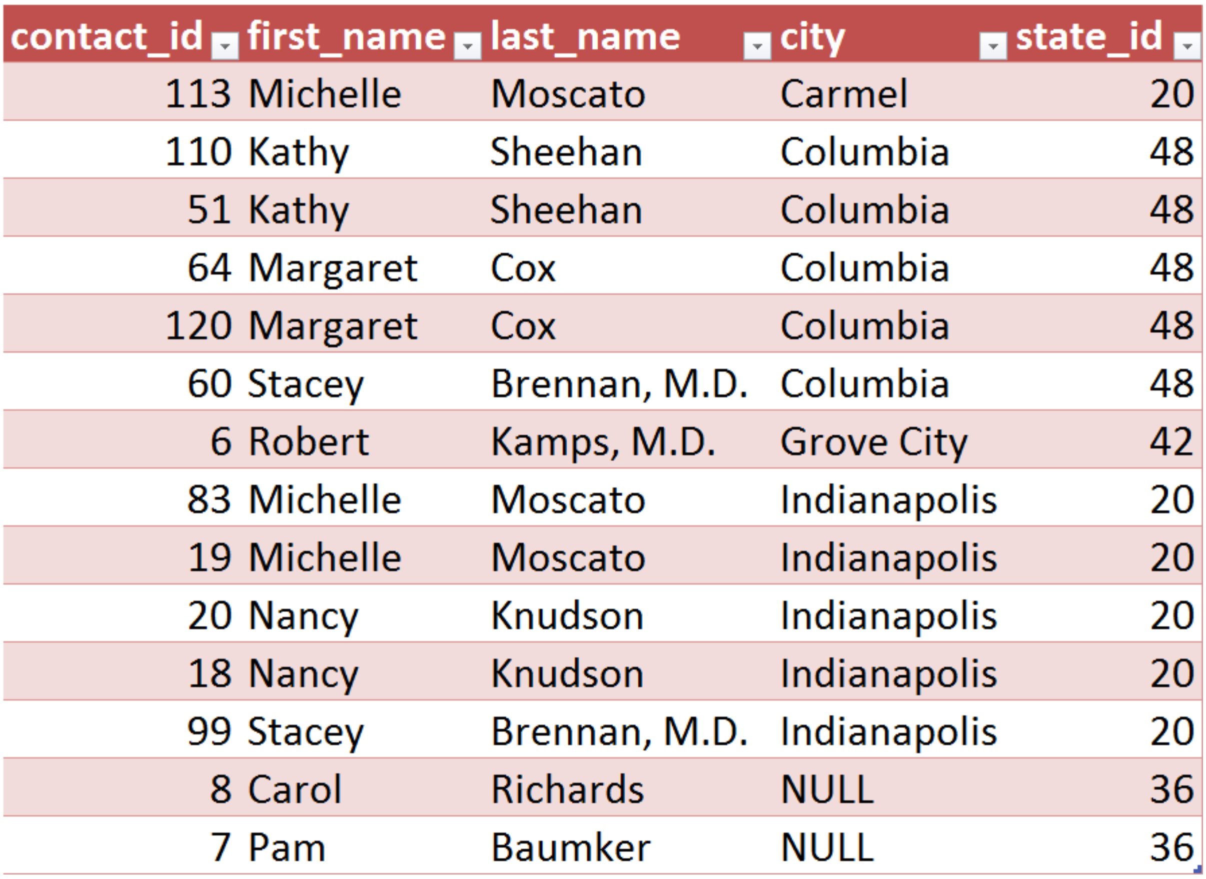
$$\textit{first\_name,city} \ {}_w\!\!\to \textit{first\_name,last\_name, city, state\_id}, \text{ and}$$
$$\textit{last\_name,city} \ {}_w\!\!\to \textit{first\_name,last\_name, city, state\_id},$$

but this is probably accidental. It can further be observed that a person alone does not functionally determine the state, because people can move. For example, Stacey Brennan, M.D. must have moved from Columbia to Indianapolis. In particular, the FD *first_name, last_name* → *state_id* does not hold on `contact_draft_lookup`. The possible and certain variants of this FD coincide because the columns *first_name*, *last_name*, *state_id* do not contain null marker occurrences in the whole table. A VRNF-decomposition of `contact_draft_lookup` using σ eliminates data redundancy: the set-projection on [*first_name, last_name, city, state_id*] contains 105 rows, i.e., 19 sources of potential inconsistency are eliminated. The c-key

$$c\,\langle first\_name, last\_name, city\rangle$$

holds on this projection. For snippet I we obtain the decomposition shown in Figure 8. Finally, the c-FD *city* → *state_id* already fails to hold on our snippet I, so lossless decompositions according to this FD are not possible.

As a second example we apply Algorithm 3 to the `contractor` table of the **LMRP** database. This table has 22 columns and 173 rows. We decompose `contractor` into four

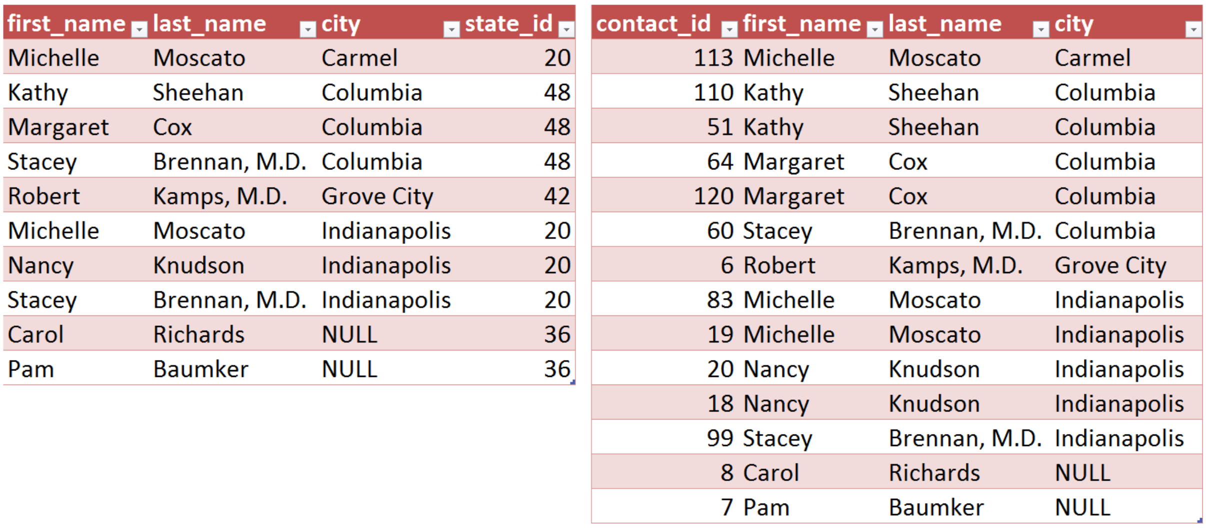


contact_id	first_name	last_name	city	state_id
113	Michelle	Moscato	Carmel	20
110	Kathy	Sheehan	Columbia	48
51	Kathy	Sheehan	Columbia	48
64	Margaret	Cox	Columbia	48
120	Margaret	Cox	Columbia	48
60	Stacey	Brennan, M.D.	Columbia	48
6	Robert	Kamps, M.D.	Grove City	42
83	Michelle	Moscato	Indianapolis	20
19	Michelle	Moscato	Indianapolis	20
20	Nancy	Knudson	Indianapolis	20
18	Nancy	Knudson	Indianapolis	20
99	Stacey	Brennan, M.D.	Indianapolis	20
8	Carol	Richards	NULL	36
7	Pam	Baumker	NULL	36

Figure 7: Snippet I of `contact_draft_lookup`

tables with the following λ-FDs (without repeating LHS attributes on the RHSs)
1. *city,url* ${}_w\!\rightarrow$ *dmerc_rgn,status*
2. *cmd_name,phone,url* ${}_w\!\rightarrow$ *contractor_version,status_flag*
3. *address1,contractor_bus_name,contractor_type_id* ${}_w\!\rightarrow$ *url*.

Table 1 has 4 attributes and 38 rows, table 2 has 5 attributes and 67 rows, table 3 has 4 attributes and 73 rows, and the remaining table has 17 attributes and 173 rows (being the multi-set projection). In the process we eliminated 448 redundant data values, namely: 1 for *dmerc_rng*, 135 for *status*, 106 for *contractor_version*, 106 for *status_flag*, and 100 for *url*. The example shows that VRNF-decomposition can also eliminate null marker redundancies (but this is not guaranteed), in fact, we eliminated 134 of those in the *dmerc_rng* column. While storage saving is not the primary goal of normalization, we can compare the total number of cells over all tables (the sum of rows × columns). The original data set contains $173 \times 22 = 3806$ cells, while the decomposed tables contain $173 \times 17 + 38 \times 4 + 67 \times 5 + 73 \times 4 = 3720$ cells. Finally, we include a very simple comparison of the query and update performance between the normalized and non-normalized data set. Because of the small number of rows, we took the cross product of `contractor` with the new column $(1, \ldots, 1000)$, called *new*, in order to obtain a data set of a more typical size. The new column was then part of each FD and key, and each table in the decomposition. The time for validating the c-FD *new,city,url* ${}_w\!\rightarrow$ *dmerc_rgn,status* on the



first_name	last_name	city	state_id
Michelle	Moscato	Carmel	20
Kathy	Sheehan	Columbia	48
Margaret	Cox	Columbia	48
Stacey	Brennan, M.D.	Columbia	48
Robert	Kamps, M.D.	Grove City	42
Michelle	Moscato	Indianapolis	20
Nancy	Knudson	Indianapolis	20
Stacey	Brennan, M.D.	Indianapolis	20
Carol	Richards	NULL	36
Pam	Baumker	NULL	36

contact_id	first_name	last_name	city
113	Michelle	Moscato	Carmel
110	Kathy	Sheehan	Columbia
51	Kathy	Sheehan	Columbia
64	Margaret	Cox	Columbia
120	Margaret	Cox	Columbia
60	Stacey	Brennan, M.D.	Columbia
6	Robert	Kamps, M.D.	Grove City
83	Michelle	Moscato	Indianapolis
19	Michelle	Moscato	Indianapolis
20	Nancy	Knudson	Indianapolis
18	Nancy	Knudson	Indianapolis
99	Stacey	Brennan, M.D.	Indianapolis
8	Carol	Richards	NULL
7	Pam	Baumker	NULL

Figure 8: VRNF-decomposition of I from Figure 7

non-normalized data set took 122ms, while the time for validating the c-key *new,city,url* on the normalized table took 15ms. This shows a clear performance gain in validating consistency, the main purpose of normalization. In addition, the loss in efficiency when processing queries is rather small: Selecting all tuples from the non-normalized table took 2,957ms, while the selection of all tuples from the join of all normalized tables took 3,150ms.

Discovering c-FDs. The problem of computing a cover for the set of c-FDs from a given data set is not in the scope of this paper. Nevertheless, we provide some brief insight related to the performance of our discovery algorithm for c-FDs. Reference [33] contains a detailed performance analysis of the seven most popular algorithms that discover classical FDs (treating nulls as any other domain value). The following table shows for three of the Naumann data sets the number of classical FDs discovered, the time of the best performing algorithm to compute them according to [33], as well as the number of c-FDs we discovered and the time required by our algorithm.

data set	columns	rows	FDs	time	c-FDs	time
	[#]	[#]	[#]	[s]	[#]	[s]
breast-cancer	11	699	46	0.5	54	0.1
adult	14	48,842	78	5.9	78	10.4
hepatitis	20	155	8,250	0.8	264	1.2

In general, c-FDs and classical FDs are incomparable which means that data sets may exhibit different numbers of them, and algorithms to discover them may perform quite differently. Nevertheless, we can say that the performance of our algorithm is rather competitive in comparison to that of the best breed discovery algorithms for classical FDs.

8 Conclusion and Future Work

The impact of relational normalization theory on database practice has been limited. Well-known results that enable relational normalization do not apply to SQL-compliant data in which duplicate and partial information readily occur. We have developed the first SQL schema design framework based on the new notion of a certain functional dependency. Coupled with possible keys, certain keys, possible FDs, and `NOT NULL` constraints, certain FDs handle duplicate information and null marker occurrences in SQL data appropriately. These dependencies can express many desirable constraints in practice, form a rich source of SQL data redundancy, yet can still be reasoned about in linear time. We have proposed a normal form for well-designed SQL schemata, and semantically justified the proposal by showing that it permits exactly those instances that are free from data redundancy. In contrast to relational normalization, SQL schema normalization is not always achievable. Nevertheless, we identified total FDs for which we can transform any given SQL schema into one that permits exactly those instances that are free from redundant data values but not redundant null marker occurrences. Our experiments confirm that certain and total FDs occur frequently in real data sets, and our framework is effective in decomposing these data sets such that all data redundancy is eliminated and no information lost.

Future work will address the scalable discovery of certain FDs from large data sets, extending [1, 33] in which null markers are treated as any other domain value. Regarding SQL normalization, it is interesting to define notions of i) dependency-preservation and extensions of the Third normal form [8], ii) update anomalies and investigate what our normal forms achieve in terms of these [40], and iii) other dependencies and normal forms [12, 14, 27]. There are strong relationships with conditional independencies, paramount in statistics and machine learning [29, 32].

Acknowledgement. The research is supported by the Marsden Fund Council grant 3701176 from Government funding, administered by the Royal Society of New Zealand.

References

[1] Z. Abedjan, L. Golab, and F. Naumann. Profiling relational data: a survey. *VLDB J.*, 24(4):557–581, 2015.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[3] M. Arenas. Normalization theory for XML. *SIGMOD Record*, 35(4):57–64, 2006.

[4] W. W. Armstrong. Dependency structures of data base relationships. In *IFIP Congress*, pages 580–583, 1974.

[5] P. Atzeni and N. M. Morfuni. Functional dependencies and constraints on null values in database relations. *Information and Control*, 70(1):1–31, 1986.

[6] C. Beeri and P. A. Bernstein. Computational problems related to the design of normal form relational schemas. *ACM Trans. Database Syst.*, 4(1):30–59, 1979.

[7] P. A. Bernstein and N. Goodman. What does boyce-codd normal form do? In *VLDB*, pages 245–259, 1980.

[8] J. Biskup, U. Dayal, and P. A. Bernstein. Synthesizing independent database schemas. In *SIGMOD*, pages 143–151, 1979.

[9] J. Biskup and T. Polle. Adding inclusion dependencies to an object-oriented data model with uniqueness constraints. *Acta Inf.*, 39(6-7):391–449, 2003.

[10] E. F. Codd. Recent investigations in relational data base systems. In *IFIP Congress*, pages 1017–1021, 1974.

[11] E. F. Codd. *The Relational Model for Database Management, Version 2*. Addison-Wesley, 1990.

[12] H. Darwen, C. Date, and R. Fagin. A normal form for preventing redundant tuples in relational databases. In *ICDT*, pages 114–126, 2012.

[13] J. Diederich and J. Milton. New methods and fast algorithms for database normalization. *ACM Trans. Database Syst.*, 13(3):339–365, 1988.

[14] R. Fagin. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst.*, 2(3):262–278, 1977.

[15] R. Fagin. A normal form for relational databases that is based on domains and keys. *ACM Trans. Database Syst.*, 6(3):387–415, 1981.

[16] F. Ferrarotti, S. Hartmann, H. Köhler, S. Link, and M. W. Vincent. The Boyce-Codd-Heath normal form for SQL. In *WoLLIC*, pages 110–122, 2011.

[17] S. Hartmann and S. Link. When data dependencies over SQL tables meet the logics of paradox and S-3. In *PODS*, pages 317–326, 2010.

[18] S. Hartmann and S. Link. The implication problem of data dependencies over SQL table definitions. *ACM Trans. Database Syst.*, 37(2):13, 2012.

[19] I. J. Heath. Unacceptable file operations in a relational data base. In *SIGFIDET Workshop*, pages 19–33, 1971.

[20] R. T. Herrera, J. Tekli, R. Chbeir, S. Laborie, I. Dongo, and R. Guzman. Toward RDF normalization. In *ER*, pages 261–275, 2015.

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

[22] G. Lausen. Relational databases in RDF: Keys and foreign keys. In *SWDB-ODBIS*, pages 43–56, 2007.

[23] M. Levene. *The Nested Universal Relation Database Model*, volume 595 of *LNCS*. Springer, 1992.

[24] M. Levene and G. Loizou. Axiomatisation of functional dependencies in incomplete relations. *Theor. Comput. Sci.*, 206(1-2):283–300, 1998.

[25] M. Levene and G. Loizou. Database design for incomplete relations. *ACM Trans. Database Syst.*, 24(1):80–125, 1999.

[26] M. Levene and G. Loizou. *A guided tour of relational databases and beyond*. Springer, 1999.

[27] M. Levene and M. W. Vincent. Justification for inclusion dependency normal form. *IEEE Trans. Knowl. Data Eng.*, 12(2):281–291, 2000.

[28] Y. E. Lien. On the equivalence of database models. *J. ACM*, 29(2):333–362, 1982.

[29] S. Link. Reasoning about saturated conditional independence under uncertainty. In *AAAI*, 2013.

[30] J. A. Makowsky and E. V. Ravve. Dependency preserving refinements and the fundamental problem of database design. *Data Knowl. Eng.*, 24(3):277–312, 1998.

[31] H. Mannila and K.-J. Räihä. *Design of Relational Databases*. Addison-Wesley, 1992.

[32] M. Niepert, M. Gyssens, B. Sayrafi, and D. V. Gucht. On the conditional independence implication problem: A lattice-theoretic approach. *Artif. Intell.*, 202:29–51, 2013.

[33] T. Papenbrock, J. Ehrlich, J. Marten, T. Neubert, J. Rudolph, M. Schönberg, J. Zwiener, and F. Naumann. Functional dependency discovery. *PVLDB*, 8(10):1082–1093, 2015.

[34] J. Paredaens, P. D. Bra, M. Gyssens, and D. V. Gucht. *The Structure of the Relational Database Model*, volume 17 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1989.

[35] J. Rissanen. Independent components of relations. *ACM Trans. Database Syst.*, 2(4):317–325, 1977.

[36] Z. Tari, J. Stokes, and S. Spaccapietra. Object normal forms and dependency constraints for object-oriented schemata. *ACM Trans. Database Syst.*, 22(4):513–569, 1997.

[37] B. Thalheim. *Entity-relationship modeling - foundations of database technology*. Springer, 2000.

[38] D.-M. Tsou and P. C. Fischer. Decomposition of a relation scheme into Boyce-Codd normal form. *SIGACT News*, 14(3):23–29, 1982.

[39] Y. Vassiliou. Functional dependencies and incomplete information. In *VLDB*, pages 260–269, 1980.

[40] M. W. Vincent. Semantic foundations of 4NF in relational database design. *Acta Inf.*, 36(3):173–213, 1999.

[41] M. W. Vincent, J. Liu, and C. Liu. Strong functional dependencies and their application to normal forms in XML. *ACM Trans. Database Syst.*, 29(3):445–462, 2004.

A Reasoning Tools: Proofs

Lemma 3
The following statements holds:

i) If $X \subseteq Y$ then $X^{*p} \subseteq Y^{*p}$ and $X^{*c} \subseteq Y^{*c}$.

ii) $X, X^{*c} \subseteq X^{*p}$

iii) $(X^{*c})^{*c} \subseteq X^{*c}$ and $(X^{*p})^{*c} \subseteq X^{*p}$

Proof i) Follows from L-Augmentation.

ii) $X \subseteq X^{*p}$ follows from Reflexivity. $X^{*c} \subseteq X^{*p}$ follows from Reflexivity and Pseudo-Transitivity:

$$\frac{\dfrac{}{X \;{}_{s}\!\!\rightarrow X}\,(R) \qquad X \;{}_{w}\!\!\rightarrow Z}{X \;{}_{s}\!\!\rightarrow Z}\,(T)$$

iii) $(X^{*c})^{*c} \subseteq X^{*c}$ follows by Union, L-Augmentation and Pseudo-Transitivity:

$$\frac{\dfrac{\vdots}{X \;{}_{w}\!\!\rightarrow X^{*c}}\,(U) \qquad \dfrac{X^{*c} \;{}_{w}\!\!\rightarrow Z}{XX^{*c} \;{}_{w}\!\!\rightarrow Z}\,(A)}{X \;{}_{w}\!\!\rightarrow Z}\,(T)$$

$(X^{*p})^{*c} \subseteq X^{*p}$ follows similarly:

$$\frac{\dfrac{\vdots}{X \;{}_{s}\!\!\rightarrow X^{*p}}\,(U) \qquad \dfrac{X^{*p} \;{}_{w}\!\!\rightarrow Z}{XX^{*p} \;{}_{w}\!\!\rightarrow Z}\,(A)}{X \;{}_{s}\!\!\rightarrow Z}\,(T)$$

This concludes the proof. ∎

Theorem 18 (Theorem 2 restated)
Let (T, T_S, Σ) be a schema with Σ containing p-FDs and c-FDs. Then Σ implies $X \;{}_{s}\!\!\rightarrow Y$ if and only if $Y \subseteq X^{*p}$, and Σ implies $X \;{}_{w}\!\!\rightarrow Y$ if and only if $Y \subseteq X^{*c}$.

Proof This are direct consequences of the soundness of the union and decomposition rules. ∎

Theorem 19 (Theorem 3 restated)
The p- and c-Closure algorithms are correct, and the implication problem for the combined class of p-FDs, c-FDs, and `NOT NULL` constraints can be decided in linear time.

Proof The correctness of the Algorithms are shown in Theorems 20 and 21. The linear-time complexity follows from standard optimization techniques [6], as explained in the main section. ∎

Theorem 20 *Algorithm p-Closure is correct.*

Proof We will show that the invariant $\mathcal{C} \subseteq X^{*p}$ holds, and that $\mathcal{C} \supseteq X^{*p}$ holds upon termination. Note that $\mathcal{C} \subseteq X^{*p}$ implies $\Sigma \models X \;{}_{s}{\rightarrow}\; \mathcal{C}$ by the Union rule.

($\mathcal{C} \subseteq X^{*p}$) Initially we have $\mathcal{C} = X$. By Reflexivity we get $\Sigma \models X \;{}_{s}{\rightarrow}\; X$ so the invariant holds.

Consider now $Y \;{}_{w}{\rightarrow}\; Z \in \Sigma$ with $Y \subseteq \mathcal{C}$. We can derive

$$\dfrac{\dfrac{X \;{}_{s}{\rightarrow}\; \mathcal{C}}{X \;{}_{s}{\rightarrow}\; Y}\,(D) \qquad \dfrac{Y \;{}_{w}{\rightarrow}\; Z}{XY \;{}_{w}{\rightarrow}\; Z}\,(A)}{X \;{}_{s}{\rightarrow}\; Z}\,(T)$$

so the invariant is preserved.

Finally let $Y \;{}_{s}{\rightarrow}\; Z \in \Sigma$ with $Y \subseteq (\mathcal{C} \cap T_S) \cup X$. We can derive

$$\dfrac{\dfrac{X \;{}_{s}{\rightarrow}\; \mathcal{C}}{X \;{}_{s}{\rightarrow}\; (Y - X)}\,(D) \qquad \dfrac{Y \;{}_{s}{\rightarrow}\; Z}{X(Y - X) \;{}_{s}{\rightarrow}\; Z}\,(A)}{X \;{}_{s}{\rightarrow}\; Z}\,(NT)$$

so again the invariant is preserved.

($\mathcal{C} \supseteq X^{*p}$) Consider a table $I = \{t, t'\}$ on (T, T_S) with

$$\left.\begin{array}{l} t[A] \sim_s t'[A] \\ t[A] = \bot = t'[A] \\ t[A] \not\sim_w t'[A] \end{array}\right\} \text{for} \left\{\begin{array}{l} A \in \mathcal{C} \cap (T_S \cup X) \\ A \in \mathcal{C} - (T_S \cup X) \\ A \notin \mathcal{C} \end{array}\right.$$

In particular we have $t[X] \sim_s t'[X]$ and $t[A] \neq t'[A]$ for $A \notin \mathcal{C}$. We will show that Σ holds on I.

Let $Y \;{}_{w}{\rightarrow}\; Z \in \Sigma$. If $Y \not\subseteq \mathcal{C}$ then $t[Y] \not\sim_w t'[Y]$ so $Y \;{}_{w}{\rightarrow}\; Z$ holds on I. Otherwise Z has been added to $\mathcal{C}$ in the first for-loop. Thus $t[Z] = t'[Z]$ meaning $Y \;{}_{w}{\rightarrow}\; Z$ holds on I.

Now let $Y \;{}_{s}{\rightarrow}\; Z \in \Sigma$. If $Y \not\subseteq \mathcal{C} \cap (T_S \cup X)$ then $t[Y] \not\sim_s t'[Y]$ so $Y \;{}_{s}{\rightarrow}\; Z$ holds on I. Otherwise $Y \subseteq \mathcal{C} \cap (T_S \cup X) = (\mathcal{C} \cap T_S) \cup X$, so Z has been added to $\mathcal{C}$ in the second for-loop. Thus $t[Z] = t'[Z]$ meaning $Y \;{}_{w}{\rightarrow}\; Z$ holds on I.

Thus I respects Σ but violates $X \;{}_{s}{\rightarrow}\; A$ for $A \notin \mathcal{C}$, showing $\mathcal{C} \supseteq X^{*p}$.

This concludes the proof. ∎

Theorem 21 *Algorithm c-Closure is correct.*

Proof We will show that the invariant $\mathcal{C} \subseteq X^{*c}$ holds, and that $\mathcal{C} \supseteq X^{*c}$ holds upon termination.

($\mathcal{C} \subseteq X^{*c}$) Initially we have $\mathcal{C} = X \cap T_S$. We can derive

$$\dfrac{\dfrac{\dfrac{}{X \cap T_S \;{}_{s}{\rightarrow}\; X \cap T_S}\,(R)}{X \cap T_S \;{}_{w}{\rightarrow}\; X \cap T_S}\,(S)}{X \;{}_{w}{\rightarrow}\; X \cap T_S}\,(A)$$

so the invariant holds.

Consider now $Y \;{}_{w}{\rightarrow}\; Z \in \Sigma$ with $Y \subseteq \mathcal{C} \cup X$. We can derive

$$\dfrac{\dfrac{X \;{}_{w}{\rightarrow}\; \mathcal{C}}{X \;{}_{w}{\rightarrow}\; (Y - X)}\,(D) \qquad \dfrac{Y \;{}_{w}{\rightarrow}\; Z}{X(Y - X) \;{}_{w}{\rightarrow}\; Z}\,(A)}{X \;{}_{w}{\rightarrow}\; Z}\,(T)$$

so the invariant is preserved.

Finally let $Y \;{}_{s}{\rightarrow}\; Z \in \Sigma$ with $Y \subseteq \mathcal{C} \cap T_S$. We can derive

$$\dfrac{\dfrac{X \;{}_{w}{\rightarrow}\; \mathcal{C}}{X \;{}_{w}{\rightarrow}\; Y}\,(D) \qquad \dfrac{\dfrac{Y \;{}_{s}{\rightarrow}\; Z}{Y \;{}_{w}{\rightarrow}\; Z}\,(S)}{XY \;{}_{w}{\rightarrow}\; Z}\,(A)}{X \;{}_{w}{\rightarrow}\; Z}\,(T)$$

so again the invariant is preserved.

($\mathcal{C} \supseteq X^{*c}$) Consider a table $I = \{t, t'\}$ on (T, T_S) with

$$\left.\begin{array}{l} t[A] \sim_s t'[A] \\ t[A] = \bot = t'[A] \\ t[A] = \bot \neq t'[A] \\ t[A] \not\sim_w t'[A] \end{array}\right\} \text{for} \left\{\begin{array}{l} A \in \mathcal{C} \cap T_S \\ A \in \mathcal{C} - T_S \\ A \in X - \mathcal{C} \\ A \notin \mathcal{C} \cup X \end{array}\right.$$

In particular we have $t[X] \sim_w t'[X]$ and $t[A] \neq t'[A]$ for $A \notin \mathcal{C}$. We will show that Σ holds on I.

Let $Y \;{}_{w}{\rightarrow}\; Z \in \Sigma$. If $Y \not\subseteq \mathcal{C} \cup X$ then $t[Y] \not\sim_w t'[Y]$ so $Y \;{}_{w}{\rightarrow}\; Z$ holds on I. Otherwise Z has been added to $\mathcal{C}$ in the first for-loop. Thus $t[Z] = t'[Z]$ meaning $Y \;{}_{w}{\rightarrow}\; Z$ holds on I.

Now let $Y \;{}_{s}{\rightarrow}\; Z \in \Sigma$. If $Y \not\subseteq \mathcal{C} \cap T_S$ then $t[Y] \not\sim_s t'[Y]$ so $Y \;{}_{s}{\rightarrow}\; Z$ holds on I. Otherwise Z has been added to $\mathcal{C}$ in the second for-loop. Thus $t[Z] = t'[Z]$ meaning $Y \;{}_{w}{\rightarrow}\; Z$ holds on I.

Thus I respects Σ but violates $X \;{}_{w}{\rightarrow}\; A$ for $A \notin \mathcal{C}$, showing $\mathcal{C} \supseteq X^{*c}$.

This concludes the proof. ∎

Theorem 22 (Theorem 1 restated)
The set $\mathfrak{F}$ of inference rules from Table 1 are sound and complete for the implication of certain and possible functional dependencies and `NOT NULL` constraints.

Proof The soundness proofs are straightforward.

For completeness, let $\Sigma \models X \ {}_{s}\!\!\rightarrow Y$ or $\Sigma \models X \ {}_{w}\!\!\rightarrow Y$. Then $Y \subseteq X^{*p}$ or $Y \subseteq X^{*c}$. However, as we have seen in the correctness proofs for Algorithms p-Closure and c-Closure, we can derive $X \ {}_{s}\!\!\rightarrow X^{*p}$ and $X \ {}_{w}\!\!\rightarrow X^{*c}$ from Σ using the axioms of Theorem 1. The FDs $X \ {}_{s}\!\!\rightarrow Y$ or $X \ {}_{w}\!\!\rightarrow Y$ follow with the Decomposition rule. $\blacksquare$

Theorem 23
Let $X \rightarrow Y$ be a possible or certain functional dependency over (T, T_S) and Σ a set of possible and certain keys and functional dependencies over (T, T_S). Then Σ implies $X \rightarrow Y$ iff $\Sigma|_{FD}$ implies $X \rightarrow Y$.

Proof Since Σ implies $\Sigma|_{\mathrm{FD}}$, any constraint implied by $\Sigma|_{\mathrm{FD}}$ is implied by Σ.

Conversely, let $X \rightarrow Y$ be a possible or certain FD implied by Σ but not by $\Sigma|_{\mathrm{FD}}$. Then there must exist a table I over (T, T_S) such that I respects $\Sigma|_{\mathrm{FD}}$ but violates $X \rightarrow Y$, and in particular there exist two tuples $t, t' \in I$ such that $\{t, t'\}$ violates $X \rightarrow Y$ but respects $\Sigma|_{\mathrm{FD}}$, so we may assume $I = \{t, t'\}$. As Σ implies $X \rightarrow Y$, I must violate Σ and hence some key $_{(p/c)}\langle Z\rangle \in \Sigma$. This is only possible if $t[Z] \sim t'[Z]$, and since I respects $Z \rightarrow T \in \Sigma|_{\mathrm{FD}}$, we must have $t = t'$. However, this means that I does not violate $X \rightarrow Y$. Contradiction. $\blacksquare$

Theorem 24
Let Σ be a set of possible and certain keys and functional dependencies over (T, T_S). Then

(i) Σ implies $p\langle X\rangle$ iff $\Sigma|_{key}$ implies $c\left\langle X^{*p}_{\Sigma|_{FD}}\right\rangle$ or $p\left\langle X(X^{*p}_{\Sigma|_{FD}} \cap T_S)\right\rangle$, and

(ii) Σ implies $c\langle X\rangle$ iff $\Sigma|_{key}$ implies $c\left\langle XX^{*c}_{\Sigma|_{FD}}\right\rangle$.

Proof In both cases, the 'if' direction follows directly from the correctness of the key-Transitivity and key-NULL-Transitivity rules. For the 'only if' direction, assume that the RHS conditions are false.

(i) Consider a table $I = \{t, t'\}$ over (T, T_S) with

$$\left.\begin{array}{l} t[A] \sim_s t'[A] \\ t[A] = \bot = t'[A] \\ t[A] \not\sim_w t'[A] \end{array}\right\} \text{for} \begin{cases} A \in X(X^{*p}_{\Sigma|_{\mathrm{FD}}} \cap T_S) \\ A \in X^{*p}_{\Sigma|_{\mathrm{FD}}} - X(X^{*p}_{\Sigma|_{\mathrm{FD}}} \cap T_S) \\ A \notin X^{*p}_{\Sigma|_{\mathrm{FD}}} \end{cases}$$

Let $c\langle Y\rangle \in \Sigma|_{\mathrm{key}}$. Since $\Sigma|_{\mathrm{key}} \not\models c\left\langle X^{*p}_{\Sigma|_{\mathrm{FD}}}\right\rangle$, we cannot have $Y \subseteq X^{*p}_{\Sigma|_{\mathrm{FD}}}$. Thus $t[Y] \not\sim_w t'[Y]$, so $c\langle Y\rangle$ holds on I. Let $p\langle Y\rangle \in \Sigma|_{\mathrm{key}}$. Since $\Sigma|_{\mathrm{key}} \not\models p\left\langle X(X^{*p}_{\Sigma|_{\mathrm{FD}}} \cap T_S)\right\rangle$, we cannot have $Y \subseteq X(X^{*p}_{\Sigma|_{\mathrm{FD}}} \cap T_S)$. Thus $t[Y] \not\sim_s t'[Y]$, so $p\,\langle Y\rangle$ holds on I. Together this gives us that I respects $\Sigma|_{\mathrm{key}}$.

Now let $Y \;{}_{w}\!\!\rightarrow Z \in \Sigma|_{\mathrm{FD}}$. If $t[Y] \not\sim_w t'[Y]$ then $Y \;{}_{w}\!\!\rightarrow Z$ holds on I. If $t[Y] \sim_w t'[Y]$ we must have $Y \subseteq X^{*p}_{\Sigma|_{\mathrm{FD}}}$. As $Y \;{}_{w}\!\!\rightarrow Z \in \Sigma|_{\mathrm{FD}}$ this means $Z \subseteq (X^{*p}_{\Sigma|_{\mathrm{FD}}})^{*c}_{\Sigma|_{\mathrm{FD}}} \subseteq (X^{*p}_{\Sigma|_{\mathrm{FD}}})^{*p}_{\Sigma|_{\mathrm{FD}}} = X^{*p}_{\Sigma|_{\mathrm{FD}}}$. Thus $t[Z] = t'[Z]$ and $Y \;{}_{w}\!\!\rightarrow Z$ holds on I. Finally let $Y \;{}_{s}\!\!\rightarrow Z \in \Sigma|_{\mathrm{FD}}$. If $t[Y] \not\sim_s t'[Y]$ then $Y \;{}_{w}\!\!\rightarrow Z$ holds on I. If $t[Y] \sim_s t'[Y]$ we must have $Y \subseteq X(X^{*p}_{\Sigma|_{\mathrm{FD}}} \cap T_S)$. Due to $Y \;{}_{s}\!\!\rightarrow Z \in \Sigma|_{\mathrm{FD}}$ it follows that $Z \subseteq (X(X^{*p}_{\Sigma|_{\mathrm{FD}}} \cap T_S))^{*p}_{\Sigma|_{\mathrm{FD}}} \subseteq (X^{*p}_{\Sigma|_{\mathrm{FD}}})^{*p}_{\Sigma|_{\mathrm{FD}}} = X^{*p}_{\Sigma|_{\mathrm{FD}}}$. Thus $t[Z] = t'[Z]$ and $Y \;{}_{s}\!\!\rightarrow Z$ holds on I. Together this gives us that I respects $\Sigma|_{\mathrm{FD}}$.

As I respects both $\Sigma|_{\mathrm{key}}$ and $\Sigma|_{\mathrm{FD}}$, it respects Σ. But $p\,\langle X\rangle$ is violated by I, so $p\,\langle X\rangle$ is not implied by Σ.

(ii) Consider a table $I = \{t, t'\}$ over (T, T_S) with

$$\left.\begin{array}{l} t[A] \sim_s t'[A] \\ t[A] = \bot = t'[A] \\ t[A] \not\sim_w t'[A] \end{array}\right\} \text{ for } \begin{cases} A \in (XX^{*c}_{\Sigma|_{\mathrm{FD}}}) \cap T_S \\ A \in (XX^{*c}_{\Sigma|_{\mathrm{FD}}}) - T_S \\ A \notin XX^{*c}_{\Sigma|_{\mathrm{FD}}} \end{cases}$$

Let $c\,\langle Y\rangle \in \Sigma|_{\mathrm{key}}$. Since $\Sigma|_{\mathrm{key}} \not\models c\,\langle XX^{*c}_{\Sigma|_{\mathrm{FD}}}\rangle$, we cannot have $Y \subseteq XX^{*c}_{\Sigma|_{\mathrm{FD}}}$. Thus $t[Y] \not\sim_w t'[Y]$, so $c\,\langle Y\rangle$ holds on I. Let $p\,\langle Y\rangle \in \Sigma|_{\mathrm{key}}$. Since $\Sigma|_{\mathrm{key}} \not\models c\,\langle XX^{*c}_{\Sigma|_{\mathrm{FD}}}\rangle$, we cannot have $Y \subseteq (XX^{*c}_{\Sigma|_{\mathrm{FD}}}) \cap T_S$. Thus $t[Y] \not\sim_s t'[Y]$, so $p\,\langle Y\rangle$ holds on I. Together this gives us that I respects $\Sigma|_{\mathrm{key}}$.

Now let $Y \;{}_{w}\!\!\rightarrow Z \in \Sigma|_{\mathrm{FD}}$. If $t[Y] \not\sim_w t'[Y]$ then $Y \;{}_{w}\!\!\rightarrow Z$ holds on I. If $t[Y] \sim_w t'[Y]$ we must have $Y \subseteq XX^{*c}_{\Sigma|_{\mathrm{FD}}}$. As $Y \;{}_{w}\!\!\rightarrow Z \in \Sigma|_{\mathrm{FD}}$ this means $Z \subseteq (XX^{*c}_{\Sigma|_{\mathrm{FD}}})^{*c}_{\Sigma|_{\mathrm{FD}}} \subseteq XX^{*c}_{\Sigma|_{\mathrm{FD}}}$. Thus $t[Z] = t'[Z]$ and $Y \;{}_{w}\!\!\rightarrow Z$ holds on I. Finally let $Y \;{}_{s}\!\!\rightarrow Z \in \Sigma|_{\mathrm{FD}}$. If $t[Y] \not\sim_s t'[Y]$ then $Y \;{}_{w}\!\!\rightarrow Z$ holds on I. If $t[Y] \sim_s t'[Y]$ we must have $Y \subseteq (XX^{*c}_{\Sigma|_{\mathrm{FD}}}) \cap T_S$. Due to $Y \;{}_{s}\!\!\rightarrow Z \in \Sigma|_{\mathrm{FD}}$ it follows that $Z \subseteq ((XX^{*c}_{\Sigma|_{\mathrm{FD}}}) \cap T_S)^{*p}_{\Sigma|_{\mathrm{FD}}} = ((XX^{*c}_{\Sigma|_{\mathrm{FD}}}) \cap T_S)^{*c}_{\Sigma|_{\mathrm{FD}}} \subseteq XX^{*c}_{\Sigma|_{\mathrm{FD}}}$. Thus $t[Z] = t'[Z]$ and $Y \;{}_{s}\!\!\rightarrow Z$ holds on I. Together this gives us that I respects $\Sigma|_{\mathrm{FD}}$.

As I respects both $\Sigma|_{\mathrm{key}}$ and $\Sigma|_{\mathrm{FD}}$, it respects Σ. But $c\,\langle X\rangle$ is violated by I, so $c\,\langle X\rangle$ is not implied by Σ.

This concludes the proof. ▮

Theorem 25 (Theorem 4 restated)
The union of the inference rules from sets $\mathfrak{F}$ in Table 1, $\mathfrak{K}$ from Table 2, and $\mathfrak{FK}$ from Table 3 are sound and complete for the implication of possible and certain keys, functional dependencies, and `NOT NULL` constraints.

Proof (Theorem 1, completeness)
Let $\Sigma \models X \rightarrow Y$. Then by Theorem 23 we have $\Sigma|_{\mathrm{FD}} \models X \rightarrow Y$. Every FD in $\Sigma|_{\mathrm{FD}} - \Sigma$ can be derived from Σ using the key-FD-Weakening rule. And since the rules in $\mathfrak{F}$ are a subset of the rules in $\mathfrak{FK}$, $X \rightarrow Y$ can be derived from $\Sigma|_{\mathrm{FD}}$ by Theorem 1. This shows completeness for FDs.

Let $\Sigma \models_{(p/c)} \langle X \rangle$. By Theorem 24 we have $\Sigma|_{\text{key}} \models c \left\langle X^{*p}_{\Sigma|_{\text{FD}}} \right\rangle$ or $\Sigma|_{\text{key}} \models p \left\langle X(X^{*p}_{\Sigma|_{\text{FD}}} \cap T_S) \right\rangle$ in case (i), and $\Sigma|_{\text{key}} \models c \left\langle XX^{*c}_{\Sigma|_{\text{FD}}} \right\rangle$ in case (ii). Since the rules in $\mathfrak{K}$ are a subset of the rules in $\mathfrak{FK}$, the aforementioned keys can be derived from $\Sigma|_{\text{key}}$ by the completeness of $\mathfrak{K}$ [21]. Due to completeness of our axiomatization for FDs, we can derive $X \ {}_{s}\!\!\rightarrow X^{*p}_{\Sigma|_{\text{FD}}}$, $X \ {}_{s}\!\!\rightarrow X(X^{*p}_{\Sigma|_{\text{FD}}} \cap T_S)$ and $X \ {}_{w}\!\!\rightarrow X^{*c}_{\Sigma|_{\text{FD}}}$ from Σ. Thus $p\langle X \rangle$ or $c\langle X \rangle$ can be derived using key-Transitivity or key-Null-Transitivity rules in each case. This shows completeness for keys. ∎

B Normal Forms: Proofs

Theorem 26 (Theorem 6 restated)
The schema (T, T_S, Σ) is in BCNF if and only if (i) for all non-trivial $X \ {}_{s}\!\!\rightarrow Y \in \Sigma$, $p\langle X \rangle \in \Sigma^+$, and (ii) for all non-trivial $X \ {}_{w}\!\!\rightarrow Y \in \Sigma$, $c\langle X \rangle \in \Sigma^+$.

Proof (only if) This direction follows directly from the definition of BCNF as $\Sigma \subseteq \Sigma^+$.

(if) Here, we need to show that if (i) for all non-trivial $X \ {}_{s}\!\!\rightarrow Y \in \Sigma$, $p\langle X \rangle \in \Sigma^+$, and (ii) for all non-trivial $X \ {}_{w}\!\!\rightarrow Y \in \Sigma$, $c\langle X \rangle \in \Sigma^+$, then it is also true that (i') for all non-trivial $X \ {}_{s}\!\!\rightarrow Y \in \Sigma^+$, $p\langle X \rangle \in \Sigma^+$, and (ii') for all non-trivial $X \ {}_{w}\!\!\rightarrow Y \in \Sigma^+$, $c\langle X \rangle \in \Sigma^+$. We show that not (i') implies not (i), and not (ii') implies not (ii), each by induction over the inference length. For that purpose, we assume w.L.o.G. that $\Sigma = \Sigma_0 \subset \Sigma_1 \subset \cdots \subset \Sigma_k = \Sigma^+$ where Σ_{i+1} results from Σ_i by application of one of the inference rules in $\mathfrak{FK}$ to elements from Σ_i. If $k = 0$, then $\Sigma = \Sigma^+$ and there is nothing to show. Otherwise, we assume that our hypothesis holds for all $j = 0, \ldots, i < k$, and we establish that both claims also hold for $j = i + 1$. The proof proceeds by analyzing all inference rules in $\mathfrak{FK}$ which could have produced the single dependency in $\Sigma_{i+1} - \Sigma_i$. Due to lack of space we illustrate the proof on an interesting case.

Assume that the non-trivial p-FD $X \ {}_{s}\!\!\rightarrow Z \in \Sigma_{i+1} - \Sigma_i$ was inferred from $X \ {}_{s}\!\!\rightarrow Y$ and $XY \ {}_{w}\!\!\rightarrow Z$ by application of (T). As $X \ {}_{s}\!\!\rightarrow Z$ is non-trivial there must be some $A \in Z - X$. If $A \in Y$, then the p-FD $X \ {}_{s}\!\!\rightarrow Y \in \Sigma_i$ is non-trivial. As $p\langle X \rangle \notin \Sigma^+$ the claim follows in this case. If $A \notin Y$, then $A \in Z - XY$ and the c-FD $XY \ {}_{w}\!\!\rightarrow Z \in \Sigma_i$ is non-trivial. If $c\langle XY \rangle \in \Sigma^+$, then $X \ {}_{s}\!\!\rightarrow Y, c\langle XY \rangle \in \Sigma^+$ and an application of (kT) would show that $p\langle X \rangle \in \Sigma^+$, a contradiction. Consequently, the claim follows also in this case.

Assume that the non-trivial c-FD $X \ {}_{w}\!\!\rightarrow Z \in \Sigma_{i+1} - \Sigma_i$ was inferred from $X \ {}_{w}\!\!\rightarrow Y$ and $XY \ {}_{w}\!\!\rightarrow Z$ by application of (T). As $X \ {}_{w}\!\!\rightarrow Z$ is non-trivial there must be some $A \in Z - (X \cap T_S)$. If $A \in Y$, then the c-FD $X \ {}_{w}\!\!\rightarrow Y \in \Sigma_i$ is non-trivial. As $c\langle X \rangle \notin \Sigma^+$ the claim follows in this case. If $A \notin Y$, then $A \in Z - (XY \cap T_S)$ and the c-FD $XY \ {}_{w}\!\!\rightarrow Z \in \Sigma_i$ is non-trivial. If $c\langle XY \rangle \in \Sigma^+$, then $X \ {}_{w}\!\!\rightarrow Y, c\langle XY \rangle \in \Sigma^+$ and an application of (kT) would show that $c\langle X \rangle \in \Sigma^+$, a contradiction. Consequently, the claim follows also in this case.

This concludes the proof sketch. ∎

Theorem 27 (Theorem 7 restated)
The problem whether a given schema is in BCNF can be decided in time quadratic in the input.

Proof This follows directly from Theorem 6 and Theorem 3. ▮

Lemma 4
Consider a schema (T, T_S, Σ) and table $I = \{t_0, t_1\}$ over (T, T_S).

i) Let $\Sigma \not\models p\langle X\rangle$ and t_0, t_1 constructed as

$$t_i[A] = \begin{cases} 0 & \textit{if } A \in X^{*p} \textit{ and } A \in X \cup T_S \\ \bot & \textit{if } A \in X^{*p} \textit{ and } A \notin X \cup T_S \\ i & \textit{if } A \notin X^{*p} \end{cases}$$

Then Σ holds on I.

ii) Let $\Sigma \not\models c\langle X\rangle$ and t_0, t_1 constructed as

$$t_i[A] = \begin{cases} 0 & \textit{if } A \in XX^{*c} \textit{ and } A \in T_S \\ \bot & \textit{if } A \in XX^{*c} \textit{ and } A \notin T_S \\ i & \textit{if } A \notin XX^{*c} \end{cases}$$

Then Σ holds on I.

Proof For each case, we consider the different types of constraints in Σ. We may assume here that FDs are of the form $Y \to A$, with A denoting a single attribute.

i)
- $p\langle Y\rangle \in \Sigma$ violated: Then $Y \subseteq X$, so $p\langle Y\rangle$, and thus Σ, implies $p\langle X\rangle$. Contradiction.
- $Y \;{}_s\!\!\to A \in \Sigma$ violated: Then $Y \subseteq X$, so $A \in Y^{*p} \subseteq X^{*p}$. But then $t_0[A] = t_1[A]$ so $Y \;{}_s\!\!\to A$ holds on I.
- $c\langle Y\rangle \in \Sigma$ violated: Then $Y \subseteq X^{*p}$, so $\Sigma \models c\langle X^{*p}\rangle$. But by Theorem 24 this mean $\Sigma \models p\langle X\rangle$.
- $Y \;{}_w\!\!\to A \in \Sigma$ violated: Then $Y \subseteq X^{*p}$, so $A \in Y^{*c} \subseteq (X^{*p})^{*c} \subseteq X^{*p}$ by Lemma 3. But then $t_0[A] = t_1[A]$ so $Y \;{}_w\!\!\to A$ holds on I.

ii)
- $p\langle Y\rangle \in \Sigma$ violated: Then $Y \subseteq XX^{*c} \cap T_S$, so $\Sigma \models p\langle XX^{*c} \cap T_S\rangle \models c\langle XX^{*c} \cap T_S\rangle \models c\langle XX^{*c}\rangle$. By Theorem 24 it follows that $\Sigma \models c\langle X\rangle$, a contradiction.
- $Y \;{}_s\!\!\to A \in \Sigma$ violated: Then $Y \subseteq XX^{*c} \cap T_S$, so $A \in (XX^{*c} \cap T_S)^{*p} = (XX^{*c} \cap T_S)^{*c} \subseteq (XX^{*c})^{*c} = X^{*c}$ with the equalities following from the Strengthening (S) and Pseudo-Transitivity (T) rules, respectively. But then $t_0[A] = t_1[A]$ so $Y \;{}_s\!\!\to A$ holds on I.
- $c\langle Y\rangle \in \Sigma$ violated: Then $Y \subseteq XX^{*c}$, so $\Sigma \models c\langle XX^{*c}\rangle$. But by Theorem 24 this mean $\Sigma \models c\langle X\rangle$.

- $Y \ {}_{w}\!\!\rightarrow A \in \Sigma$ violated: Then $Y \subseteq XX^{*c}$, so $A \in Y^{*c} \subseteq (XX^{*c})^{*c} = X^{*c}$ (by Pseudo-Transitivity). But then $t_0[A] = t_1[A]$ so $Y \ {}_{w}\!\!\rightarrow A$ holds on I.

This concludes the proof. ▮

Theorem 28 (Theorem 9 restated)
A schema (T, T_S, Σ) is in RFNF if and only if (T, T_S, Σ) is in BCNF.

Proof By definition 4, redundancy-freeness means existence of a p_0-value substitution for every I and p_0.

(if) Let Σ imply $p\langle X\rangle$ or $c\langle X\rangle$ for every non-trivial possible or certain FD $X \rightarrow Y \in \Sigma^+$, respectively. Let further I be any instance over (T, T_S, Σ) and p_0 any position in I. We construct a p_0-value substitution I' of I by assigning a new value $v'_0 \neq \bot$ to p_0 which does not occur in I. To show that I' is indeed a p_0-value substitution of I, we must show that Σ holds on I'.

Assume there exist tuples $t'_0, t_1 \in I'$ that violate some key or FD $\sigma \in \Sigma^+$. As σ holds on I, either t'_0 or t_1 must contain p_0, say t'_0, and denote by t_0 the tuple in I containing p_0. If $\sigma = {}_{(p/c)}\langle X\rangle$ is a key, then $t'_0[X] \sim t_1[X]$ implies $t_0[X] \sim t_1[X]$, so t_0, t_1 would already violate σ. If $\sigma = X \rightarrow Y$ is a non-trivial FD, and $p_0 \notin X$[1], then $t'_0[X] = t_0[X] \not\sim t_1[X]$ as ${}_{(p/c)}\langle X\rangle$ holds on I by assumption. That leaves the case $p_0 \in X$. If $\sigma = X \ {}_{s}\!\!\rightarrow Y$ is a possible FD, then $t'_0[X] \not\sim_s t_1[X]$ due to choice of v'_0, so $X \ {}_{s}\!\!\rightarrow Y$ is not violated by t'_0, t_1. If $\sigma = X \ {}_{w}\!\!\rightarrow Y$ is a certain FD, then $t'_0[X] \sim_w t_1[X]$ can only hold if $t_1[p_0] = \bot$. But this implies $t_0[X] \sim_w t_1[X]$, which would violate $c\langle X\rangle$ in I, and thus Σ.

(only if, possible) Let Σ not imply $p\langle X\rangle$ for a non-trivial possible FD $X \ {}_{s}\!\!\rightarrow Y \in \Sigma^+$. Consider the instance $I = \{t_0, t_1\}$ over (T, T_S, Σ) with

$$t_i[A] = \begin{cases} 0 & \text{if } A \in X^{*p} \text{ and } A \in X \cup T_S \\ \bot & \text{if } A \in X^{*p} \text{ and } A \notin X \cup T_S \\ i & \text{if } A \notin X^{*p} \end{cases}$$

which is valid by Lemma 4.

As $X \ {}_{s}\!\!\rightarrow Y$ is non-trivial, there must exist some attribute $A \in Y - X$. Consider now the position $p_0 := (t_0, A)$. If we substitute the value at p_0 with any other value, then this new instance I' violates $X \ {}_{s}\!\!\rightarrow Y$. Hence no p_0-value substitution exists, so p_0 is redundant and (T, T_S, Σ) not redundancy-free.

(only if, certain) Let Σ not imply $c\langle X\rangle$ for a non-trivial certain FD $X \ {}_{w}\!\!\rightarrow Y \in \Sigma^+$. Consider the instance $I = \{t_0, t_1\}$ over (T, T_S, Σ) with

$$t_i[A] = \begin{cases} 0 & \text{if } A \in XX^{*c} \text{ and } A \in T_S \\ \bot & \text{if } A \in XX^{*c} \text{ and } A \notin T_S \\ i & \text{if } A \notin XX^{*c} \end{cases}$$

[1] Here, and in the following, we silently identify p_0 with its column.

which is valid by Lemma 4. As $X \ {}_{w}\!\!\rightarrow Y$ is non-trivial, there must exist some attribute $A \in Y - (X \cap T_S)$. Consider again the position $p_0 := (t_0, A)$. If we substitute the value at p_0 with any other value, then this new instance I' violates $X \ {}_{w}\!\!\rightarrow Y$: $t_0[A] \neq t_1[A]$ and $t_0[X] \sim_w t_1[X]$ hold whether $A \notin X$ or $A \notin T_S$. Hence no p_0-value substitution exists, so p_0 is redundant and (T, T_S, Σ) not redundancy-free.

This concludes the proof. ▮

C Normalization: Proofs

Theorem 29 (Theorem 11 restated)
Let (T, T_S, Σ) be a schema with $\Sigma \models X \ {}_{w}\!\!\rightarrow Y, X \cap Y = \emptyset$, and I an instance over (T, T_S, Σ). Then

$$I[[X(T - XY)]] \bowtie I[XY] = I$$

Proof Let t occur in I exactly n times. Then $t[X(T-XY)]$ appears in $I[[X(T - XY)]]$ at least n times, and $t[XY]$ appears in $I[XY]$. Thus t appears in $I[[X(T - XY)]] \bowtie I[XY]$ at least n times, showing $I[[X(T - XY)]] \bowtie I[XY] \supseteq I$.

Conversely, let t occur in $I[[X(T - XY)]] \bowtie I[XY]$ exactly n times. Then there must exist exactly n tuples $t_i \in I$ with $t_i[X(T - XY)] = t[X(T - XY)]$. Furthermore, there must exist some tuple $t_Y \in I$ with $t_Y[XY] = t[XY]$. Since $X \subseteq X(T - XY)$ and $X \ {}_{w}\!\!\rightarrow Y$ holds on I, we have $t_i[Y] = t_Y[Y] = t[Y]$ for all $t_i \in I$ with $t_i[X(T - XY)] = t[X(T - XY)]$. Hence $t = t_i$ occurs in I exacty n times, showing $I[[X(T - XY)]] \bowtie I[XY] \subseteq I$. ▮

Theorem 30 (Theorem 12 restated) *Let (T, T_S, Σ) be a schema with $\Sigma \models X \ {}_{w}\!\!\rightarrow Y, X \cap Y = \emptyset$, and I an instance over (T, T_S, Σ). If $\Sigma \models X \ {}_{w}\!\!\rightarrow XY$, then $c\langle X \rangle$ holds on $I[XY]$.*

Proof Let $\Sigma \models X \ {}_{w}\!\!\rightarrow XY$, and let $t_{XY}, t'_{XY} \in I[XY]$ with $t_{XY}[X] \sim_w t'_{XY}[X]$. Then there must exist $t, t' \in I$ with $t[XY] = t_{XY}[XY]$ and $t'[XY] = t'_{XY}[XY]$. It follows that $t[X] \sim_w t'[X]$. Since $X \ {}_{w}\!\!\rightarrow XY$ holds on I we have $t[XY] = t'[XY]$ and thus

$$t_{XY}[XY] = t[XY] = t'[XY] = t'_{XY}[XY]$$

Since $I[XY]$ is a set, this means $t =_{id} t'$. Thus $c\langle X \rangle$ holds on $I[XY]$. ▮

Theorem 31 (Theorem 14 restated)
A schema (T, T_S, Σ) is in SQL-BCNF if and only if for every external *c-FD $X \ {}_{w}\!\!\rightarrow Y \in \Sigma$, $c\langle X \rangle \in \Sigma^+$. It can therefore be decided in time quadratic in the input whether a given schema is in SQL-BCNF.*

Proof (only if) This direction follows directly from the definition of SQL-BCNF as $\Sigma \subseteq \Sigma^+$.

(if) Here, we need to show that if (i) for all external c-FDs $X \ _w\!\!\rightarrow Y \in \Sigma$, $c\langle X\rangle \in \Sigma^+$, then it is also true that (i') for all external c-FDs $X \ _w\!\!\rightarrow Y \in \Sigma^+$, $c\langle X\rangle \in \Sigma^+$. We show that not (i') implies not (i) by induction over the inference length. For that purpose, we assume w.L.o.G. that $\Sigma = \Sigma_0 \subset \Sigma_1 \subset \cdots \subset \Sigma_k = \Sigma^+$ where Σ_{i+1} results from Σ_i by application of one of the inference rules in $\mathfrak{FK}$ to elements from Σ_i. If $k = 0$, then $\Sigma = \Sigma^+$ and there is nothing to show. Otherwise, we assume that our hypothesis holds for all $j = 0, \ldots, i < k$, and we establish that both claims also hold for $j = i + 1$. The proof proceeds by analyzing all inference rules in $\mathfrak{FK}$ which could have produced the single dependency in $\Sigma_{i+1} - \Sigma_i$. Due to lack of space we illustrate the proof on an interesting case.

Assume that the external c-FD $X \ _w\!\!\rightarrow Z \in \Sigma_{i+1} - \Sigma_i$ was inferred from $X \ _w\!\!\rightarrow Y$ and $XY \ _w\!\!\rightarrow Z$ by application of (T). As $X \ _w\!\!\rightarrow Z$ is external there must be some $A \in Z - X$. If $A \in Y$, then the c-FD $X \ _w\!\!\rightarrow Y \in \Sigma_i$ is external. As $c\langle X\rangle \notin \Sigma^+$ the claim follows in this case. If $A \notin Y$, then $A \in Z - XY$ and the c-FD $XY \ _w\!\!\rightarrow Z \in \Sigma_i$ is external. If $c\langle XY\rangle \in \Sigma^+$, then $X \ _w\!\!\rightarrow Y, c\langle XY\rangle \in \Sigma^+$ and an application of (kT) would show that $c\langle X\rangle \in \Sigma^+$, a contradiction. Consequently, the claim follows also in this case.

This concludes the proof sketch. ▮

Theorem 32 (Theorem 15 restated)
A schema (T, T_S, Σ) is in VRNF if and only if it is in SQL-BCNF.

Proof For ease of presentation we shall assume that all FDs in Σ are singular (of the form $X \rightarrow A$). Absence of value redundancy means existence of a p_0-value substitution for every I and p_0 where $I[p_0] \neq \perp$.

(if) Let Σ imply $c\langle X\rangle$ for every external FD $X \ _w\!\!\rightarrow A \in \Sigma$, respectively. Let further I be any instance over (T, T_S, Σ) and p_0 any position in I with $I[p_0] \neq \perp$. We construct a p_0-value substitution I' of I by assigning a new value $v'_0 \neq \perp$ to p_0 which does not occur in I. To show that I' is indeed a p_0-value substitution of I, we must show that Σ holds on I'.

Assume there exist tuples $t'_0, t_1 \in I'$ that violate some key or FD $\sigma \in \Sigma$. As σ holds on I, either t'_0 or t_1 must contain p_0, say t'_0, and denote by t_0 the tuple in I containing p_0. If $\sigma = c\langle X\rangle$ is a key, then $t'_0[X] \sim_w t_1[X]$ implies $t_0[X] \sim_w t_1[X]$, so t_0, t_1 would already violate σ. If $\sigma = X \ _w\!\!\rightarrow A$ and $p_0 \neq A$ then σ holds on I' since it holds on I. If $\sigma = X \ _w\!\!\rightarrow A$ and $p_0 = A \notin X$ then σ is external, so by assumption $c\langle X\rangle$ holds on I. Thus $t'_0[X] = t_0[X] \not\sim_w t_1[X]$, so σ holds trivially on I'.

That leaves the case $\sigma = X \ _w\!\!\rightarrow A$ and $p_0 = A \in X$. Here $t'_0[X] \sim_w t_1[X]$ can only hold if $t_1[p_0] = \perp$. As σ holds on I we must have $t_0[p_0] = \perp$, violating our choice of p_0.

(only if) Let Σ not imply $c\langle X\rangle$ for an external certain FD $X \ _w\!\!\rightarrow A \in \Sigma$.

Consider the instance $I = \{t_0, t_1\}$ over (T, T_S, Σ) with

$$t_i[B] = \begin{cases} 0 & \text{if } B \in XX^{*c} \text{ and } B \in T_S \\ \bot & \text{if } B \in XX^{*c} \text{ and } B \notin T_S \\ i & \text{if } B \notin XX^{*c} \end{cases}$$

which is valid by Lemma 4. If $A \notin T_S$, modify I by setting $t_0[A] = t_1[A] = 0$. As Σ contains only certain keys and FDs, the resulting instance is still valid.

Now consider the position $p_0 := (t_0, A)$, noting that $t_0[A] \neq \bot$. If we substitute the value at p_0 with any other value, then this new instance I' violates $X \ {}_w\!\!\rightarrow A$: $t_0'[A] \neq t_1[A]$ and $t_0[X] \sim_w t_1[X]$. Hence no p_0-value substitution exists, so p_0 and thus (T, T_S, Σ) are value redundant.

This concludes the proof. ▮

Lemma 5
Let Σ be a set of total FDs and certain keys over (T, T_S). Then for every LHS-minimal FD $X \ {}_w\!\!\rightarrow Y$ implied by Σ, the total FD $X \ {}_w\!\!\rightarrow XY$ is also implied by Σ.

Proof By Theorem 23 we may assume that Σ contains only total FDs (no keys). Consider now Algorithm c-Closure, which computes the c-closure $\mathcal{C}$ of X. Since $X \ {}_w\!\!\rightarrow Y$ is LHS-minimal, every column $A \in X - T_S$ must participate in the LHS of some FD $Y \ {}_w\!\!\rightarrow Z \in \Sigma$ with $Y \subseteq \mathcal{C} \cup X$. Since all FDs in Σ are total, we must have $A \in Z \subseteq \mathcal{C}$. Thus Σ implies $X \ {}_w\!\!\rightarrow X - T_S$. This implies $\Sigma \models X \ {}_w\!\!\rightarrow X$ and together with $\Sigma \models X \ {}_w\!\!\rightarrow Y$ the claim follows. ▮

Theorem 33 (Theorem 16 restated)
Algorithm 3 is correct, and terminates in exponential time in the input.

Proof By Lemma 5 it follows that a total FD $X \ {}_w\!\!\rightarrow XY$ on T_i exists in line 4; if $X \ {}_w\!\!\rightarrow Y$ is not LHS-minimal, replace X with LHS-minimal $X' \subset X$. From Theorem 11 it follows that the decomposition $\{\ [[T_i - XY]], [XY]\ \}$ of $[[T_i]]$ is lossless. For decomposition of $[T_i]$, consider that $[T_i]$ is a relation and $\Sigma \models T_i - XY \ {}_w\!\!\rightarrow T_i$. Thus $T_i - XY$ is a certain key on $[T_i]$, so projection and multi-set projection onto $T_i - XY$ are identical. It follows (again by Theorem 11) that $\{\ [T_i - XY], [XY]\ \}$ is a lossless decomposition of $[T_i]$. This shows losslessness as loop invariant. The condition of the while loop ensures that any returned decomposition is in VRNF.

Next we show termination. Consider the following partial order between subschemas:

$$\begin{aligned} [T_i] < [[T_j]] &\Leftrightarrow T_i \subseteq T_j \\ [T_i] < [T_j] &\Leftrightarrow T_i \subset T_j \\ [[T_i]] < [[T_j]] &\Leftrightarrow T_i \subset T_j \end{aligned}$$

During each iteration, a subschema is replaced with strictly smaller subschemata. As the number of distinct subschemata is finite, no infinite replacement chains can exist. Thus Algorithm 3 terminates.

The exponential time bound is a result of having to compute the projected set of constraints $\Sigma[X]$ to determine whether a sub-schema is in VRNF. For this purpose, it is enough to check for all subsets $Y \subseteq X$ whether Σ implies $p\langle Y\rangle$ and/or $c\langle Y\rangle$, and to check for all subsets $YA \subseteq X$ whether Σ implies $Y \;{}_{s}\!\!\rightarrow A$ and/or $Y \;{}_{w}\!\!\rightarrow A$. As implication can be decided in linear time for each of the exponentially many cases, the claim follows. ▮

Theorem 34 (Theorem 17 restated)
The problem of deciding whether the projection of a given schema onto a given attribute set is in SQL-BCNF (VRNF) is co-NP complete.

Proof We prove the claim for SQL-BCNF, and the claim for VRNF then follows by Theorem 15.

Co-NP hardness follows by a reduction from the co-NP hard problem of deciding whether the projection of a given relation schema is in BCNF [2].

The problem is in co-NP as we can guess an external c-FD $X \;{}_{w}\!\!\rightarrow Y \in \Sigma$ and verify in polynomial time that $c\langle X\rangle \notin \Sigma^{+}$. ▮