# Embedded Functional Dependencies and Data-completeness Tailored Database Design

Ziheng Wei
Department of Computer Science
The University of Auckland
Auckland, New Zealand
z.wei@auckland.ac.nz

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

## ABSTRACT

We establish a robust schema design framework for data with missing values. The framework is based on the new notion of an embedded functional dependency, which is independent of the interpretation of missing values, able to express completeness and integrity requirements on application data, and capable of capturing many redundant data value occurrences. We establish axiomatic and algorithmic foundations for reasoning about embedded functional dependencies. These foundations allow us to establish generalizations of Boyce-Codd and Third normal forms that do not permit any redundancy in any future application data, or minimize their redundancy across dependency-preserving decompositions, respectively. We show how to transform any given schema into application schemata that meet given completeness and integrity requirements and the conditions of the generalized normal forms. Data over those application schemata are therefore fit for purpose by design. Extensive experiments with benchmark schemata and data illustrate our framework, and the effectiveness and efficiency of our algorithms, but also provide quantified insight into database schema design trade-offs.

**PVLDB Reference Format:**
Ziheng Wei, Sebastian Link. Embedded Functional Dependencies and Data-completeness Tailored Database Design. *PVLDB*, 12(11): 1458-1470, 2019.
DOI: https://doi.org/10.14778/3342263.3342626

## 1. INTRODUCTION

SQL continues to be the de-facto industry standard and choice for data management. This holds true even after several decades of use and even in the light of new application data such as complex-value data, Web data, big data, or uncertain data. Nevertheless, even the simplest extensions to the underlying relational model cause significant issues in database practice. A prime example is the handling of incomplete data, which has attracted continuous interest from academics and practitioners over decades. While many advances can be celebrated, it is still unclear what a right notion of a query answer of incomplete data constitutes. In this research we are interested in the design of database schemata in the presence of incomplete data. SQL handles incomplete data by the use of a null marker, denoted by $\perp$, which indicates a missing data value occurrence on an attribute. The null marker is distinguished from actual data values, and treated differently. Different null marker occurrences, however, are treated uniformly in SQL to avoid processing overheads. Occurrences of null markers are on the rise for modern applications such as data integration. Indeed, in order to accommodate the integration of data from different schemata, frequent uses of null markers are necessary to conform to the structure of the underlying schema that integrates the data.


*Proceedings of the VLDB Endowment,* Vol. 12, No. 11
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3342263.3342626

### 1.1 Desiderata

The primary aim of database design is to find a schema that facilitates the processing of the future application workload as well as possible. The well-known database design framework for relational databases is centered around the notion of data redundancy [6, 29, 31]. In practice, most redundant data value occurrences originate from functional dependencies (FDs) [8]. These dependencies can capture important integrity requirements of the underlying application domain. Informally, an FD $X \to Y$ expresses that every pair of tuples with matching values on all the attributes in $X$ also has matching values on all the attributes in $Y$. Relational database design provides a framework to compute schemata in which integrity requirements can be enforced efficiently during updates. However, real-world requirements on the completeness of data are not accommodated, as no missing data is permitted to occur at all.

We will now summarize which properties an FD should exhibit to advance database design in practice. There has been a plethora of research on finding suitable extensions for the notion of an FD to accommodate missing data. This has led to useful notions such as weak and strong FDs [24, 25], no information FDs [4, 16], or possible and certain FDs [19, 21]. However, all of these notions assume that the same interpretation applies to all occurrences of the null marker in the given data set, such as "value does not exist", "value exists but is unknown", or "no information". This sensitivity to a fixed null marker interpretation is difficult to justify in practice. Furthermore, it is not clear why we would want null marker occurrences to have an impact on the validity of an FD at all. Instead, it is more sensible to make the seman-

**Table 1: Sample $r$**

| *p(arent)* | *b(enefit)* | *c(hild)* |
|---|---|---|
| Homer | **610** | Bart |
| Homer | **610** | Lisa |
| Homer | 915 | $\perp$ |

**Table 2: Lossless redundancy-eliminating decomposition of $r$ from Table 1 for applications that require complete data values on $c$, $p$, and $b$**

| *p(arent)* | *b(enefit)* |
|---|---|
| Homer | 610 |

| *p(arent)* | *c(hild)* |
|---|---|
| Homer | Bart |
| Homer | Lisa |

tics of FDs dependant on actual data value occurrences only. This would mean the FD $X \to Y$ is satisfied whenever for every pair of records that have no null marker occurrences in any columns in the set union $XY$ of $X$ and $Y$, matching values on all attributes in $X$ imply matching values on all the attributes in $Y$. This notion is robust under different interpretations of null marker occurrences. It is beyond guesswork around the meaning of missing data in relations.

Hence, *firstly* we desire a notion of an FD that is ignorant of missing data, in contrast to all previous notions that are sensitive to it. *Secondly*, for a notion of an FD to be practically useful for schema design it should capture many redundant data value occurrences and facilitate lossless decompositions to eliminate them. *Thirdly*, applications for real-world data have not only integrity requirements but also completeness requirements. In contrast to previous work, we require our notion of an FD to accommodate completeness requirements in addition to the integrity requirements. *Fourthly*, we expect that the resulting schema design framework coincides with the well-known relational framework for the fragment of the data that meet the completeness requirements of the applications. In fact, the point of the framework is to tailor relational schema designs to the completeness and integrity requirements of applications.

## 1.2 Motivating example

We illustrate our desiderata on the simple example of Table 1. The schema collects information about the benefit that parents receive for all their children together. As the benefit changes, updates must be processed efficiently.

**Robustness.** It is easy to observe that the FD $p \to b$ does not hold, as the second and third tuple have matching (non-null) values on $p$ but different values on $b$. Since no missing data is involved, this is true for any interpretation of null markers. Instead, for the FD $cp \to b$ the situation is quite different. If the sole occurrence of $\perp$ in $r$ is interpreted as "value does not exist", then the FD should be true. If its interpretation is "value exists but unknown", then there are possible worlds of $r$ (originating from the replacement of $\perp$ by actual values) that satisfy the FD and others that do not. Hence, under this interpretation, the FD is possible but not certain to hold in $r$. If we interpret $\perp$ as "no information", then the FD holds because there are no tuples with matching (non-null) values on $p$ and $c$. Hence, if we do not know which interpretation applies to a given null marker occurrence, or if different null marker occurrences have different interpretations, then the semantics of an FD is not robust.

**Data redundancy and their elimination by lossless decompositions.** Under our robust semantics, the FD $cp \to b$ is satisfied. However, since there are no tuples with matching (non-null) values on $c$ and $p$, this FD does not capture any redundant data value occurrences. In fact, we cannot express that the FD $p \to b$ actually holds on the subset of all tuples that have no missing values on attribute $c$. This novel observation leads us to the new notion of an *embedded functional dependency* (eFD). This is a statement $E : X \to Y$ where $E$ is a set of attributes and $X, Y \subseteq E$. Indeed, $E$ defines the subset of tuples $r^E \subseteq r$ that have no missing data on any of the attributes in $E$. We require $X, Y \subseteq E$ to make our notion of an FD robust, as explained before. For convenience, we sometimes simply write $E - XY : X \to Y$, understanding implicitly that all attributes in $XY$ belong to $E$. We call an eFD $E : X \to Y$ *pure* whenever $E - XY$ is non-empty. In our example, we obtain the eFD $c : p \to b$, which clarifies the roles of the attributes $c$ and $p$: $p$ functionally determine $b$ whenever $c$ (and $p$ and $b$) have no missing values. Now, our example shows that the eFD $c : p \to b$ captures redundant data value occurrences. Each of the two values is redundant in the classical sense [35] that every change to one of those values to a different value will result in a violation of the eFD $c : p \to b$. Our experiments will show that pure eFDs cause a significant number of redundant data value occurrences that cannot be captured by previously studied FDs, that is, eFDs of the special case $\varnothing : X \to Y$. In fact, we will show that pure eFDs are frequent among those eFDs that cause the most redundant data value occurrences on real-world benchmark data. The ability to capture many redundant data values enables eFDs to facilitate new lossless decompositions that can eliminate those redundancies.

**Data completeness.** The need to accommodate data quality requirements are another driver for our notion of an eFD. An eFD $E : X \to Y$ enables users to declare completeness as well as integrity requirements, and to carefully distinguish between these two data quality dimensions. In fact, this notion now allows us to separate dependence from completeness concerns: the attribute subsets $X$ and $Y$ form the actual FD $X \to Y$ which must hold on the subset $r^E$ of all tuples that have no missing data on attributes in $X$, $Y$, and $E - XY$. In fact, as $r^E$ satisfies the FD $X \to Y$ over relation schema $R$, $r^E$ is the lossless join $r^E = r^E[XY] \bowtie r^E[X(R - Y)]$ over its two projections on $XY$ and $X(R - Y)$. Hence, we can eliminate all redundant data values on $Y$ caused by the eFD $E : X \to Y$ without a loss of information for the fragment $r^E$ of our application data that meets the requirement of having complete data values on all columns in $E$. This illustrates that eFDs drive data-completeness tailored database design. Tuples that do not meet the completeness requirements, that is tuples in $r - r^E$, could be stored under the original schema $R$, or be subject to normalization approaches that are sensitive to the interpretation of null markers.

In summary, the example shows that eFDs are different from previous FDs in three aspects: they are robust under different interpretations of missing data and accommodate completeness requirements, they capture redundant data values occurrences that could not be captured before, and they facilitate lossless decompositions that eliminate redundant values for the data that is meeting the completeness requirements. Table 2 shows a lossless decomposition for the

fragment of our example relation $r$ that meets the requirements for tuples to be complete on $c$, $p$, and $b$, and eliminates the redundant data value occurrences from $r$. This is not achievable for previous notions of FDs. Hence, changes to the benefit $b$ for a parent just require one update on the decomposed relation. As the data evolves, more tuples may meet the completeness requirements. For example, if the occurrence of $\perp$ in $r$ is updated to *Maggie*, then the resulting relation violates our eFD $c : p \rightarrow b$. In response we may update both value occurrences of 610 on $b$ to 915 to reflect the new information that Homer has three children. On the decomposed schema, this would be represented by an insertion of *(Homer, Maggie)* and a single update of the value 610 on *benefit* to 915.

## 1.3 Contributions

We establish the first fully-fledged framework that brings forward schema designs tailored to the data completeness requirements of applications. (1) We propose the new notion of an *embedded functional dependency* (eFD). We demonstrate that eFDs provide an intuitive and useful notion for database schema design in practice. They a) capture an intrinsic relationship between completeness and integrity requirements not observed before, b) identify redundant data values, c) are robust under different interpretations of null markers, and d) enable users to declare completeness and integrity requirements under one framework. (2) We develop a full design theory including axiomatic and algorithmic characterizations of the implication problem for eFDs. Just like reasoning about FDs is indispensable for relational normalization, our design theory for eFDs is essential to our design framework. (3) We establish a schema design framework that accommodates completeness and integrity requirements of applications, based on suitable extensions of a) the notion of redundant data values, b) Boyce-Codd normal form, and c) Third normal form. (4) We show how to embed data completeness requirements into the relational normalization framework with no additional overheads, including BCNF decomposition and 3NF synthesis algorithms. For data that does not meet the completeness requirements, previous approaches that are sensitive to the interpretation of null markers can be applied. (5) We conduct comprehensive experiments on real-world benchmark schemata and data. In particular, we provide insight on how many redundant data values occur in the data, rank the relevance of our eFDs by how many data redundancies they cause, show how often schemata satisfy a normal form condition, how much redundancy 3NF syntheses permit, how many dependencies BCNF decompositions preserve, and how large decomposed schemata become. We consider the times of computations, and suggest by examples how data stewards can use our ranking of eFDs.

## 1.4 Organization

We explain our contributions over related work in Section 2, fix notation and propose eFDs in Section 3. Schema design foundations are developed in Section 4, normal forms are proposed in Section 5, and normalization is discussed in Section 6. An analysis of our comprehensive experiments is presented in Section 7. We conclude and outline future work in Section 8. More details can be found in the technical report [38] and at `https://bit.ly/2AoOis6`.

# 2. RELATED WORK

We review previous work to emphasize our contributions.

Firstly, our core objective is tailoring classical schema design to data-completeness requirements of applications. Hence, the achievements of classical schema design are fundamental to our work. These achievements are handsomely summarized in surveys and textbooks [7, 29]. As our article starts research on data-quality driven database design, we focus on the most common class of integrity constraints and source of data redundancy, namely functional dependencies. More general constraints such as join or inclusion dependencies are left for future work [10, 13, 26, 34]. Hence, we are interested in Boyce-Codd Normal Form (BCNF) [9, 17] and Third Normal Form (3NF) [8], with their well-known tradeoffs [7, 12, 23]: Any relation schema enjoys lossless BCNF decompositions that eliminate all data redundancy caused by FDs, but may not be dependency-preserving. On the other hand, every relation schema enjoys lossless 3NF syntheses that are guaranteed to be dependency-preserving, but may not eliminate all data redundancy. Our work subsumes all of these results as the special case where an application requires that no data values are missing, that is, when $E = R$. Important for these achievements is Vincent's classical notion of data redundancy [35], which we generalize to a notion of data redundancy under data completeness requirements. This makes it possible to clearly state and demonstrate the achievements of our normal forms. Recently, classical FD discovery was combined with classical BCNF-decomposition to drive schema normalization from data [30]. They did not consider data quality criteria, and null markers were handled like any domain value.

Secondly, schema design for data with missing values has been a long-standing open problem [15, 19, 25]. Almost exclusively, the main focus of the research has been on suitable extensions of FDs to incorporate null markers. In that area there is a plethora of research, mostly focusing on foundational aspects such as reasoning. Among those extensions, there are approaches where null-free sub-schemata have been considered for reasoning about FDs [4, 16]. That work is different from our approach, and also focused on reasoning rather than schema design. In particular, the approach is more restricted because null-free sub-schemata do not permit any null marker occurrences in the columns of the subschema. Instead, we do permit null marker occurrences in any columns, but let the application requirements decide whether such records should be considered for design decisions. An interesting FD extension are weak and strong FDs that hold in some or all possible worlds of data sets with missing values [24, 25]. As with other extensions, the semantics of the extended FDs depends strongly on the interpretation of the null marker occurrences. This makes it difficult to address modern applications, such as data integration, where different null marker occurrences may require different interpretations. A second limitation is that the complexity of reasoning becomes often prohibitively expensive to guarantee efficient schema designs [25]. Recently, the concept of possible and certain FDs was introduced and shown to provide suitable extensions of schema design for data with missing values, at least in terms of BCNF [19, 21]. The work contains a review and comparison of FD extensions to data with missing values. We refer the interested reader to this survey. In summary, these approaches focus on the interpretation of null markers, aiming at their inclusion

in decisions about schema design. These approaches can be criticized in different respects. Firstly, it is doubtful whether missing information should have an impact on schema design decisions. Secondly, modern applications such as data integration accommodate missing data values that require different interpretations, which makes it difficult to justify these approaches. Finally, application requirements have not been considered in these approaches, even though design decisions should be based on them. In contrast, our approach bases decisions about the design only on information that is available. That is, we design schemata beyond guesswork by considering complete data fragments. At the same time, application requirements become the primary focus point of our approach. In fact, the requirements can be declared as part of the FDs.

Thirdly, embedded unique constraints (eUCs) and embedded cardinality constraints (eCCs) have been investigated in previous work [36, 40]. Those articles investigated primarily their implication problem. In particular, the embedded version of a unique or cardinality constraint with embedding $E$ holds on a data set $r$ whenever the uniqueness or cardinality constraint holds on the scope $r^E$ of the given data set $r$. Our results on the implication problem for the combined class of eUCs and eFDs subsume the results of [40] on the individual class of eUCs. Neither eFDs nor data-completeness tailored database design have been considered before.

Finally, an important extension of classical FDs are conditional FDs (cFDs) [14], which encode data quality rules that target the cleaning of data without missing values but not schema design for data with missing values. Specifically, eFDs encode data completeness requirements and are a major source of $E$-data redundancy. This makes them important for schema design. Note that the implication problem of general cFDs is coNP-complete to decide [14], and already their consistency problem is NP-complete [14]. In contrast, classical FDs are always consistent and implication is linear-time decidable. As we show, this is also achieved by eFDs, which are therefore suited for schema design purposes from a computational point of view as well.

In summary, our work marks the first approach to tailor classical database schema designs to data-completeness requirements of applications.

# 3. EMBEDDED CONSTRAINTS

We introduce the data model and constraints.

We begin with basic terminology. A *relation schema* is a finite non-empty set $R$ of *attributes*. Each attribute $A$ of a relation schema $R$ is associated with a 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 $\perp$. In line with SQL, and to cater for all different types of missing values, the interpretation of $\perp$ is to mean "no information" [41]. 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.

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* (or *record*) over $R$ is a function $t : R \to \bigcup_{A \in R} dom(A)$ with $t(A) \in dom(A)$ for all $A \in R$. For $X \subseteq R$ let $t(X)$ denote the restriction of the tuple $t$ over $R$ to $X$. We say that a tuple $t$ is *$X$-total* (*$X$-complete*) if $t(A) \neq \perp$ for all $A \in X$. A tuple $t$ over $R$ is said to be a *total tuple* if it is $R$-total. A *relation* $r$ over $R$ is a finite set of tuples over $R$. A relation $r$ over $R$ is a *total relation* if every tuple $t \in r$ is total. The sub-set $r^X$ of $X$-total tuples in $r$ is called the *scope* of $r$ with respect to $X$, or simply the scope of $r$ when $X$ is fixed. We say that two tuples $t, t'$ over $R$ have matching values on an attribute $A \in R$ whenever $t(A) = t'(A)$.

A *key* over $R$ is a subset $X \subseteq R$. A total relation $r$ over $R$ satisfies the key $X$ over $R$ whenever there are not any two distinct tuples in $r$ with matching values on all the attributes in $X$. A *functional dependency* (FD) over $R$ is an expression $X \to Y$ with $X, Y \subseteq R$. A total relation $r$ over $R$ satisfies the FD $X \to Y$ over $R$ whenever every pair of records in $r$ with matching values on all the attributes in $X$ has also matching values on all the attributes in $Y$. The semantics of keys and FDs can be extended to relations with missing values by adopting uniformly (that is, for all null marker occurrences) either the $\perp = \perp$ or $\perp \neq \perp$ semantics. Under either of these semantics, $\perp$ is considered to be different from any actual domain value. Other semantics have been defined, and we refer the interested reader to [19, 21] for an overview of those. In a nutshell, different semantics lead to different notions of constraints each of which is useful in different contexts. However, in classical but even more in modern applications such as data integration, different null marker occurrences in a relation may require different interpretations. This makes it difficult, if not impossible, to justify any uniform interpretation of $\perp$.

In this article we take a different approach. Firstly, we let the application decide which data completeness requirements tuples must meet to be fit for use by the application. That is, we embed the data completeness requirements in the declaration of constraints. Secondly, the semantics of our constraints is based exclusively on the complete information embedded in the underlying relation. In other words, we follow the principled approach that missing values must not impact the decision whether a constraint is satisfied by the given relation or not. This decision is entirely determined by the actual data values that are available.

Similar ideas motivated us [40] to introduce embedded unique constraints (eUCs). Given a relation schema $R$, an *embedded unique* (eUC) is an expression of the form $E : U$ where $U \subseteq E \subseteq R$ holds. A relation $r$ satisfies the eUC $E : U$ iff the scope $r^E = \{t \in r \mid \forall A \in E(t(A) \neq \perp)\}$ of $r$ with respect to $E$ satisfies the key $U$. If $E = U$, the eUC $U : U$ is satisfied by relation $r$ iff the key $U$ is satisfied by $r$ using the $\perp \neq \perp$ semantics iff the SQL unique constraint on $U$ is satisfied by $r$. Of course, if $E$ contains some attribute that is not in $U$, then the semantics of eUCs cannot be captured by any other notion of a key. The decision to require $U \subseteq E$ ensures that the semantics of the eUC only depends on the complete fragments embedded in the given relation. This motivates the following definition.

Definition 1. *Given a relation schema $R$, an* embedded functional dependency *(eFD) is an expression of the form $E : X \to Y$ where $XY \subseteq E \subseteq R$ holds. A relation $r$ satisfies the eFD $E : X \to Y$ if and only if the scope $r^E$ of $r$ with respect to $E$ satisfies the functional dependency $X \to Y$.*

Given $E : X$ or $E : X \to Y$, we sometimes simply write $E - X : X$ or $E - (XY) : X \to Y$, respectively, to emphasize which additional attributes apart from those in $XY$ are required to have no missing values. The choice of $E$ is based

**Table 3: Sample snippet $r$ from *ncvoter* benchmark data set**

| tuple_id | *f(_name)* | *l(_name)* | *c(ity)* | *z(ip)* | *p(hone)* | *d(ate_register)* |
|---|---|---|---|---|---|---|
| $t_1$ | sam | anderson | green level | 27217 | ⊥ | 05/11/1940 |
| $t_2$ | ida | cheek | **burlington** | 27217 | 226 4848 | 05/11/1940 |
| $t_3$ | effie | massey | **burlington** | 27217 | 336 226 8544 | 05/11/1940 |
| $t_4$ | peggy | floyd | jackson | 27845 | 252 536 2668 | 06/15/1936 |
| $t_5$ | essie | warren | lasker | 27845 | 252 539 2866 | 05/10/1940 |
| $t_6$ | rodney | glockner | wilmington | 28405 | 910 509 3864 | 01/01/1930 |
| $t_7$ | sallie | blanchard | rose hill | 28458 | 910 289 3320 | 01/01/1930 |
| $t_8$ | joseph | cox | new bern | 28562 | ⊥ | 03/06/1935 |
| $t_9$ | joseph | cox | new bern | 28562 | 252 288 4763 | 03/06/1935 |
| $t_{10}$ | james | smith | chinquapin | 28521 | 910 285 3493 | 01/01/1936 |
| $t_{11}$ | james | smith | burlington | 27215 | 584 4202 | 05/06/1940 |
| $t_{12}$ | dorothy | faucette | mebane | 27302 | 919 563 1285 | 4/05/1940 |
| $t_{13}$ | dorothy | allred | mebane | 27302 | 563 1426 | 05/06/1940 |
| $t_{14}$ | eloise | croom | kinston | 28504 | 252 569 4436 | 05/02/1940 |
| $t_{15}$ | ⊥ | croom | kinston | 28504 | 252 569 9516 | 05/04/1940 |

on several factors, such as completeness requirements and the target of redundant data values.

Consider our running example from Table 3 where the underlying schema $R$ consists of attributes $c$, $d$ , $f$, $l$, $p$, and $z$. The eFD $p : dz \to c$ is satisfied by $r$ because the FD $dz \to c$ holds on the scope $r^{cdpz} = r - \{t_1, t_8\}$. In fact, all people who provided some phone number and registered on the same day under the same zip code also used the same city alias. However, the eFD $\varnothing : dz \to c$ does not hold on $r$ because the FD $dz \to c$ does not hold on the scope $r^{cdz} = r$. In fact, there are people who registered on the same day under the same zip code, but used different alias for the city. Similarly, the eUC $p : cfl$ is satisfied by $r$ because the compound key $cfl$ is satisfied on the scope $r^{cflp} = r - \{t_1, t_8\}$. In fact, there are no two people who both provided a phone number and are registered in the same city with the same first and last name. However, the eUC $\varnothing : cfl$ is violated by $r$ because there are two different registrations in the same city with the same first and last name.

Every total relation over $R$ satisfies the FD $X \to R$ iff it satisfies the key $X$. This relationship occurs in our framework as well: A relation over $R$ satisfies the eFD $R : X \to R$ iff it satisfies the eUC $R : X$. For arbitrary $E \subseteq R$, however, only the following holds: if a relation satisfies $E : X$, then it also satisfies $E : X \to E$, but not necessarily vice versa. In fact, the relation in Table 3 satisfies the eFD $dpl : dp \to dpl$, but it violates the eUC $dpl : dp$. It does therefore not suffice for our targeted schema design framework to consider eFDs in isolation from eUCs. In particular, eFDs drive data redundancy, while eUCs inhibit data redundancy. Hence, the combined class of eFDs and eUCs will be studied. This is different from the special case of total relations where any key $X$ over $R$ (an eUC of type $R : X$) can be expressed by the FD $X \to R$ (an eFD of type $R : X \to R$).

For our example we regard the eFD $p : dz \to c$ as a meaningful constraint of our application domain. That is, for people that provide some phone number and register on the same day under the same zip code we will always use the same city alias. Hence, different city alias may only be associated with the same zip code for people on different registration dates or who prefer not to provide a phone number. In this case, there are relations that exhibit data redundancy. Indeed, each of the two bold *city* occurrences of 'burlington' in tuples $t_2$ and $t_3$ in Table 3 is redundant. However, such redundant values are intrinsically linked to the requirement that the tuples must be complete on *phone*, since the eFD does not apply otherwise. This link between data redundancy and data completeness requirements is encoded explicitly in the eFDs. In what follows, we will develop a full-fledged normalization framework that tailors classical schema design to data completeness requirements.

# 4. FOUNDATIONS

This section establishes axiomatic and algorithmic characterizations of the implication problem for eUCs and eFDs. The linear-time decidability we establish is important for the schema design framework we will develop subsequently.

Let $\Sigma \cup \{\varphi\}$ denote a set of eUCs and eFDs over relation schema $R$. We say that $\Sigma$ *implies* $\varphi$, denoted $\Sigma \models \varphi$, iff every relation over $R$ that satisfies all $\sigma \in \Sigma$ also satisfies $\varphi$. The *implication problem* for a class $\mathcal{C}$ of constraints is to decide, for arbitrary $R$ and $\Sigma \cup \{\varphi\}$ in $\mathcal{C}$, whether $\Sigma$ implies $\varphi$. Strictly speaking, the implication problem we have just defined is the finite implication problem because we restrict relations to be finite. Permitting also infinite relations would lead us to the unrestricted implication problem. For our class of constraints, however, it is easy to see that finite and unrestricted implication problems coincide. We will therefore speak of *the* implication problem.

## 4.1 Axiomatic Characterization

Firstly, we would like to obtain an axiomatization for eUCs and eFDs which extends the well-known Armstrong axioms [3]. An axiomatization does not only help us understand the interaction of the constraints, but also enables us to prove that our syntactic normal form proposal captures precisely the semantic normal form proposal in which no redundant data value can ever occur in any future database instance. This is an important use case of axiomatizations. The definitions of inference from a system $\mathfrak{S}$ ($\vdash_{\mathfrak{S}}$), as well as the definitions of *sound* and *complete* sets of inference rules are standard [29, 31].

Table 4 shows three axiomatizations. The top box is one for eUCs alone [40], the middle box is one for eFDs alone, and all boxes form the axiomatization $\mathfrak{E}$ for eUCs and eFDs together. The following rules

$$\frac{E : X \to YZ}{E : X \to Y}\ \text{(eFD decompose)} \qquad \frac{E : X \to Y \quad E : X \to Z}{E : X \to YZ}\ \text{(eFD union)} \qquad \frac{E : X \to Y}{EE' : X \to Y}\ \text{(eFD add-on)}$$

follow from $\mathfrak{E}$. Proofs are in [38].

**Table 4: Axiomatizations for eUCs and eFDs**

| | | |
|---|---|---|
| $\frac{}{R : R}$ (trivial ekey) | $\frac{E : U}{EE' : UU'}$ (eUC extension) | |
| $\frac{}{E : XY \to X}$ (trivial eFD) | $\frac{E : X \to Y}{E : X \to XY}$ (eFD extension) | $\frac{E : X \to Y \quad E' : Y \to Z}{EE' : X \to Z}$ (eFD transitivity) |
| $\frac{E : X}{E : X \to E}$ (eUC to eFD) | $\frac{E : XY \quad E : X \to Y}{E : X}$ (eUC pullback) | |

**Table 5: 2-tuples for Theorem 1 proof and sample**

| | $X^+_{E,\Sigma}$ | $E - X^+_{E,\Sigma}$ | $R - E$ | | |
|---|---|---|---|---|---|
| | 0⋯0 | 0⋯0 | 0⋯0 | | |
| | 0⋯0 | 1⋯1 | ⊥ ⋯ ⊥ | | |
| $d$ | $z$ | $c$ | $p$ | $f$ | $l$ |
| 5/11/1940 | 27217 | burlington | 226 4848 | ida | cheek |
| 5/11/1940 | 27217 | green level | ⊥ | ⊥ | ⊥ |

THEOREM 1. *$\mathfrak{E}$ is a sound and complete axiomatization for the implication of eUCs and eFDs.*

The proof of Theorem 1 is based on the *closure* of an attribute set $X$ with respect to the data completeness requirement $E$ and the set $\Sigma$ of eUCs and eFDs: $X^+_{E,\Sigma} = \{A \in E \mid \Sigma \vdash_{\mathfrak{E}} E : X \to A\}$. Indeed, the completeness proof uses the two-tuple relation from Table 5 to show that $\Sigma$ does not imply $\varphi$ whenever $\varphi$ cannot be inferred from $\Sigma$ using $\mathfrak{E}$. Hence, the implication problems for eUCs and eFDs coincide, independently of whether we consider infinite relations, just finite relations, or even just relations with two tuples.

EXAMPLE 1. *Consider our running example where $R = \{c, d, f, l, p, z\}$ and $\Sigma = \{p : cfl, p : dz \to c\}$. Then $\Sigma$ implies the eUC $\varphi = c : dflpz$ as the following inference shows:*

$$\dfrac{\dfrac{\dfrac{}{cdflpz : dflpz \to dz} \quad p : dz \to c}{cdflpz : dflpz \to c} \quad \dfrac{}{cdflpz : cdflpz}}{cdflpz : dflpz}$$

*However, the eFD $\varnothing : dz \to c$ is not implied by $\Sigma$ as the two-tuple example in Table 5 constructed according to the general two-tuple relation from Table 5 shows.*

## 4.2 Algorithmic Characterization

Reasoning efficiently about eUCs and eFDs will help us decide if a given schema meets a normal form condition, or transform the schema into one that meets the condition. In the relational model, FD implication can be decided in linear time. We will achieve the same for eUCs and eFDs. Many other tasks, including data profiling, transaction processing, and query optimization, benefit from the ability to efficiently decide implication.

Let $E$ denote an attribute set that represents the completeness requirements of a given application. The technical underpinning of our framework translates every set $\Sigma$ of eUCs and eFDs into a set $\Sigma[E]$ of FDs. The translation makes it possible to utilized any existing algorithms for deciding FD implication to decide implication of eUCs and eFDs. The translation is given next by the following definition.

DEFINITION 2. *For a given set $\Sigma$ of eUCs and eFDs over relation schema $R$, and a given attribute set $E \subseteq R$, let $\Sigma[E] := \{X \to R \mid \exists E' \subseteq E(E' : X \in \Sigma)\} \cup \{X \to Y \mid \exists E' \subseteq E(E' : X \to Y \in \Sigma)\}$ denote the* (E,FD)-reduct *of $\Sigma$.*

We illustrate the notion of an *(E,FD)*-reduct on our running example.

EXAMPLE 2. *Recall that $R = \{c, d, f, l, p, z\}$ and $\Sigma = \{p : cfl, p : dz \to c\}$. For $E = cdflpz = R$, the* (E,FD)*-reduct of $\Sigma$ is $\Sigma[E] = \{cfl \to dpz, dz \to c\}$.*

The significance of the *(E,FD)*-reduct is embodied in the following algorithmic characterization of the implication problem for eUCs and eFDs.

THEOREM 2. *Let $\Sigma \cup \{E : X, E : X \to Y\}$ denote a set of eUCs and eFDs over relation schema $R$. Then:*

1. *$\Sigma \models E : X \to Y$ if and only if $\Sigma[E] \models X \to Y$*
2. *$\Sigma \models E : X$ if and only if a) $E = R = X$, or b) there is some $E' : X' \in \Sigma$ such that $E' \subseteq E$ and $X' \subseteq X^+_{\Sigma[E]}$.*

Here, *1.* says that the eFD $E : X \to Y$ is implied by the eUC/eFD set $\Sigma$ if and only if the FD $X \to Y$ is implied by the *(E,FD)*-reduct $\Sigma[E]$ of $\Sigma$. Furthermore, *2.* says that the eUC $E : X$ is implied by the eUC/eFD set $\Sigma$ if and only if the eUC $E : X$ is the trivial eUC $R : R$, or there is another eUC $E' : X'$ in $\Sigma$ such that $E' \subseteq E$ and the *(E,FD)*-reduct $\Sigma[E]$ implies the FD $X \to X'$.

An analysis of Theorem 2 results in the proposal of Algorithm 1 for deciding our implication problem. If $\varphi = R : R$, we answer yes (lines 3/4). Otherwise, standard algorithms compute the closure $X^+_{\Sigma[E]}$ of the attribute set $X$ given $\Sigma[E]$ (lines 6/7). If $\varphi$ is an eUC and *2.* in Theorem 2 is met, then we answer yes (lines 8/9). If $\varphi$ is an eFD and *1.* in Theorem 2 is met, then we answer yes (lines 11/12). Otherwise, we answer no (lines 13/14).

**Algorithm 1** Deciding Implication

```
1: Input:  Set Σ ∪ {φ} of eUCs and eFDs over schema R
2: Output: { Yes , if Σ ⊨ φ
           { No  , otherwise
3: if φ = R : R then
4:     return(Yes);
5: else
6:     if φ = E : X or φ = E : X → Y then
7:         Compute X⁺_{Σ[E]};              ▷ FD attribute set closure
8:         if φ = E : X ∧ ∃E' : X' ∈ Σ(E' ⊆ E ∧ X' ⊆ X⁺_{Σ[E]}) then
9:             return(Yes);
10:        else
11:            if φ = E : X → Y ∧ Y ⊆ X⁺_{Σ[E]} then
12:                return(Yes);
13:            else
14:                return(No);
```

The soundness of Algorithm 1 follows from Theorem 2, linear time decidability from that of FD implication [5], and PTIME-hardness from a reduction of HORN-SAT [11, 18].

COROLLARY 1. *The implication problem for eUCs and eFDs is complete in PTIME. On input $(\Sigma \cup \{\varphi\}, R)$, Algorithm 1 decides implication $\Sigma \models \varphi$ in time $\mathcal{O}(|\Sigma \cup \{\varphi\}|)$.*

Note that the PTIME-hardness means that deciding implication for eUCs and eFDs is at least as hard as any other decision problem for which there is some deterministic Turing machine that can decide any instance of the problem in polynomial time.

EXAMPLE 3. *In our running example* $R = \{c, d, f, l, p, z\}$ *and* $\Sigma = \{p : cfl, p : dz \to c\}$, $\Sigma$ *implies* $\varphi = c : dflpz$ *since* $\Sigma[cdflpz] = \{cfl \to dpz, dz \to c\}$, *there is some eUC* $E' = cflp : cfl = X' \in \Sigma$ *such that* $E' \subseteq E$, *and* $X' \subseteq (dflpz)^+_{\Sigma[cdflpz]} = cdflpz$, *which means* 2. *of Theorem 2 is met. The eFD* $\varnothing : dz \to c$ *is not implied by* $\Sigma$ *as* $\Sigma[E] = \Sigma[cdz] = \varnothing$, *and* $dz \to c$ *is not implied by* $\Sigma[E]$, *which means* 1. *of Theorem 2 is not met.*

As a summary, our axiomatization $\mathfrak{E}$ will enable us to formally justify the syntactic normal form proposals we will put forward in the following section, while our algorithmic characterizations will facilitate our normalization strategy to apply well-known relational decomposition and synthesis approaches to the *(E,FD)*-reduct of an input set of embedded unique constraints and embedded functional dependencies.

# 5. NORMAL FORMS

Our goal is to tailor relational schema design to data completeness requirements of applications. For that purpose, we stipulate the semantic normal form condition that no redundant data values can ever occur in any $E$-complete records on any relations that satisfy a given set of eUCs and eFDs. We will characterize this condition by generalizing the well-known Boyce-Codd normal form. Similarly, we are able to generalize the well-known 3NF to characterize the minimization of data redundancy in $E$-complete records across all dependency-preserving decompositions.

## 5.1 $E$-Redundancy Free Normal Form

Motivated by our examples, we propose notions of data redundancy that are tailored towards the requirements of records regarding their completeness. For this, we generalize the following classical proposal by Vincent [35]. Intuitively, a data value in a relation that satisfies a constraint set $\Sigma$ is redundant if every update to a different value results in a relation that violates some constraint in $\Sigma$. Formally, for relation schema $R$, attribute $A$ of $R$, tuple $t$ over $R$, and set $\Sigma$ of constraints over $R$, a *replacement* of $t(A)$ is a tuple $\bar{t}$ over $R$ such that: i) for all $\bar{A} \in R - \{A\}$ we have $\bar{t}(\bar{A}) = t(\bar{A})$, and ii) $\bar{t}(A) \neq t(A)$. For a relation $r$ over $R$ that satisfies $\Sigma$ and $t \in r$, the data value occurrence $t(A)$ in $r$ is *redundant* for $\Sigma$ if for every replacement $\bar{t}$ of $t(A)$, $\bar{r} := (r - \{t\}) \cup \{\bar{t}\}$ violates some constraint in $\Sigma$. A relation schema $R$ is in *Redundancy-Free normal form* (RFNF) for a set $\Sigma$ of constraints if there are no relation $r$ over $R$ that satisfies $\Sigma$, tuple $t \in r$, and attribute $A \in R$, such that the data value occurrence $t(A)$ is redundant for $\Sigma$ [35]. In other words, we guarantee at design time that there will never be an instance over $R$ that satisfies $\Sigma$ and has some redundant data value occurrence.

DEFINITION 3. *Let* $R$ *denote a relation schema,* $E \subseteq R$, $\Sigma$ *a set of constraints over* $R$, $A \in E$ *an attribute,* $r$ *a relation over* $R$ *that satisfies* $\Sigma$, *and* $t$ *a tuple in* $r^E$. *An* $E$-replacement *of* $t(A)$ *is a replacement of* $t(A)$ *that is* $E$-*complete. The data value occurrence* $t(A)$ *is* $E$-redundant *for* $\Sigma$ *if and only if for every* $E$-*replacement* $\bar{t}$ *of* $t(A)$, $\bar{r} := (r - \{t\}) \cup \{\bar{t}\}$ *violates some constraint in* $\Sigma$. $R$ *is in* $E$-*Redundancy-Free normal form (*$E$*-RFNF) for* $\Sigma$ *if and only if there are no relation* $r$ *over* $R$ *that satisfies* $\Sigma$, *tuple* $t \in r^E$, *and attribute* $A \in E$, *such that the data value occurrence* $t(A)$ *is* $E$-*redundant for* $\Sigma$.

We illustrate the notion of $E$-redundancy next.

EXAMPLE 4. *Recall that* $R = cdflpz$ *and* $\Sigma = \{p : cfl, p : dz \to c\}$. *The relation in Table 3 shows that* $R$ *is not in cdpz-RFNF for* $\Sigma$*: every cdpz-replacement for either of the bold occurrences would violate the eFD* $p : dz \to c$.

While $E$-RFNF is independent of the type of constraints, we will assume from now on that $\Sigma$ is a set of eUCs and eFDs. As our first result we characterize the $E$-RFNF for $\Sigma$ in terms of the RFNF for the $(E, FD)$-reduct $\Sigma[E]$.

THEOREM 3. *For all sets* $\Sigma$ *over* $R$ *and all* $E \subseteq R$, $R$ *is in* $E$-*RFNF for* $\Sigma$ *if and only if* $R$ *is in RFNF for* $\Sigma[E]$.

For our example the characterization works as follows.

EXAMPLE 5. *For* $R = cdflpz$, $\Sigma = \{p : cfl, p : dz \to c\}$ *and* $E = cdpz$, *we have* $\Sigma[E] = \{dz \to c\}$. *That is,* $R$ *is also not in RFNF for* $\Sigma[E]$.

## 5.2 $E$-BCNF

We now characterize the semantic $E$-RFNF by purely syntactic means. For that purpose, we generalize the BCNF condition to accommodate completeness requirements. Recall that a relation schema $R$ is in BCNF for an FD set $\Sigma$ iff for all $X \to Y \in \Sigma^+_{\mathfrak{A}}$ where $Y \notin X$, $X \to R \in \Sigma^+_{\mathfrak{A}}$. Here, $\mathfrak{A}$ denotes the well-known Armstrong's axioms [3].

DEFINITION 4. *For relation schema* $R$ *and* $E \subseteq R$, $R$ *is in* $E$-BCNF *for a set* $\Sigma$ *over* $R$ *if and only if for every eFD* $E : X \to Y \in \Sigma^+_{\mathfrak{E}}$ *where* $Y \notin X$, $E : X \in \Sigma^+_{\mathfrak{E}}$.

Our running example can be further analyzed as follows.

EXAMPLE 6. *For* $R = cdflpz$ *and* $\Sigma = \{p : cfl, p : dz \to c\}$, $R$ *is not in cdpz-BCNF for* $\Sigma$, *since the eFD* $p : dz \to c \in \Sigma \subseteq \Sigma^+_{\mathfrak{E}}$, $\{c\} \notin \{dz\}$, *but* $dpz : dz \notin \Sigma^+_{\mathfrak{E}}$.

Recall that sets $\Sigma$ and $\Theta$ are $\mathcal{C}$-*covers* of one another if they imply the same constraints in class $\mathcal{C}$. Being in $E$-BCNF for $\Sigma$ is independent of the representation of $\Sigma$. That is, for any cover $\Theta$ of $\Sigma$, $R$ is in $E$-BCNF for $\Sigma$ iff $R$ is in $E$-BCNF for $\Theta$. The $E$-BCNF condition for $\Sigma$ can be characterized by the BCNF condition for $\Sigma[E]$.

THEOREM 4. *Relation schema* $R$ *is in* $E$-*BCNF for the set* $\Sigma$ *if and only if* $R$ *is in BCNF for* $\Sigma[E]$.

Theorem 4 works as follows on our example.

EXAMPLE 7. *For* $R = cdflpz$, $\Sigma = \{p : cfl, p : dz \to c\}$, *and* $E = cdpz$, $R$ *is not in BCNF for* $\Sigma[E] = \{dz \to c\}$.

## 5.3 $E$-RFNF at Application Design Time

We can now characterize the semantic $E$-RFNF by the syntactic $E$-BCNF. Extending the relational case, schemata in $E$-BCNF guarantee at application design time that there will never be an instance that contains any $E$-redundant data value occurrence.

THEOREM 5. *For all relation schemata $R$, all attribute subsets $E \subseteq R$, and all sets $\Sigma$ over $R$, $R$ is in $E$-RFNF for $\Sigma$ if and only if $R$ is in $E$-BCNF for $\Sigma$.*

We can apply the characterization to our example.

EXAMPLE 8. *For $R = cdflpz$ and $\Sigma = \{p : cfl, p : dz \to c\}$, $R$ is in $cflp$-RFNF for $\Sigma$ since $R$ is in BCNF for $\Sigma[E] = \{cfl \to cflp\}$.*

## 5.4 Efficient Testing

Due to the cover-insensitivity of the $E$-BCNF condition, one may wonder about the efficiency of checking whether a given schema is in $E$-BCNF for a given set $\Sigma$. As in the relational case it suffices to check some eFDs in $\Sigma$ instead of checking all eFDs in $\Sigma^+_{\mathfrak{E}}$.

THEOREM 6. *A relation schema $R$ is in $E$-BCNF for $\Sigma$ if and only if for every eFD $E' : X \to Y \in \Sigma$ where $E' \subseteq E$ and $Y \not\subseteq X$, $E : X \in \Sigma^+_{\mathfrak{E}}$. Hence, deciding if a schema is in $E$-BCNF for $\Sigma$ is quadratic in $\Sigma$.*

We apply the simpler characterization to our example.

EXAMPLE 9. *For $R = cdflpz$ and $\Sigma = \{p : cfl, p : dz \to c\}$, $R$ is in $cflp$-BCNF for $\Sigma$ since there is no eFD $E' : X \to Y \in \Sigma$ such that $E' \subseteq E = cflp$.*

## 5.5 $E$-3NF

We now introduce $E$-Third normal form ($E$-3NF) which ensures that all FDs can be enforced locally, without the need of joining relations to check for consistency of updates. Recall the 3NF condition [8]: $R$ is in *3NF* for an FD set $\Sigma$ if for every FD $X \to Y \in \Sigma^+_{\mathfrak{A}}$ where $Y \not\subseteq X$, $X \to R \in \Sigma^+_{\mathfrak{A}}$ or every attribute in $Y - X$ is prime. An attribute $A$ is *prime* if it occurs in some minimal key of $R$ for $\Sigma$. An attribute subset $X$ of $R$ is a *key* of $R$ for $\Sigma$ if $X \to R \in \Sigma^+_{\mathfrak{A}}$. A key $X$ of $R$ is *minimal* for $\Sigma$ if every proper subset $Y \subset X$ is not a key of $R$ for $\Sigma$. We extend these concepts to handle data completeness requirements. For $E \subseteq R$ and an eUC/eFD set $\Sigma$, an eUC $E : K \in \Sigma^+_{\mathfrak{E}}$ is $E$-minimal for $\Sigma$ if and only if there is no $E$-key $E : K' \in \Sigma^+_{\mathfrak{E}}$ for $\Sigma$ such that $K' \subset K$. An attribute $A$ is *$E$-prime* for $\Sigma$ if and only if $A \in K$ for some $E$-minimal key $E' : K \in \Sigma^+_{\mathfrak{E}}$.

DEFINITION 5. *A relation schema $R$ is in $E$-3NF for $\Sigma$ if and only if for every non-trivial eFD $E : X \to Y \in \Sigma^+_{\mathfrak{E}}$ with $E' \subseteq E$, $E : X \in \Sigma^+_{\mathfrak{E}}$ or every attribute in $Y - X$ is $E$-prime.*

We can check this condition on our running example.

EXAMPLE 10. *For $R = cdflpz$ and $\Sigma = \{p : cfl, p : dz \to c\}$, we have seen that $c : dflpz$ is an $R$-minimal key for $\Sigma$, the other $R$-minimal key being $dpz : cfl$. That is, every attribute in $R$ is $R$-prime. Hence, $R$ is in $R$-3NF. However, $R$ is not in cdpz-3NF as eFD $p : dz \to c \in \Sigma$, $c \notin \{d, z\}$, $cp : dz \notin \Sigma^+_{\mathfrak{E}}$, and $c$ is not cdpz-prime.*

Similar to $E$-BCNF and BCNF, we can check that $R$ is in $E$-3NF for $\Sigma$ by testing that $R$ is in 3NF for $\Sigma[E]$.

THEOREM 7. *For all relation schemata $R$, all $E \subseteq R$, and all sets $\Sigma$ over $R$, $R$ is in $E$-3NF for $\Sigma$ if and only if $R$ is in 3NF for $\Sigma[E]$.*

EXAMPLE 11. *In our running example $R = cdflpz$, $\Sigma = \{p : cfl, p : dz \to c\}$ and $E = R$, $\Sigma[E] = \{cfl \to dpz, dz \to c\}$. The two minimal keys are $cfl$ and $dflpz$. As every attribute is prime, $R$ is in 3NF for $\Sigma[E]$.*

Finally, $E$-3NF can be validated by checking the relevant conditions for just the input $\Sigma$, rather than its closure $\Sigma^+_{\mathfrak{E}}$.

THEOREM 8. *$R$ is in $E$-3NF for a set $\Sigma$ of eUCs and eFDs over $R$ if and only if for every eFD $E' : X \to Y \in \Sigma$ where $E' \subseteq E$ and $Y \not\subseteq X$, $E : X \in \Sigma^+_{\mathfrak{E}}$ or every attribute in $Y - X$ is $E$-prime.*

This translates to our running example as follows.

EXAMPLE 12. *For $R = cdflpz$, $\Sigma = \{p : cfl, p : dz \to c\}$, $E = R$ and $p : dz \to c \in \Sigma$, $c \in R$ is $E$-prime. By Theorem 8 this suffices to establish that $R$ is in $E$-3NF for $\Sigma$.*

## 5.6 Hardness of Normal Form Criteria

As relational normalization is the special case where $E = R$, checking normal form criteria is hard in general.

THEOREM 9. *Deciding whether a sub-schema of a given schema is in $E$-BCNF for a given set $\Sigma$ is* coNP*-complete. Deciding whether a given schema is in $E$-3NF for a given set $\Sigma$ is* NP*-complete.*

# 6. TAILORING NORMALIZATION

We now establish algorithms to design relational database schemata that are tailored to the completeness requirements of applications. For that purpose, we normalize a given schema $R$ for the given set $\Sigma$ of eUCs and eFDs. The completeness requirements are consolidated in an attribute subset $E \subseteq R$, expressing that the application only handles $E$-complete records. The choice of $E$ determines the set $\Sigma[E]$ of traditional FDs that are used to normalize $R$. For each $E$ we pursue i) lossless BCNF decompositions that are redundancy-free but potentially not dependency-preserving, and ii) lossless 3NF syntheses that are dependency-preserving but potentially not redundancy-free.

## 6.1 $E$-BCNF Decomposition

We recall terminology from relational databases. A *decomposition* of relation schema $R$ is a set $\mathcal{D} = \{R_1, \ldots, R_n\}$ of relation schemata such that $R_1 \cup \cdots \cup R_n = R$. For $R_j \subseteq R$ and FD set $\Sigma$ over $R$, $\Sigma_{R_j} = \{X \to Y \mid X \to Y \in \Sigma^+_{\mathfrak{A}} \text{ and } X, Y \subseteq R_j\}$ denotes the *projection* of $\Sigma$ onto $R_j$. A decomposition $\mathcal{D}$ of $R$ with FD set $\Sigma$ is *lossless* if every relation $r$ over $R$ that satisfies $\Sigma$ is the join of its projections on the elements of $\mathcal{D}$, that is, $r = \bowtie_{R_j \in \mathcal{D}} r[R_j]$. Here, $r[R_j] = \{t(R_j) \mid t \in r\}$. A *BCNF* decomposition of $R$ with FD set $\Sigma$ is a decomposition $\mathcal{D}$ of $R$ where every $R_j \in \mathcal{D}$ is in BCNF for $\Sigma_{R_j}$. Theorem 4 motivates a definition of an $E$-lossless BCNF decomposition.

DEFINITION 6. *An* $E$-lossless BCNF decomposition *of a schema $R$ for a set $\Sigma$ of eUCs and eFDs over $R$ is a lossless BCNF decomposition of $R$ for $\Sigma[E]$.*

Instrumental to Definition 6 is the following decomposition theorem. It covers the classical decomposition theorem [32] as the special case where $E = R$. Data completeness-tailored normalization does not loose any records by following a hybrid decomposition approach. Given $E$, a relation

is decomposed horizontally into its application-relevant part $r^E$ of $E$-complete records, and its application-irrelevant part $r - r^E$ of records with missing data on $E$. Classical vertical decomposition can then be applied to $r^E$.

THEOREM 10. *Let $E : X \to Y$ be an eFD that satisfies the relation $r$ over relation schema $R$. Then the set of $E$-complete records of $r$ is the lossless join of its projections on $XY$ and $X(R-Y)$, that is, $r^E = r^E[XY] \bowtie r^E[X(R-Y)]$. Also, $r$ is the disjoint union of the set of $E$-complete records of $r$, and the set of records of $r$ with missing data on some column in $E$, that is, $r = r^E \cup (r - r^E)$.*

Hence, an $E$-lossless BCNF decomposition for a set $\Sigma$ of eUCs and eFDs can simply be obtained by a classical lossless BCNF decomposition for the $(E, FD)$-reduct $\Sigma[E]$ of $\Sigma$.

| PROBLEM: $E$-BCNF Decomposition | |
|---|---|
| INPUT: | Relation Schema $R$<br>Set $\Sigma$ of eUCs and eFDs over $R$<br>Attribute subset $E \subseteq R$ |
| OUTPUT: | $E$-lossless BCNF decomposition of $R$ for $\Sigma$ |
| METHOD: | Perform a lossless BCNF decomposition of $R$ for $\Sigma[E]$ |

We illustrate the decomposition on our running example.

EXAMPLE 13. *In our running example $R = cdflpz$, $\Sigma = \{p : cfl, p : dz \to c\}$, and $E = R$, $R$ is not in $E$-BCNF for $\Sigma$. In fact, $R$ is not in BCNF for $\Sigma[E] = \{cfl \to dpz, dz \to c\}$. A BCNF decomposition yields $R_1 = cdz$ with $\Sigma_1 = \{dz \to c\}$ and $R_2 = dflpz$ with $\Sigma_2 = \emptyset$. For the relation $r$ from Table 3, the projection of $r^E$ on the decomposed schema is as follows, except for the last row in both tables because tuple $t_{15}$ has a null marker occurrences on column* f_name.

| c(ity) | z(ip) | d(ate_register) |
|---|---|---|
| *burlington* | *27217* | *05/11/1940* |
| *jackson* | *27845* | *06/15/1936* |
| *lasker* | *27845* | *05/10/1940* |
| *wilmington* | *28405* | *01/01/1930* |
| *rose hill* | *28458* | *01/01/1930* |
| *new bern* | *28562* | *03/06/1935* |
| *chinquapin* | *28521* | *01/01/1936* |
| *burlington* | *27215* | *05/06/1940* |
| *mebane* | *27302* | *4/05/1940* |
| *mebane* | *27302* | *05/06/1940* |
| *kinston* | *28504* | *05/02/1940* |
| *kinston* | *28504* | *05/04/1940* |

| f(_name) | l(_name) | z(ip) | p(hone) | d(ate_register) |
|---|---|---|---|---|
| *ida* | *cheek* | *27217* | *226 4848* | *05/11/1940* |
| *effie* | *massey* | *27217* | *336 226 8544* | *05/11/1940* |
| *peggy* | *floyd* | *27845* | *252 536 2668* | *06/15/1936* |
| *essie* | *warren* | *27845* | *252 539 2866* | *05/10/1940* |
| *rodney* | *glockner* | *28405* | *910 509 3864* | *01/01/1930* |
| *sallie* | *blanchard* | *28458* | *910 289 3320* | *01/01/1930* |
| *joseph* | *cox* | *28562* | *252 288 4763* | *03/06/1935* |
| *james* | *smith* | *28521* | *910 285 3493* | *01/01/1936* |
| *james* | *smith* | *27215* | *584 4202* | *05/06/1940* |
| *dorothy* | *faucette* | *27302* | *919 563 1285* | *4/05/1940* |
| *dorothy* | *allred* | *27302* | *563 1426* | *05/06/1940* |
| *eloise* | *croom* | *28504* | *252 569 4436* | *05/02/1940* |
| ⊥ | *croom* | *28504* | *252 569 9516* | *05/04/1940* |

*All $E$-redundant data value occurrences from $r$ have been eliminated. However, the eUC $p : cfl$ was not preserved.*

Recall that a decomposition $\mathcal{D}$ of schema $R$ with FD set $\Sigma$ is *dependency-preserving* whenever $\Sigma^+_{\mathfrak{A}} = (\cup_{R_j \in \mathcal{D}} \Sigma[R_j])^+_{\mathfrak{A}}$.

DEFINITION 7. *An $E$-*dependency-preserving *decomposition of a schema $R$ for the eUC/eFD set $\Sigma$ is a dependency-preserving decomposition of $R$ for $\Sigma[E]$.*

## 6.2 $E$-3NF Synthesis

3NF synthesis guarantees dependency-preservation, but may exhibit data value redundancy caused by FDs. It was shown recently that 3NF exhibits minimal levels of data redundancy when achieving dependency-preservation [2, 22]. Hence, we will equip our new framework with an appropriate 3NF synthesis strategy. Recall that a 3NF decomposition of a relation schema $R$ for an FD set $\Sigma$ is a decomposition $\mathcal{D}$ of $R$ where every $R_j \in \mathcal{D}$ is in 3NF for $\Sigma_{R_j}$. Theorem 7 motivates the following definition.

DEFINITION 8. *An $E$-*dependency-preserving, $E$-lossless 3NF decomposition *of a schema $R$ for the set $\Sigma$ of eUCs and eFDs is a dependency-preserving, lossless 3NF decomposition of $R$ for $\Sigma[E]$.*

Following Theorem 10, an $E$-dependency-preserving, $E$-lossless 3NF synthesis for a set $\Sigma$ of eUCs and eFDs can simply be obtained by a classical dependency-preserving lossless 3NF synthesis for the $(E, FD)$-reduct $\Sigma[E]$ of $\Sigma$.

| PROBLEM: $E$-3NF Synthesis | |
|---|---|
| INPUT: | Relation schema $R$<br>Set $\Sigma$ of eUCs and eFDs over $R$<br>Attribute subset $E \subseteq R$ |
| OUTPUT: | $E$-dependency-preserving, $E$-lossless 3NF decomposition of $R$ wrt $\Sigma$ |
| METHOD: | Perform a dependency-preserving, lossless 3NF synthesis of $R$ for $\Sigma[E]$ |

We illustrate the synthesis on our running example.

EXAMPLE 14. *In our running example $R = cdflpz$, $\Sigma = \{p : cfl, p : dz \to c\}$, $R$ is indeed in $R$-3NF for $\Sigma$. For $E = cdpz$, however, $R$ is not in $E$-3NF for $\Sigma$. In fact, $R$ is not in 3NF for $\Sigma[E] = \{dz \to c\}$. A 3NF synthesis yields $R_1 = cdz$ with $\Sigma_1 = \{dz \to c\}$ and to ensure $E$-losslessness we add the $E$-minimal key $R_2 = dflpz$ with $\Sigma_2 = \emptyset$. For the relation $r$ from Table 3, the projection of $r^E$ onto the decomposed schema is the same as in Example 13 but inclusive of the last row in both tables because of the looser data-completeness requirements.*

**Summary.** We have tailored the entire relational schema design framework to data-completeness requirements of applications. We allow data stewards to declare these requirements as an extension to the familiar concept of a functional dependency. The results show that extensions of the familiar BCNF (3NF) normal form achieve an elimination (minimization across dependency-preserving decompositions) of data values that may occur redundantly in records that meet the completeness requirements. As an optimal result for database practice, schemata can be automatically transformed into these normal forms by applying relational normalization algorithms to a set of relational FDs that emerge from the set of extended FDs and the data-completeness requirements. The next section illustrates what our framework achieves when applied to real-world schemata and data.

# 7. EXPERIMENTS

We report on experiments with our framework using real-world benchmark schemata and data, available for download at `https://bit.ly/2AoOis6`. We provide insight on how

**Table 6: Redundant data value occurrences in benchmarks, and the time in seconds to compute all of them**

| *data set* | *#complete* | *#red* | *%red* | *time (s)* |
|---|---|---|---|---|
| horse | 6,795 | 4,775 | 70.27 | 8.075 |
| bridges | 1,327 | 411 | 30.97 | 0.002 |
| hepatitis | 2,933 | 1,695 | 57.79 | 0.179 |
| breast | 7,585 | 712 | 9.39 | 0.005 |
| echo | 1,584 | 489 | 30.87 | 0.002 |
| plista | 39,431 | 28,827 | 73.11 | 18.415 |
| flight | 57,062 | 48,414 | 84.84 | 86.585 |
| ncvoter | 16,137 | 3,170 | 19.64 | 0.047 |
| china | 4,313,980 | 2,131,677 | 49.41 | 412.867 |
| uniprot | 11,600,704 | 1,413,038 | 12.18 | 1,777.245 |
| diabetic | 1,017,738 | 543,935 | 53.45 | 3,273.183 |

many $E$-redundant data values occur in the data, rank the relevance of our eFDs by how many data redundancies they cause, show how often schemata satisfy a normal form condition, how much redundancy $E$-3NF permits, how many dependencies $E$-BCNF preserves, and how large decomposed schemata become. We consider the times of computations, and suggest how data stewards can use our ranking of eFDs, using the example of our two applications from the introduction. All our experiments are run on an Intel Xeon 3.6 GHz, 256GB RAM, Windows 10 Dell workstation.

## 7.1 $E$-redundancy and eFD Ranking

Table 6 lists for each incomplete benchmark data set the number *#complete* of data occurrences that are complete, the number *#red* of those that are redundant, the percentage *%red* of redundant data value occurrences in the data set, and the *time* in seconds to compute all the redundant occurrences given the data set and given the canonical cover of the eUCs and eFDs that hold on the data set. The canonical covers can be computed by some algorithms that will be discussed in a separate article. The sheer number and percentage of redundant occurrences clearly motivates our research, and the time taken to compute them shows that this insightful analysis is efficient on even large data sets with large numbers of constraints. Of course, if a team of domain experts selects the meaningful eUCs and eFDs for an application, then the redundant occurrences will likely be fewer and can be computed more efficiently.

Guiding data stewards in their selection of meaningful eFDs from the canonical cover, we can rank the relevance of an eFD by the number of redundant data value occurrences it causes. Figure 1 shows the number of eFDs in a canonical cover that cause not more than a given number of redundant data value occurrences in the data set. The labels on the x-axis indicate the maximum values for 0, 2.5, 5, 10, 15, 20, 40, 60, 80, and 100 percent of the maximum redundant occurrences any eFD causes. The figure illustrates clearly that most eFDs cause few data redundancies, which makes it possible for data stewards to focus their attention to select few eFDs of higher rank.

## 7.2 Pure eFDs

As indicated in our introduction, pure eFDs occur frequently in real-world data, cause many redundant data value occurrences, and also occur frequently among those eFDs that cause most redundant data value occurrences. This is illustrated on our benchmark data sets in Table 7. Here,

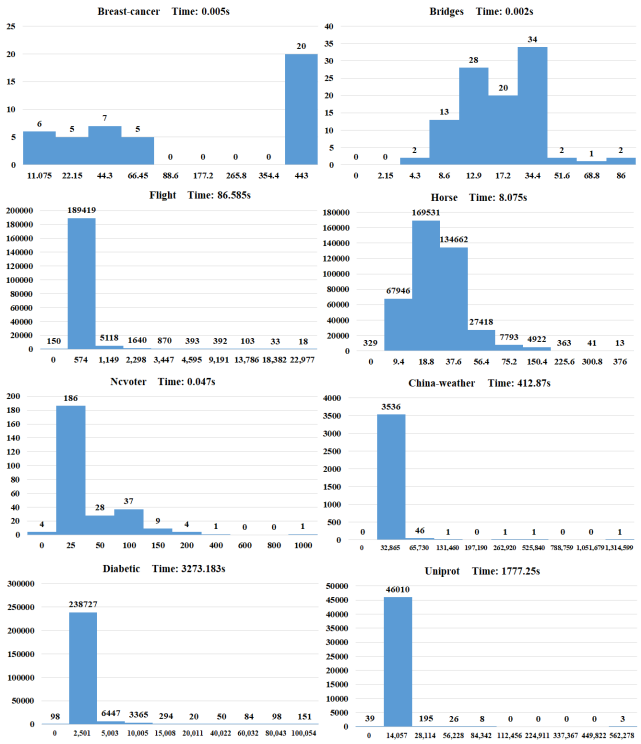


**Figure 1: Numbers of eFDs in canonical covers (y-axis) that cause not more than the given number of redundant occurrences (x-axis)**

we list the percentage of pure eFDs among all eFDs in the canonical covers we computed, the average loss of redundant data value occurrences in percent when transforming a pure eFD $E : X \to Y$ into a non-pure eFD $\varnothing : E - Y \to Y$, and the percentage of pure eFDs among those eFDs that ranked within the top-10% according to the number of redundant data values they cause.

## 7.3 Quality of Decompositions

For the following experiments we created inputs as outlined next. For each fixed size $|E|$ of the completeness requirements, we created different sets of eUCs and eFDs by picking up to 1,000 unique attribute sets $E$ of the fixed size, and then selecting all eUCs and eFDs that hold on the data

**Table 7: Statistics on pure eFDs in benchmark data**

| data set | %pure | red loss | %pure top-10% |
|---|---|---|---|
| breast | 0 | 0 | 0 |
| bridges | 29.58 | 62.85 | 69.23 |
| china | 18.83 | 81.32 | 60.08 |
| diabetic | 54.02 | 57.79 | 64.68 |
| echo | 20.32 | 65.24 | 46.67 |
| flight | 17.73 | 44.50 | 16.46 |
| hepatitis | 53.21 | 77.56 | 67.62 |
| horse | 94.25 | 20.60 | 94.38 |
| ncvoter | 16.35 | 66.88 | 50.00 |
| plista | 74.18 | 8.14 | 69.92 |
| uniprot_512k_30c | 50.73 | 63.50 | 92.55 |
| ncvoter128k | 34.00 | 45.80 | 45.89 |
| ncvoter256k | 39.88 | 50.14 | 58.86 |
| ncvoter512k | 40.30 | 46.57 | 44.13 |
| ncvoter1024k | 39.46 | 49.20 | 48.54 |

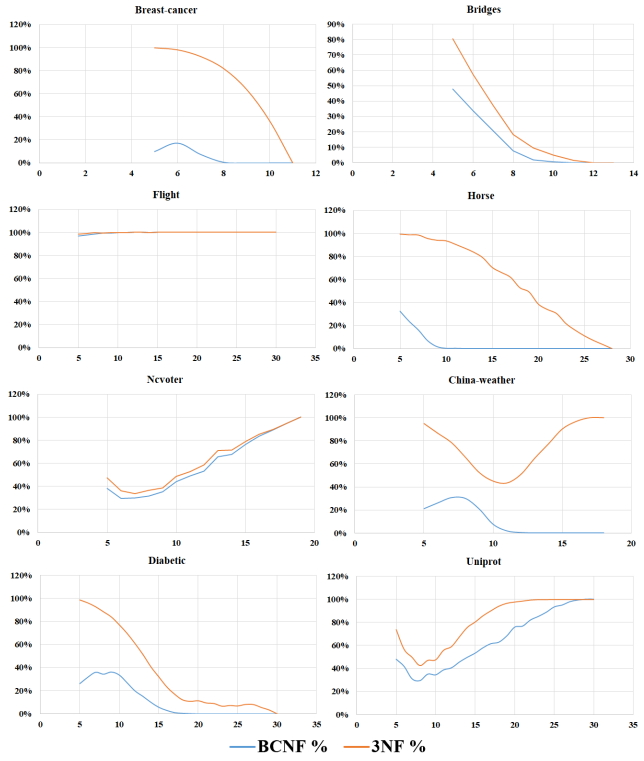


**Figure 2: Average percentage of schemata in $E$-3NF and $E$-BCNF, respectively, by given size of $E$**

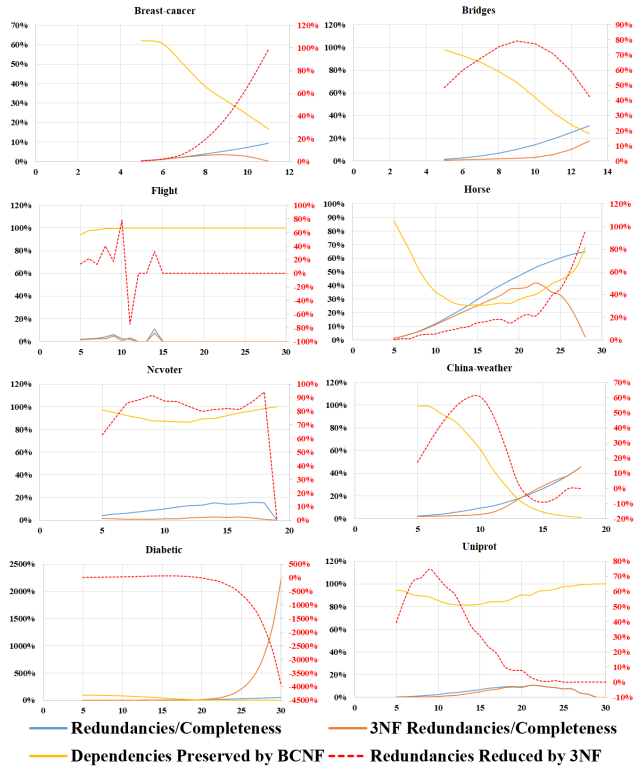


**Figure 3: Elimination of $E$-redundancy by $E$-3NF (vertical LHS), and $E$-preservation by $E$-BCNF (vertical RHS)**

set and whose embeddings are subsets of $E$. Figure 2 shows for each data set and each size of $E$, the percentage of all input sets that are in $E$-BCNF and $E$-3NF, respectively.

Figure 3 shows average percentages of i) the $E$-complete data value occurrences that are redundant (blue line), ii) those after $E$-3NF decomposition (orange line) and iii) eliminated redundancies after $E$-3NF synthesis (yellow line) all plotted against the LHS vertical axis, and iv) dependencies preserved during $E$-BCNF decomposition (red dotted line) plotted against the RHS vertical axis. In general, there is no control about the number of $E$-redundant data values that an $E$-3NF decomposition must tolerate to preserve all relevant dependencies. Vice versa, there is no control on how many relevant dependencies will be lost to eliminate all $E$-redundant data values during $E$-BCNF decomposition. For instance, $E$-3NF may duplicate non-trivial eFDs across schemata, causing a blow-up of the $E$-redundancies (orange above blue line), see *diabetic* and *china_weather*.

## 7.4 Size and time of decompositions

Figure 4 illustrates the impact of the size of $E$ on the cardinality of the decompositions, that is, their total number of attributes (LHS y-axis) and the computation time in seconds (RHS y-axis). Boldly speaking, the larger the decompositions the more updates (less redundancy) and the less queries (more joins required) will be efficient.

## 7.5 Qualitative analysis

For qualitative insight, we consider two applications for the data set *ncvoter* with 19 columns and 1000 records. The first application has attribute set $E_1$ with *full_phone_num*,


**Figure 4: Average cardinality of decompositions and time (s) taken to compute them**

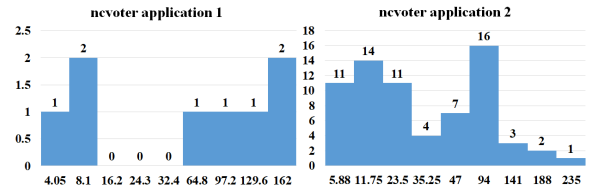


**Figure 5: Applications for *ncvoter* data set**

| Embedding E-XY | LHS X | RHS Y | #red |
|---|---|---|---|
| [] | ['last_name', 'zip_code'] | ['city'] | 158 |
| ['full_phone_num'] | ['zip_code', 'register_date'] | ['city'] | 121 |
| [] | ['street_address'] | ['city'] | 81 |
| [] | ['first_name', 'zip_code'] | ['city'] | 71 |
| [] | ['full_phone_num'] | ['city'] | 12 |
| [] | ['street_address'] | ['zip_code'] | 81 |
| [] | ['full_phone_num'] | ['zip_code'] | 12 |
| ['full_phone_num'] | ['last_name', 'city', 'register_date'] | ['zip_code'] | 8 |

**Figure 6: Ranking of some eFDs for application $E_1$**

*street_address*, *register_date*, *last_name*, *first_name*, *city*, and *zip_code*, while the second application uses the attribute set $E_2 \supseteq E_1$ plus *birthplace*, *ethnic*, *race*, *gender*, and *age*. From the canonical cover we then selected only those eFDs $E : X \to Y$ where $E$ contained $E_1$ or $E_2$, respectively.

Our rankings help identify eFDs relevant for normalization and pinpointing dirty data. Figure 5 shows the distribution of eFDs in percentiles of the redundant values they cause based on $E_1$ and $E_2$, respectively. For growing $|E|$ typically more eFDs need consideration. Here, our ranking offers a convenient measure of relevance for data stewards.

A view that might be particularly useful for data stewards is to fix a column in $E$, and list the minimal LHSs that functionally determine that column, ranked by the relevance of the corresponding eFD. This is illustrated in Figures 6 and 7, where we list all minimal LHSs for the columns *city* and *zip_code* based on the two completeness requirements $E_1$ and $E_2$, respectively.

Finally, a data steward can view the records in which the redundant data values actually occur. This helps them decide if the eFD is meaningful for the application, holds just accidentally, or identify records with dirty data. Figure 8 shows some records with $E_1$-redundant data values.

An inspection of these records reveals some dirty data: i) Hazel and Homer Hargis live at the same street address, and

| Embedding E-XY | LHS X | RHS Y | #red |
|---|---|---|---|
| ['full_phone_num', 'birth_place'] | ['zip_code'] | ['city'] | 235 |
| [] | ['last_name', 'zip_code'] | ['city'] | 158 |
| ['full_phone_num'] | ['zip_code', 'register_date'] | ['city'] | 121 |
| ['full_phone_num'] | ['age', 'zip_code'] | ['city'] | 106 |
| [] | ['street_address'] | ['city'] | 81 |
| [] | ['age', 'gender', 'zip_code', 'register_date'] | ['city'] | 73 |
| [] | ['first_name', 'zip_code'] | ['city'] | 71 |
| [] | ['full_phone_num'] | ['city'] | 12 |
| [] | ['first_name', 'last_name', 'birth_place', 'register_date'] | ['city'] | 6 |
| [] | ['first_name', 'last_name', 'age'] | ['city'] | 2 |
| [] | ['street_address'] | ['zip_code'] | 81 |
| [] | ['last_name', 'age', 'ethnic', 'city'] | ['zip_code'] | 22 |
| [] | ['full_phone_num'] | ['zip_code'] | 12 |
| ['full_phone_num'] | ['last_name', 'city', 'register_date'] | ['zip_code'] | 8 |
| [] | ['last_name', 'age', 'city', 'register_date'] | ['zip_code'] | 6 |
| ['full_phone_num'] | ['last_name', 'age', 'city'] | ['zip_code'] | 4 |
| [] | ['first_name', 'last_name', 'age'] | ['zip_code'] | 2 |
| [] | ['last_name', 'age', 'gender', 'city'] | ['zip_code'] | 2 |

**Figure 7: Ranking of some eFDs for application $E_2$**

| first_name | last_name | street_address | city | zip_code | full_phone_num | register_date |
|---|---|---|---|---|---|---|
| homer | hargis | 3962 bellemont-mt hermon rd | burlington | 27215 | 226 9906 | 10/19/1940 |
| hazel | hargis | 3962 bellemont-mt hermon rd | burlington | 27215 | 336 226 9906 | 10/19/1940 |
| sallie | futrell | 9802 us hwy 258 | murfreesboro | 27855 | 252 398 3716 | 10/21/1938 |
| herbert | futrell | 9802 us hwy 258 | murfreesboro | 27855 | 252 398 3716 | 10/21/1938 |
| vivian | etheridge | 1 pout house ln | shawboro | 27973 | 252 232 2597 | 04/30/1932 |
| john | etheridge | 0 pout house ln | shawboro | 27973 | 252 232 2597 | 04/30/1932 |
| mattie | horne | 1528 lower white store rd | polkton | 28135 | 704 272 8433 | 5/11/1940 |
| william | horne | 1528 lower white store rd | polkton | 28135 | 704 272 8433 | 5/11/1940 |

**Figure 8: $E_1$-redundancies caused by the eFD *full_phone_num* : *last_name,city,register_date* → *zip_code***

their phone numbers are different, and ii) Vivian and John Etheridge share the same phone number, but their street address is different. For i) the inconsistency can easily be resolved by giving the full phone number, while for ii) it is more likely that Vivian indicated the correct street number.

**Summary.** Our experiments illustrate on benchmark schemata and data that eFDs provide effective declarative means to capture and reason about redundant data value occurrences that are fit for application requirements. Our ranking guides data stewards in their selection of eFDs that are relevant for normalization purposes given the application requirements. Our normalization strategies result in a wide spectrum of schemata with clear achievements in terms of the elimination of pertinent data redundancies or the preservation of pertinent dependencies, accommodating tradeoffs between update and query efficiency. More experiments on perfect decompositions and an analysis of our experiments on horizontal fragments on *ncvoter* are also available [38].

## 8. CONCLUSION AND FUTURE WORK

Schema design for data with missing values has been an open problem since the 1980s. Previous work has focused on finding suitable extensions of functional dependencies to accommodate different interpretations of null markers. In contrast, we introduced the class of embedded functional dependencies (eFDs) that is only dependant on complete data, can express completeness and integrity requirements of applications, and captures many redundant data values. This has enabled us to establish a fully-fledged normalization framework that is robust under different interpretations of null markers, tailors relational schema design to data-completeness requirements, and lifts the achievements of classical BCNF and 3NF to applications with missing data. Extensive experiments on real-world benchmark schemata and data exemplify the effectiveness of our framework, the efficiency of our algorithms, and the achievements of our new normal form proposals. In particular, we illustrated the impact of the completeness requirements on trade-offs between data redundancy elimination and dependency-preservation. Next steps include an in-depth investigation into the discovery problem of embedded uniqueness constraints and functional dependencies from given relations. The discovery is important for data profiling and its applications [1, 39], but can also help identify business rules [37]. For that purpose, the ranking of eFDs can provide effective guidance. Furthermore, an extension of our framework to other data quality dimensions and other classes of data dependencies is important [27, 28, 33]. The ability to declare data accuracy or data timeliness requirements as part of functional, multivalued and inclusion dependencies [13, 20, 26] would lift important data quality dimensions to first-class citizens that impact schema design considerations based on rich sources of data redundancy.

# 9. REFERENCES


[1] Z. Abedjan, L. Golab, F. Naumann, and T. Papenbrock. *Data Profiling*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2018.

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

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

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

[5] P. A. Bernstein. Synthesizing third normal form relations from functional dependencies. *ACM Trans. Database Syst.*, 1(4):277–298, 1976.

[6] J. Biskup. Achievements of relational database schema design theory revisited. In *Semantics in Databases*, pages 29–54, 1995.

[7] J. Biskup. Achievements of relational database schema design theory revisited. In L. Libkin and B. Thalheim, editors, *Semantics in Databases*, volume 1358 of *Lecture Notes in Computer Science*, pages 29–54. Springer, 1998.

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

[9] E. F. Codd. Further normalization of the database relational model. In *Courant Computer Science Symposia 6: Data Base Systems*, pages 33–64, 1972.

[10] C. J. Date and R. Fagin. Simple conditions for guaranteeing higher normal forms in relational databases. *ACM Trans. Database Syst.*, 17(3):465–476, 1992.

[11] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *J. Log. Program.*, 1(3):267–284, 1984.

[12] R. Fagin. The decomposition versus synthetic approach to relational database design. In *VLDB 1977*, pages 441–446, 1977.

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

[14] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst.*, 33(2), 2008.

[15] S. Greco, C. Molinaro, and F. Spezzano. *Incomplete Data and Data Dependencies in Relational Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.

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

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

[18] N. D. Jones and W. T. Laaser. Complete problems for deterministic polynomial time. *Theor. Comput. Sci.*, 3(1):105–117, 1976.

[19] H. Köhler and S. Link. SQL schema design: Foundations, normal forms, and normalization. In *SIGMOD*, pages 267–279, 2016.

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

[21] H. Köhler and S. Link. SQL schema design: Foundations, normal forms, and normalization. *Inf. Syst.*, 76:88–113, 2018.

[22] S. Kolahi. Dependency-preserving normalization of relational and XML data. *J. Comput. Syst. Sci.*, 73(4):636–647, 2007.

[23] S. Kolahi and L. Libkin. An information-theoretic analysis of worst-case redundancy in database design. *ACM Trans. Database Syst.*, 35(1):5:1–5:32, 2010.

[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. *A guided tour of relational databases and beyond*. Springer, 1999.

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

[27] S. Link and H. Prade. Relational database schema design for uncertain data. In *CIKM*, pages 1211–1220, 2016.

[28] S. Link and H. Prade. Relational database schema design for uncertain data. *Inf. Syst.*, 84:88 – 110, 2019.

[29] D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1983.

[30] T. Papenbrock and F. Naumann. Data-driven schema normalization. In *EDBT*, pages 342–353, 2017.

[31] R. Ramakrishnan and J. Gehrke. *Database management systems*. McGraw-Hill, 2003.

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

[33] S. W. Sadiq, T. Dasu, X. L. Dong, J. Freire, I. F. Ilyas, S. Link, R. J. Miller, F. Naumann, X. Zhou, and D. Srivastava. Data quality: The role of empiricism. *SIGMOD Record*, 46(4):35–43, 2017.

[34] M. W. Vincent. A corrected 5NF definition for relational database design. *Theor. Comput. Sci.*, 185(2):379–391, 1997.

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

[36] Z. Wei and S. Link. Embedded cardinality constraints. In *CAiSE*, pages 523–538, 2018.

[37] Z. Wei and S. Link. DataProf: Semantic profiling for iterative data cleansing and business rule acquisition. In *SIGMOD*, pages 1793–1796, 2018.

[38] Z. Wei and S. Link. Data-completeness tailored database design. Technical Report CDMTCS-537, The University of Auckland, 2019.

[39] Z. Wei and S. Link. Discovery and ranking of functional dependencies. In *ICDE*, pages 1526–1537, 2019.

[40] Z. Wei, S. Link, and J. Liu. Contextual keys. In *ER*, pages 266–279, 2017.

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