CDMTCS
Research
Report
Series

Logical Schema Design that Quantifies Update Inefficiency and Join Efficiency

Sebastian Link
The University of Auckland

Ziheng Wei
The University of Auckland

CDMTCS-553
March 2021

Centre for Discrete Mathematics and
Theoretical Computer Science

Logical Schema Design that Quantifies Update Inefficiency and Join Efficiency

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

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

March 31, 2021

Abstract

The goal of classical normalization is to maintain data consistency under updates, with a minimum level of effort. Given functional dependencies (FDs) alone, this goal is only achievable in the special case an FD-preserving Boyce-Codd Normal Form (BCNF) decomposition exists. As we show, in all other cases the level of effort can be neither controlled nor quantified. In response, we establish the ℓ-Bounded Cardinality Normal Form, parameterized by a positive integer ℓ. For every ℓ, the normal form condition requires from every instance that every value combination over the left-hand side of every non-trivial FD does not occur in more than ℓ tuples. BCNF is captured when $\ell = 1$. We demonstrate that schemata in this normal form characterize the instances that are i) free from level ℓ data redundancy and update inefficiency, and ii) permit level ℓ join efficiency. We establish algorithms that compute schemata in ℓ-Bounded Cardinality Normal Form for the smallest level ℓ attainable across all FD-preserving decompositions. Additional algorithms i) attain even smaller levels of effort based on the loss of some FDs, and ii) decompose schemata based on prioritized FDs that cause high levels of effort. Our framework informs de-normalization already during logical design. In particular, level ℓ quantifies both the incremental maintenance and join support of materialized views. Experiments with synthetic and real-world data illustrate which properties the schemata have that result from our algorithms, and how these properties predict the performance of update and query operations on instances over the schemata, without and with materialized views.

Keywords: Cardinality constraint; Data redundancy; Functional dependency; Join; Normal form; Normalization; Update

Table 1: Schema HAP in 3NF with instance r over HAP

HAP

Event	*Venue*	*Company*	*Time*
Party	v_1	**Kilo**	t_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Party	v_{ℓ_1}	**Kilo**	t_{ℓ_1}
e_1	Dome	**Mega**	t'_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
e_{ℓ_2}	Dome	**Mega**	t'_{ℓ_2}

Table 2: Design $\mathcal{D}_1 = \{R_2, R_3, R_4\}$ of HAP with instance $\{r_i = \pi_{R_i}(r)\}_{i=2}^4$

R_2

Venue	*Company*
v_1	Kilo
$\vdots$	$\vdots$
v_{ℓ_1}	Kilo
Dome	Mega

R_3

Event	*Venue*	*Time*
Party	v_1	t_1
$\vdots$	$\vdots$	$\vdots$
Party	v_{ℓ_1}	t_{ℓ_1}
e_1	Dome	t'_1
$\vdots$	$\vdots$	$\vdots$
e_{ℓ_2}	Dome	t'_{ℓ_2}

R_4

Event	*Company*	*Time*
Party	**Kilo**	t_1
$\vdots$	$\vdots$	$\vdots$
Party	**Kilo**	t_{ℓ_1}
e_1	Mega	t'_1
$\vdots$	$\vdots$	$\vdots$
e_{ℓ_2}	Mega	t'_{ℓ_2}

1 Introduction

Schema design aims at finding a layout of the data that facilitates the efficient processing of common queries and updates. The problem is challenging as data redundancy typically causes update inefficiency but promotes join efficiency. So far, the challenge has been addressed by performing normalization during logical schema design to achieve update efficiency, followed by de-normalization during physical design to boost query efficiency. In particular, de-normalization is done only after the database is operational and patterns of data access on normalized databases emerge.

The goal of classical normalization is to maintain data consistency under updates, with a minimum level of effort. Normalization aims at eliminating any occurrence of redundant data values in any future database instance. This is attempted by structurally transforming functional dependencies (FDs), that cause data redundancy, into keys, that prohibit data redundancy. The well-known Boyce-Codd Normal Form (BCNF) requires the left-hand side of every non-trivial FD to be a key. Hence, data redundancy can never be caused by FDs transformed into keys. However, while every schema can be decomposed into BCNF, FDs may be lost during this process and still cause data redundancy [5]. A more liberal condition is given by the Third Normal Form (3NF) where the left-hand side of every non-trivial FD must be a key or every attribute on the right-hand side must be prime (that is, to be part of some minimal key). 3NF synthesis can transform every schema into 3NF without losing any FD, but some FDs still cause data redundancy as they were not transformed into keys. Hence, classical normalization can only measure its success when no effort is required at all to maintain data consistency. This is only possible when an FD-preserving BCNF decomposition exists. In all other cases, the level of effort can be neither controlled nor even measured.

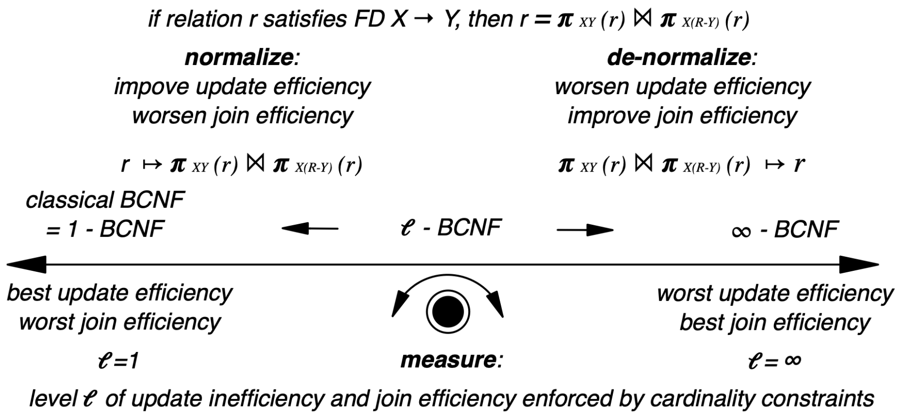
As running example consider the event management schema HAP with each record representing an *E(vent)* at a *V(enue)* and *T(ime)* with some *C(ompany)* in charge. HAP uses FDs to model business rules. The FD $E \to C$ says that only one company is in charge of every event, $V \to C$ says there cannot be different companies in charge at the same venue, $VT \to E$ says that no different events happen at the same venue at the same time, $ET \to V$ says that no event takes place at different venues at the same time, and $CT \to V$ expresses that no company is in charge of different venues at the same time. The minimal keys are ET, VT and CT. Hence, every attribute of HAP is prime. While HAP is in 3NF it has no FD-preserving BCNF decomposition. Given FD-preservation, classical normalization cannot achieve more.

However, we cannot measure the level of effort to maintain data consistency for HAP. In instance r of Table 1 each of the $\ell_1 > 1$ occurrences of C-value *Kilo* is redundant due to the FD $E \to C$, and each of the $\ell_2 > 1$ occurrences of C-value *Mega* is redundant due to $V \to C$. Recall that a value occurrence is redundant whenever every update of the occurrence to a different value results in a violation of the FD. We refer to the number of different tuples in which a given data value can occur redundantly as *the level of data redundancy*. Clearly, the level of data redundancy on HAP is unbounded. This is true for every schema that is in 3NF but not in BCNF. As we will show, every FD lost during BCNF decomposition still causes an unbounded level of data redundancy. Hence, the number of data values that need updating is a priori unbounded. Changing one occurrence of *Kilo* means that a total of ℓ_1 occurrences of *Kilo* need updating to ensure data consistency. We refer to the total number of occurrences that require an update to achieve consistency as the *level of update inefficiency*. So, for FD-preserving BCNF decompositions the level of effort for data consistency is at optimum 1, while it is unbounded in all other cases.

In practice, however, ℓ_1 and ℓ_2 represent upper bounds of *cardinality constraints* (CCs). They constitute a different class of business rules than FDs. For a positive integer ℓ, the CC $card(X) \leq \ell$ says that every instance can have up to ℓ different records with matching values on all the attributes in X. For $\ell = 1$, X is a key. For $\ell = \infty$, no bound has been specified. For example, the CC $card(E) \leq \ell_1$ with $\ell_1 = 1,000$ ($1k$) expresses that every event can have up to $1k$ different combinations of venues and times. Similarly, the CC $card(V) \leq \ell_2$ with $\ell_2 = 1,000,000$ ($1m$) expresses that every venue can have up to $1m$ different combinations of events and times. CCs inform schema design beyond classical normalization. Given CCs, the 3NF schema HAP admits level ℓ_2 data redundancy and update inefficiency. Without CCs no integer bounds are specified, and every non-trivial or lost FD causes an unbounded level of data redundancy and update inefficiency, unless its left-hand side is a key. Hence, we propose to include CCs in schema design.

With CCs we can quantify the level of effort required to maintain data consistency under updates. For example, we may ask which FD-preserving decompositions of HAP minimize level ℓ. Applying one of our new algorithms results in schema $\mathcal{D}_1 = \{(R_2, \Sigma_2), (R_3, \Sigma_3), (R_4, \Sigma_4)\}$ in Table 2 with $R_2 = CV$ and $\Sigma_2 = \{V \to C\}$, $R_3 = ETV$

Figure 1: Level ℓ of Effort Required For Data Consistency, and its Impact on Update and Join Efficiency



and $\Sigma_3 = \{TV \to E, ET \to V\}$, and $R_4 = CET$ and $\Sigma_4 = \{CT \to E, E \to C\}$. The schemata (R_2, Σ_2) and (R_3, Σ_3) are both in BCNF and cannot exhibit any redundant data value. However, R_4 is in 3NF and still exhibits an $\ell_1 = 1k$ level of data redundancy and update inefficiency caused by $E \to C$. Indeed, $\ell_1 = 1k$ is the optimum level achievable by any FD-preserving decomposition of Hap. The given FD $CT \to V$ is preserved by $\mathcal{D}_1$ as it is implied by $CT \to E$ and $ET \to V$. The decomposition $\mathcal{D}_2 = \{(R_1, \Sigma_1), (R_3, \Sigma_3), (R_5, \Sigma_5)\}$ with $R_1 = EC$ and $\Sigma_1 = \{E \to C\}$, $R_5 = CTV$ and $\Sigma_5 = \{CT \to V, V \to C\}$ is in 3NF and achieves level $\ell_2 = 1m$ data redundancy.

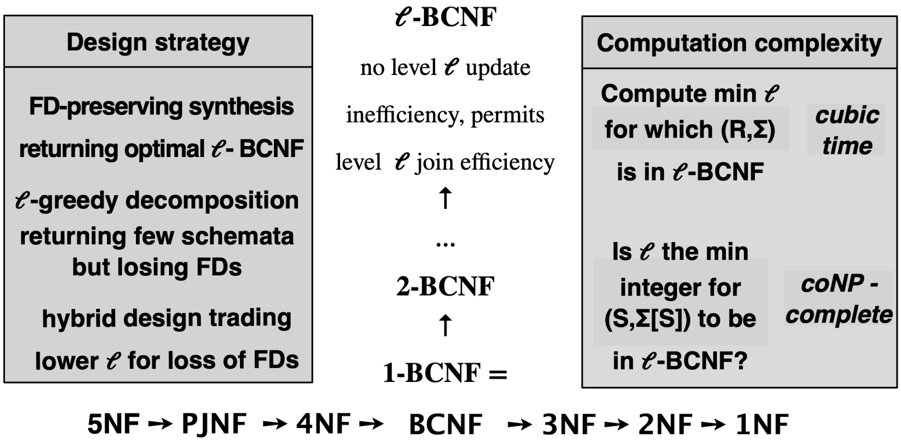
CCs were already introduced in Chen's seminal ER paper [9] and used by UML, XML, and OWL. Surprisingly, they have not been used for normalization. We also observe that CCs quantify *the level of join efficiency* for a schema. We define the latter as the maximum number of values an FD can join with some redundant value in any instance of the schema. Instance r_4 over R_4 in Table 2 satisfies $E \to C$. Hence, r_4 is equal to the join $\pi_{EC}(r_4) \bowtie \pi_{ET}(r_4)$. Indeed, the redundant data C-value *Kilo* can be joined with the ℓ_1 T-values $t_1, \ldots, t_{\ell_1}$ via the E-value *Party*. Without CCs, the join strength is unbounded, so the classical theory does not provide insight into the join efficiency of a schema. CCs inform the selection of materialized views, even before the database is operational. Here the level ℓ of a materialized view quantifies both its effort required for incremental maintenance and its support for joins. Defining Hap as view over $\mathcal{D}_1$, its level of maintenance and join efficiency is $\ell_2 = 1m$.

Contributions. Our two main contributions are:

- We establish the first framework for logical schema normalization that quantifies the effort required to achieve data consistency during updates.
- Our framework also quantifies the join efficiency of schemata, which means it informs de-normalization already during logical design.

Figure 1 illustrates our main ideas, in particular that ℓ quantifies both the update effort and join capability of schemata. Figure 2 illustrates our technical results, which are:

Figure 2: Achievements of ℓ-BCNF



(1) We relax the classical BCNF condition by permitting every value combination over the left-hand side of every non-trivial FD to occur in up to ℓ tuples of every instance. Hence, we obtain the infinite hierarchy of ℓ-Bounded Cardinality Normal Forms (ℓ-BCNF), with classical BCNF captured for $\ell = 1$. Therefore, the minimum ℓ for which ℓ-BCNF is attainable measures the effort required to achieve data consistency.
(2) We show for every ℓ that schemata in ℓ-BCNF characterize instances that are i) free from level ℓ data redundancy and update inefficiency, and ii) permit a level ℓ of join efficiency.
(3) We establish an algorithm that computes schemata in ℓ-BCNF for the minimum level ℓ attainable across FD-preserving decompositions. Another algorithm prioritizes FDs for lossless decompositions based on the level of data redundancy they cause. This algorithm produces few output schemata, but FDs may be lost. Combining both algorithms to a hybrid strategy, we further reduce the minimum level attainable from FD-preserving decompositions by losing FDs.
(4) Level ℓ informs the selection of materialized views as it captures both the incremental effort to maintain the views under updates, and the support for join queries by the views.
(5) Experiments with synthetic and real-world data illustrate which properties the schemata have that result from our algorithms, and how these properties predict the physical performance of update and query operations on instances over the schemata, without and with materialized views.
Organization. We recall preliminaries in Section 2. In Section 3 we introduce our family of ℓ-BCNF and its computational properties. In Section 5 we establish which properties schemata in ℓ-BCNF exhibit in terms of updates and joins. We discuss three design algorithms in Section 6. Experimental results are presented in Section 7. Related work is discussed in Section 9. We conclude in Section 10. The data set are available for download[1].

[1] `https://bit.ly/3c0fE9k`

Table 3: Axiomatizations of CCs and FDs

$\dfrac{}{card(R) \leq 1}$ (set, $\mathcal{S}$)	$\dfrac{}{card(X) \leq \infty}$ (unbounded, $\mathcal{U}$)	$\dfrac{card(X) \leq \ell}{card(X) \leq \ell + 1}$ (loosen, $\mathcal{L}$)	$\dfrac{card(X) \leq \ell}{card(XY) \leq \ell}$ (adding, $\mathcal{A}$)
$\dfrac{}{XY \to Y}$ (reflexivity, $\mathcal{R}$)	$\dfrac{X \to Y}{X \to XY}$ (extension, $\mathcal{E}$)	$\dfrac{X \to Y \quad Y \to Z}{X \to Z}$ (transitivity, $\mathcal{T}$)	
$\dfrac{card(X) \leq 1}{X \to Y}$ (key, $\mathcal{K}$)	$\dfrac{X \to Y \quad card(Y) \leq \ell}{card(X) \leq \ell}$ (pullback, $\mathcal{P}$)		

2 Preliminaries

Classical normalization for FDs is part of most textbooks for relational databases. For a detailed background on classical normalization we refer to [4, 3, 38].

2.1 Design Foundations

Just like BCNF and 3NF need to efficiently decide the implication problem for FDs, Bounded Cardinality Normal Forms depends on the ability to efficiently decide the implication problem for the combination of CCs and FDs. We adopt standard notions such as the *implication* (*inference*) of a constraint φ by a set Σ of constraints (using a set $\mathfrak{R}$ of inference rules), denoted by $\Sigma \models \varphi$ ($\Sigma \vdash_{\mathfrak{R}} \varphi$), the *semantic closure* Σ^* (*syntactic closure* $\Sigma^+_{\mathfrak{R}}$) for a set Σ of constraints as the set of constraints implied (inferred) by Σ (using $\mathfrak{R}$), an *axiomatization* as a set of rules $\mathfrak{R}$ that is sound and complete ($\Sigma^* = \Sigma^+_{\mathfrak{R}}$ for all Σ), and that two constraint sets Σ and Θ are *covers* of one another iff $\Sigma^* = \Theta^*$ [38].

An FD over relation schema R is an expression $X \to Y$ with attribute sets $X, Y \subseteq R$, and a relation r over R satisfies $X \to Y$ whenever every pair of tuples $t, t' \in r$ that has matching values on all the attributes in X also has matching values on all the attributes in Y, that is, $\forall t, t' \in r(t(X) = t'(X) \Rightarrow t(Y) = t'(Y))$. For example, the relation over $R = \{Emp, Dep, Mgr\}$ satisfies the FDs $Emp \to \{Dep, Mgr\}$, $Dep \to Mgr$ and $Mgr \to Dep$, but violates the FD $\{Dep, Mgr\} \to Emp$. Armstrong's axioms are sound and complete for the implication of FDs alone [2]. They are shown in Table 3 as the set $\mathfrak{A} = \{\mathcal{R}, \mathcal{E}, \mathcal{T}\}$. In fact, Table 3 shows two additional axiomatizations. The system $\mathfrak{H} = \{\mathcal{S}, \mathcal{U}, \mathcal{L}, \mathcal{A}\}$ forms an axiomatization for CCs alone [21], and the system $\mathfrak{L} = \mathfrak{H} \cup \mathfrak{A} \cup \{\mathcal{K}, \mathcal{P}\}$ forms an axiomatization for CCs and FDs together [21]. A CC over relation schema R is an expression $card(X) \leq l$ where $X \subseteq R$ and ℓ is a positive integer, and a relation r over R satisfies $card(X) \leq l$ whenever there are no more than ℓ different tuples in r that all have matching values on all the attributes in X, that is, $\forall t_1, \ldots, t_{\ell+1} \in r(t_1(X) = \ldots = t_{\ell+1}(X) \Rightarrow \exists i, j \in \{1, \ldots, \ell + 1\}(t_i = t_j))$. In the introduction, the relation over $R = \{Emp, Dep, Mgr\}$ satisfies the CCs $card(Emp) \leq 1$, $card(Dep, Mgr) \leq \ell + 1$ but violates the CC $card(Dep, Mgr) \leq \ell$ as there are $\ell + 1$ different employees at the same department with the same manager.

Typically, axiomatizations lead us to efficient algorithms that decide implication. An FD $X \to Y$ is implied by an FD set Σ iff the attribute set closure $X^+_\Sigma = \{A \in R \mid \Sigma \models X \to A\}$ contains Y, that is, $Y \subseteq X^+_\Sigma$. This provides a linear-time algorithm for FD implication. Though more expressive than CCs and FDs in isolation, the combined implication problem can be reduced to that for FDs alone by translating any set Σ of CCs and FDs into the FD set $\Sigma[FD] = \{X \to Y \mid X \to Y \in \Sigma\} \cup \{X \to R \mid \mathit{card}(X) \leq 1 \in \Sigma\}$ [21].

Theorem 1 ([21]) *For a given set Σ of CCs and FDs over a relation schema R, the following hold: (1) $\Sigma \models X \to Y$ if and only if $X \to Y \in \Sigma[FD]$, and (2) For every positive integer ℓ: $\Sigma \models \mathit{card}(X) \leq \ell$ if and only if $Y \subseteq X^+_{\Sigma[FD]}$ for some $\mathit{card}(Y) \leq \ell' \in \Sigma \cup \{\mathit{card}(R) \leq 1\}$ where $\ell' \leq \ell$.*

It follows that the implication problem for the combined class of CCs and FDs can thus be decided in time $\mathcal{O}(|\Sigma| \times ||\Sigma||)$, so in worst-case quadratic time, where $|\Sigma|$ denotes the number of elements in Σ and $||\Sigma||$ denotes the total number of attribute occurrences in Σ. Continuing our example, if Σ consists of the FD $\mathit{Emp} \to \{\mathit{Mgr}\}$ and the CC $\mathit{card}(\mathit{Mgr}) \leq 3$, then the CC $\mathit{card}(\mathit{Emp}) \leq 3$ is implied by Σ since $\{\mathit{Emp}\}^+_{\Sigma[\mathrm{FD}]} = \{\mathit{Emp}, \mathit{Mgr}\}$ and $\mathit{Mgr} \in \{\mathit{Emp}\}^+_{\Sigma[\mathrm{FD}]}$ for the CC $\mathit{card}(\mathit{Mgr}) \leq 3$ and $\ell' = 3 \leq 3 = \ell$.

2.2 Boyce-Codd Normal Form

For a given FD set Σ over a given relation schema R, (R, Σ) is in Boyce-Codd Normal Form (BCNF) iff for every FD $X \to Y \in \Sigma^+_{\mathfrak{A}}$ where $Y \not\subseteq X$, $X \to R \in \Sigma^+_{\mathfrak{A}}$ [38]. In fact, (R, Σ) is in BCNF iff for every FD $X \to Y \in \Sigma$ where $Y \not\subseteq X$, $X \to R \in \Sigma^+_{\mathfrak{A}}$ [38]. Hence, deciding whether (R, Σ) is in BCNF can be done in time quadratic in the input. However, for a given sub-schema $S \subseteq R$ it is *coNP*-complete to decide whether $(S, \Sigma[S])$ is in BCNF, where $\Sigma[S] = \{X \to Y \in \Sigma^+_{\mathfrak{A}} \mid X, Y \subseteq S\}$ [3, 38]. A *decomposition* of relation schema R is a set $\mathcal{D}$ of relation schemata such that $\bigcup_{S \in \mathcal{D}} S = R$. A decomposition $\mathcal{D}$ of R with FD set Σ is *lossless* if for every relation r over R that satisfies Σ, $r = \bowtie_{S \in \mathcal{D}} r[S]$. Here, $r[S] = \{t(S) \mid t \in r\}$. A *BCNF* decomposition of R with FD set Σ is a decomposition $\mathcal{D}$ of R where for every $S \in \mathcal{D}$, $(S, \Sigma[S])$ is in BCNF. A decomposition $\mathcal{D}$ of (R, Σ) is FD-preserving iff $\forall \sigma \in \Sigma \left(\bigcup_{S \in \mathcal{D}} \Sigma[S] \models \sigma\right)$. Vincent introduced *redundant data values* and showed that BCNF is equivalent to schemata that do not permit any redundant data values in any instances that satisfy the given FD set [49]. The notion of the level of data redundancy is a new.

2.3 Third Normal Form

For an arbitrary schema (R, Σ) we cannot guarantee that there is any FD-preserving, lossless BCNF decomposition [38]. In response, there is the *Third Normal Form* (3NF) proposal [6]. For a given FD set Σ over a given relation schema R, (R, Σ) is in 3NF iff for every FD $X \to Y \in \Sigma^+_{\mathfrak{A}}$ where $Y \not\subseteq X$ holds, $X \to R \in \Sigma^+_{\mathfrak{A}}$ or every attribute in $Y - X$ is a prime attribute. Evidently from this definition, every schema in BCNF is also in 3NF. An attribute $A \in R$ is *prime* iff it is contained in some minimal key, that is, $A \in K$ for some $K \to R \in \Sigma^+_{\mathfrak{A}}$ such that for all proper subsets $K' \subset K$, $K' \to R \notin \Sigma^+_{\mathfrak{A}}$. A *3NF* synthesis of R with FD set Σ is a decomposition $\mathcal{D}$ of R where for every $S \in \mathcal{D}$, $(S, \Sigma[S])$ is in 3NF. The main idea behind 3NF was to minimize data redundancy while preserving all FDs [28]. As illustrated by our simple example, 3NF cannot guarantee any upper bound on the level of data redundancy. Validating if a given schema (R, Σ) is in 3NF is *coNP*-complete due to the requirement of having to compute all minimal keys [3].

2.4 Normalization

A relation is the lossless join over its projections on XY and $X(R-Y)$ whenever the relation satisfies the FD $X \to Y$ [38]. Splitting (R, Σ) into $(XY, \Sigma[XY])$ and $(X(R-Y), \Sigma[X(R-Y)])$ eliminates all redundant data values caused by $X \to Y$ as X is a key on $(XY, \Sigma[XY])$. BCNF decomposition starts with (R, Σ) and keeps splitting any schema $(S, \Sigma[S])$ that is not yet in BCNF. A cover Σ for a given FD set is *non-redundant* iff for all $\sigma \in \Sigma$, $\Sigma - \{\sigma\} \not\models \sigma$. A cover Σ for a given FD set is *L-reduced* iff for every $X \to Y \in \Sigma$ there is no proper subset $Z \subset X$ such that $\Sigma \models Z \to Y$. We call a non-redundant, L-reduced cover with unique left-hand sides a *canonical cover* [37]. If (R, Σ) is not in 3NF, *3NF synthesis* computes a canonical cover, and selects the output schemata as $(XY, \Sigma[XY])$ where XY is maximal under set containment among the FDs $X \to Y$ of the canonical cover, and adds a minimal key $(S, \Sigma[S])$ to remain lossless [6].

2.5 Summary

There is no previous schema design approach that addresses update and join efficiency together. With FDs alone, BCNF decomposition cannot measure the level of data redundancy caused by lost FDs, and 3NF cannot measure the level of data redundancy caused by non-key FDs.

3 The Family of ℓ-BCNF

We establish the family of ℓ-Bounded Cardinality Normal Forms and some computational properties. Its levels of update inefficiency and join efficiency are studied in Section 5.

A schema is in BCNF whenever the left-hand side (LHS) X of every non-trivial FD $X \to Y$ is a key. Since X is a key whenever $card(X) \leq 1$ holds, CCs provide a convenient mechanism to relax the BCNF condition as follows.

Definition 1 *Let Σ denote a set of CCs and FDs over relation schema R, and let $\ell \in \mathbb{N}^{\infty}_{\geq 1}$. (R, Σ) is in* ℓ-Bounded Cardinality Normal Form *(ℓ-BCNF) iff for all $X \to Y \in \Sigma^+_{\mathfrak{L}}$ where $Y \not\subseteq X$ we have that $card(X) \leq \ell \in \Sigma^+_{\mathfrak{L}}$.*

Example 1 *For our running example* Hap *consider the FD set Θ of $E \to C$, $V \to C$, $VT \to E$, $ET \to V$, and $CT \to V$. While* $(\textsc{Hap}, \Theta)$ *is in 3NF, it is in ∞-BCNF as the smallest ℓ for which $card(E) \leq \ell \in \Theta^+_{\mathfrak{L}}$ is $\ell = \infty$. Intuitively, this matches our understanding that the levels of data redundancy, update inefficiency, and join efficiency* *are unbounded. Consider now (*HAP,Σ*) where* Σ *consists of the FDs in* Θ *and the CCs* $card(E) \leq 1k$ *and* $card(V) \leq 1m$*. Since* VT*,* ET*, and* CT *are (minimal) keys, we obtain the CCs* $card(VT) \leq 1$*,* $card(ET) \leq 1$*, and* $card(CT) \leq 1$*. Hence, every non-trivial* $X \to Y \in \Sigma$ *satisfies* $card(X) \leq 1m \in \Sigma^+_{\mathfrak{C}}$*. The latter condition is equivalent to being in* ℓ*-BCNF (Theorem 4), so* (HAP, Σ) *is in* $1m$*-BCNF.* (HAP, Σ) *is not in any* ℓ*-BCNF for* $\ell < 1m$ *since we have the FD* $V \to C \in \Sigma$ *but the smallest* ℓ_2 *with* $card(V) \leq \ell_2 \in \Sigma^+_{\mathfrak{C}}$ *is* $\ell_2 = 1m$*.*

Our definition of ℓ-BCNF has the desirable property to be invariant under covers, as stated in the next result.

Theorem 2 *Let* Σ *and* Θ *denote two CC/FD sets over* R *that are covers of one another. For all* $\ell \in \mathbb{N}^{\infty}_{\geq 1}$*,* (R, Σ) *is in* ℓ*-BCNF iff* (R, Θ) *is in* ℓ*-BCNF.*

Proof (R, Σ) is in ℓ*-BCNF* iff for all $X \to Y \in \Sigma^+_{\mathfrak{C}}$ where $Y \not\subseteq X$ we have that $card(X) \leq \ell \in \Sigma^+_{\mathfrak{C}}$. Since Θ and Σ are covers of one another, it follows that $\Sigma^+_{\mathfrak{C}} = \Theta^+_{\mathfrak{C}}$. Consequently, for all $X \to Y \in \Theta^+_{\mathfrak{C}}$ where $Y \not\subseteq X$ we have that $card(X) \leq \ell \in \Theta^+_{\mathfrak{C}}$ iff (R, Σ) is in ℓ*-BCNF.* ∎

Theorem 2 means we need not worry how we represent application semantics as integrity constraints. For example, let Σ' denote the FDs in Θ together with the CCs $card(EC) \leq 1k$ and $card(VC) \leq 1m$. Then Σ and Σ' are covers of one another, as easily seen from the adding rule $\mathcal{A}$, the extension rule $\mathcal{E}$ and the pullback rule $\mathcal{P}$. Hence, (HAP,Σ') is in $1m$-BCNF but not in ℓ-BCNF for any $\ell < 1m$.

Our definition gives rise to a family of syntactic normal forms that exhibit a strict hierarchy. At the bottom is 1-BCNF, which is equivalent to the classical Boyce-Codd normal form.

Theorem 3 *For every schema* (R, Σ) *and every positive integer* ℓ *we have: If* (R, Σ) *is in* ℓ*-BCNF, then* (R, Σ) *is also in* $\ell+1$*-BCNF. However, for every positive integer* ℓ *there are schemata* (R, Σ) *in* $\ell + 1$*-BCNF that are not in* ℓ*-BCNF.*

Proof For the first part assume that (R, Σ) is in ℓ-BCNF. That is, for every $X \to Y \in \Sigma^+_{\mathfrak{C}}$ where $Y \not\subseteq X$ we have $card(X) \leq \ell \in \Sigma^+_{\mathfrak{C}}$. Consequently, for every $X \to Y \in \Sigma^+_{\mathfrak{C}}$ where $Y \not\subseteq X$ we have $card(X) \leq \ell + 1 \in \Sigma^+_{\mathfrak{C}}$.

For the second part let $R = \{A, B, C\}$ and $\Sigma = \{A \to B, card(A) \leq \ell{+}1\}$. Obviously, (R, Σ) is in $\ell + 1$-BCNF, but not in ℓ-BCNF. Indeed, the relation $r = \{t_0, \ldots, t_\ell\}$ over R where $t_i(A) = t_i(B) = 0$ and $t_i(C) = i$ for all $i = 0, \ldots, \ell$, satisfies Σ, but r does not satisfy $card(A) \leq \ell$. ∎

For example, (HAP, Σ) is in ℓ-BCNF if and only if $\ell \geq 1m$. Theorem 3 establishes the first infinite hierarchy of normal forms for relational schema design. It is orthogonal to known normal forms such as 3NF, BCNF, 4NF, etc.

Combining CCs and FDs raises expressiveness without adding (much) computational complexity. As a generalization of BCNF, known hardness results apply to ℓ-BCNF.

3.1 Efficient Decidability Locally

Firstly, we show that deciding whether for a given schema (R, Σ) and for a given positive integer ℓ, (R, Σ) is in ℓ-BCNF can be done in cubic time in the input. This is a consequence of showing that it suffices to check the FDs in Σ to validate whether (R, Σ) is in ℓ-BCNF.

Theorem 4 *For all (R, Σ) and $\ell \in \mathbb{N}_{\geq 1}^{\infty}$, (R, Σ) is in ℓ-BCNF iff for all $X \to Y \in \Sigma$ where $Y \not\subseteq X$, we have that $card(X) \leq \ell \in \Sigma_{\mathfrak{C}}^{+}$. We can decide in $\mathcal{O}(|\Sigma|^2 \times ||\Sigma||)$ time if for a given schema (R, Σ) and $\ell \in \mathbb{N}_{\geq 1}^{\infty}$, (R, Σ) is in ℓ-BCNF.*

Proof If (R, Σ) is in ℓ-BCNF, then certainly for all $X \to Y \in \Sigma$ where $Y \not\subseteq X$, we have that $card(X) \leq \ell \in \Sigma_{\mathfrak{C}}^{+}$, because $\Sigma \subseteq \Sigma_{\mathfrak{C}}^{+}$. However, vice versa we need to show the following: if for all $X \to Y \in \Sigma$ where $Y \not\subseteq X$, we have that $card(X) \leq \ell \in \Sigma_{\mathfrak{C}}^{+}$, then for all $X \to Y \in \Sigma_{\mathfrak{C}}^{+}$ where $Y \not\subseteq X$, we have that $card(X) \leq \ell \in \Sigma_{\mathfrak{C}}^{+}$.

For that purpose, consider the strict chain:

$$\Sigma = \Sigma_0 \subset \Sigma_1 \subset \cdots \subset \Sigma_k = \Sigma_{\mathfrak{C}}^{+}$$

where the single constraint in $\Sigma_j - \Sigma_{j-1}$ results from an application of a single rule in $\mathfrak{G}$ to the number of required constraints in Σ_{j-1}. We show that there is already some FD $X' \to Y' \in \Sigma_{j-1}$ where $Y' \not\subseteq X'$ and $card(X') \leq \ell \notin \Sigma_{\mathfrak{C}}^{+}$, whenever there is some FD $X \to Y \in \Sigma_j$ where $Y \not\subseteq X$ and $card(X) \leq \ell \notin \Sigma_{\mathfrak{C}}^{+}$. Let $j > 0$ and $X \to Y \in \Sigma_j - \Sigma_{j-1}$ such that $Y \not\subseteq X$ and $card(X) \leq \ell \notin \Sigma_{\mathfrak{C}}^{+}$. Since $Y \not\subseteq X$, $X \to Y$ was not inferred by application of the reflexivity axiom $\mathcal{R}$.

Assume that $X \to XY$ was inferred from $X \to Y \in \Sigma_{j-1}$ using the extension rule $\mathcal{E}$. Since $XY \not\subseteq X$, it certainly follows that $Y \not\subseteq X$. Hence, there is some non-trivial FD $X \to Y$ in Σ_{j-1} such that $card(X) \leq \ell \notin \Sigma_{\mathfrak{C}}^{+}$.

Assume that $X \to Y$ was inferred from $X \to Z, Z \to Y \in \Sigma_{j-1}$ using the transitivity rule $\mathcal{T}$. In case that $Z \not\subseteq X$, we have the non-trivial FD $X \to Z$ in Σ_{j-1} such that $card(X) \leq \ell \notin \Sigma_{\mathfrak{C}}^{+}$. Otherwise, $Z \subseteq X$ holds. Consequently, $Y \not\subseteq Z$ as otherwise $Y \subseteq X$ would contradict our assumption. If $card(Z) \leq \ell \in \Sigma_{\mathfrak{C}}^{+}$ held, then from $Z \subseteq X$ and $X \to Z \in \Sigma_{\mathfrak{C}}^{+}$ (by application of the reflexivity axiom) it would follows that $card(X) \leq \ell \in \Sigma_{\mathfrak{C}}^{+}$, too. This, however, is a contradiction to our assumption, and consequently $card(Z) \leq \ell \notin \Sigma_{\mathfrak{C}}^{+}$. That is, there is some non-trivial FD $Z \to Y$ in Σ_{j-1} such that $card(Z) \leq \ell \notin \Sigma_{\mathfrak{C}}^{+}$.

Finally, assume that $X \to Y$ was inferred from $card(X) \leq 1 \in \Sigma_{j-1}$ using the demotion rule $\mathcal{D}$. This, however, is not possible as otherwise $card(X) \leq \ell \in \Sigma_{\mathfrak{C}}^{+}$ by a finite number of applications of the weakening rule $\mathcal{W}$.

As there are no other possibilities of an inference, the proof is complete.

Hence, to decide whether (R, Σ) is in ℓ-BCNF for a given ℓ, we just need to decide for every FD $X \to Y\Sigma$ whether it is non-trivial and whether $card(X) \leq \ell \in \Sigma_{\mathfrak{C}}^{+}$ holds. The latter condition can be verified in time $\mathcal{O}(|\Sigma| \times ||\Sigma||)$ by Theorem 1. This completes the proof. ∎

4 Computing the strongest normal form locally

Our new setting motivates the problems of computing i) the minimum level of data redundancy and update inefficiency that a given schema prevents, and iii) the minimum level of join efficiency that a given schema permits. Due to Theorem 6, these problems amount to computing the smallest positive integer ℓ for which the given schema is in ℓ-BCNF. This is done as follows: If there are no FDs in the input, the output level will be 1. Otherwise, we start with the worst possible level ∞. For each non-trivial input FD with LHS X we determine the smallest positive integer ℓ_X by which X is bound according to the input, using Theorem 1(2). We terminate with ∞ as soon as we find that $\ell_X = \infty$ for some X. Otherwise, we return the maximum ℓ_X among those computed. In this procedure we check for each FD $X \to Y$ and for each CC in the input whether it implies some smaller bound ℓ_X.

Algorithm 1 Strongest ℓ-Bounded Cardinality Normal Form

Require: (R, Σ) with CC/FD set Σ over schema R
Ensure: Minimum integer ℓ such that (R, Σ) in ℓ-BCNF

if $\Sigma_{\text{FD}} = \emptyset$ **then**
    return 1
$\ell \leftarrow \infty$
$\Sigma_{\text{card}} \leftarrow \Sigma_{\text{card}} \cup \{card(R) \leq 1\}$
for all $X \to Y \in \Sigma_{\text{FD}}$ such that $Y \not\subseteq X$ **do**
    if ℓ_X has not been defined before **then**
        $\ell_X \leftarrow \infty$
    for all $card(Y) \leq \ell' \in \Sigma_{\text{card}}$ with $\ell' < \ell_X$ **do**
        if $Y \subseteq X^+_{\Sigma[\text{FD}]}$ **then**
            $\ell_X \leftarrow \ell'$
    if $\ell_X \leftarrow \infty$ **then**
        return ∞
return $\max_X\{\ell_X\}$

Algorithm 1 computes the strongest ℓ-BCNF in the sense that ℓ is the smallest positive integer with that property. For each FD $X \to Y$ in the input we check for each cardinality constraint in the input whether it implies some smaller bound ℓ_X. Algorithm 1 therefore operates in $\mathcal{O}(|\Sigma_{\text{FD}}| \times |\Sigma_{\text{card}}| \times ||\Sigma||)$ time. We thus obtain the following result.

Corollary 1 *Given a set Σ of CCs and FDs over relation schema R, we can compute in $\mathcal{O}(|\Sigma_{FD}| \times |\Sigma_{card}| \times ||\Sigma||)$ time the smallest positive integer ℓ such that (R, Σ) is in ℓ-BCNF (and equivalently in ℓ-RFNF, ℓ-UINF, or in ℓ-JENF).*

For (HAP, Σ), $\ell_E = 1k$, $\ell_V = 1m$, $\ell_{VT} = 1$, $\ell_{ET} = 1$, and $\ell_{CT} = 1$. Since the maximum is $\ell = \ell_2 = 1m$, ℓ_2 is the optimum ℓ for which (HAP, Σ) is in ℓ-BCNF.

4.1 Likely intractability globally

While we can efficiently compute the strongest normal form locally, this is unlikely to be efficient globally. Given (R, Σ, ℓ) and a subschema $S \subseteq R$, it is unlikely we can find a PTIME algorithm deciding if $(S, \Sigma[S])$ is in ℓ-BCNF.

Theorem 5 *Let Σ denote a set of CCs and FDs over R, $S \subset R$, and ℓ a positive integer. Given (R, S, Σ, ℓ), it is* coNP*-complete to decide whether $(S, \Sigma[S])$ is in ℓ-BCNF.*

Proof The *coNP*-hardness follows from the special case $\ell = 1$, since the problem of deciding whether a subschema is in Boyce-Codd Normal Form is already *coNP*-complete [3]. Membership follows from the fact that we can guess $\Sigma[S]$ and verify in polynomial-time that $(S, \Sigma[S])$ is not in ℓ-BCNF. ∎

While it is already *coNP*-complete to decide if a given subschema is in BCNF (1-BCNF), Theorem 5 encourages us to view schema normalization from the new perspective of computing the smallest ℓ for which a schema is in ℓ-BCNF.

5 Useful Properties of ℓ-BCNF

We show that schemata in ℓ-BCNF characterize instances with useful properties for updates and joins. In particular, ℓ quantifies the highest number of values that need updating to achieve data consistency, but also how many data values can be joined with a given redundant value.

5.1 ℓ-redundancy

Intuitively, Vincent [49] defined a single occurrence of a data value as *redundant* whenever every change to this occurrence results in a relation that violates some given constraint. Our idea is to fix some positive integer ℓ, and define a data value as *ℓ-redundant* whenever there are ℓ distinct tuples in which the value occurs and every update to at least one of these ℓ occurrences results in a relation that violates a given constraint. That is, the value of these ℓ occurrences is already uniquely determined by the other values and knowing the constraints are satisfied by the relation.

For example, the value *Kilo* in relation r_4 in Table 1 is 999- but not $1k$-redundant ($\ell_1 = 1k$): Concealing between 1 and 999 occurrences of the value *Kilo* in the C-column still allows us to determine each of the concealed occurrences as $E \rightarrow C$ must hold and there is at least one tuple left that has a matching E-value and C-value *Kilo*, as illustrated next.

E	C	T
Party	?	t_1
$\vdots$	$\vdots$	$\vdots$
Party	?	t_{999}
Party	Kilo	t_{1k}

$E \rightarrow C$
? = Kilo

E	C	T
Party	Kilo	t_1
$\vdots$	$\vdots$	$\vdots$
Party	Kilo	t_{999}
Party	Kilo	t_{1k}

However, concealing all $1k$ occurrences means the tuples could have any value on C (as long as they all match on C).

We define a family of semantic normal forms by stipulating the absence of any relations that feature any ℓ-redundancy. A schema (R, Σ) is in *ℓ-Redundancy Free Normal Form* (*ℓ-RFNF*) iff there is no relation r over R that satisfies Σ, there is no $\sigma \in \Sigma$, and there is no attribute $A \in R$ for which there is a data value $v \in r(A)$ that is ℓ-redundant for σ.

$(\textsc{Hap}, \Sigma)$ from Example 1 is in $1m$-RFNF ($\ell_2 = 1m$), but not in ℓ-RFNF for any $\ell < \ell_2$. Concealing any fewer than the ℓ_2 occurrences of *Mega* will allow us to deduce their value.

More formally, the data value $v \in r(A)$ is *ℓ-redundant* for $\sigma \in \Sigma$ iff there are ℓ distinct tuples $t_1, \ldots, t_\ell \in r$ such that $t_1(A) = \ldots = t_\ell(A) = v$, and for every ℓ-transaction $\{t'_1, \ldots, t'_\ell\}$ of $t_1, \ldots, t_\ell$ for A, the relation $r' := (r - \{t_1, \ldots, t_l\}) \cup \{t'_1, \ldots, t'_\ell\}$ violates σ.

An ℓ-transaction causes an actual update to at least one of ℓ values. More formally, an *ℓ-transaction* of $t_1, \ldots, t_\ell$ for A is a set $\{t'_1, \ldots, t'_\ell\}$ of tuples over R such that for all $i = 1, \ldots, \ell$ and for all $A' \in R - \{A\}$, $t'_i(A') = t_i(A')$, and there is some $j \in \{1, \ldots, \ell\}$ such that $t'_j(A) \neq t_j(A)$.

5.2 ℓ-update inefficiency

An ℓ-redundant data value is synonymous with that of an ℓ-update inefficiency in the following sense. No matter how any ℓ occurrences of the ℓ-redundant data value are updated, consistency cannot be achieved since there will still remain some occurrence of the value that could not have been updated. For the table on the right above, no matter how we update 999 occurrences of *Kilo*, as long as we update at least one of them we cannot satisfy $E \to C$. However, updating all 1000 occurrences of *Kilo* consistently to a new value will satisfy $E \to C$. Based on the CC $\mathit{card}(E) \leq 1k$, we require the update of at most 1000 tuples to maintain consistency for $E \to C$, in any legal relation. Hence, it is justified to say that (R, Σ) is in *ℓ-Update Inefficiency Normal Form* (UINF) iff (R, Σ) is in ℓ-RFNF.

If (R, Σ) is in *ℓ-RFNF* (ℓ-UINF), then ℓ is a *level of data redundancy (update inefficiency) that* (R, Σ) *prevents*. Otherwise, (R, Σ) *permits this level*. If there is no ℓ for which (R, Σ) is in *ℓ-RFNF* (ℓ-UINF), no a priori upper bound exists for the level that (R, Σ) prevents. In this case, $\ell := \infty$.

5.3 ℓ-join efficiency

We define join efficiency as the maximum number of tuples that have matching values on the LHS of a non-trivial FD. This captures the update and join efficiency due to FDs, a core trade-off for logical schema design.

Formally, if r satisfies the non-trivial FD $X \to Y$ over relation schema R, then r is the lossless join of its projections on XY and $X(R-Y)$. That is, $r = \pi_{XY}(r) \bowtie \pi_{X(R-Y)}(r)$. Hence, the X-value for a fixed tuple $t \in r$ joins the unique Y-value of t with any $R-Y$-values of all tuples $t'_1, \ldots, t'_\ell \in r$ that have matching values with t on X. The number ℓ of those tuples denotes the *join strength* of $t \in r$ for $X \to Y$. The join strength of r is the maximum join strength of any tuple in r. Finally, the *join efficiency* of (R, Σ) is the maximum join strength of any relation over R that satisfies Σ. A schema (R,Σ) is in *ℓ-Join Efficiency Normal Form* (JENF) iff the join efficiency of (R,Σ) is at most ℓ.

For example, the join efficiency of schema $(ECT, \{CT \to E, E \to C, \mathit{card}(E) \leq 1k\})$ is $1k$. The following relation r joins the redundant C-value *Kilo* with the $1k$ different values $t_1,\ldots,t_{1k}$. As r satisfies $E \to C$, we have $r = r[EC] \bowtie r[ET]$.

$$\begin{array}{ccc} \hline E & C & T \\ \hline \text{Party} & \text{Kilo} & t_1 \\ \vdots & \vdots & \vdots \\ \text{Party} & \text{Kilo} & t_{1k} \\ \hline \end{array} = \begin{array}{cc} \hline E & C \\ \hline \text{Party} & \text{Kilo} \\ \hline \end{array} \bowtie \begin{array}{cc} \hline E & T \\ \hline \text{Party} & t_1 \\ \vdots & \vdots \\ \text{Party} & t_{1k} \\ \hline \end{array}$$

If (R,Σ) is in *ℓ-JENF*, then ℓ is a *level of join efficiency that* (R,Σ) *permits*. Otherwise, ℓ is a *level of join efficiency that* (R,Σ) *prevents*. If there is no ℓ for which (R,Σ) is in *ℓ-JENF*, no a priori upper bound exists for the level of join efficiency that (R,Σ) permits. In this case, $\ell := \infty$.

5.4 Justification

It turns out for every $\ell \in \mathbb{N}^{\infty}_{\geq 1}$ that ℓ-BCNF coincides with ℓ-RFNF (ℓ-UINF) and with ℓ-JENF. Hence, schemata in ℓ-BCNF capture instances that are i) free from any ℓ-redundant data value occurrences (ℓ-update inefficiencies), and ii) permit level ℓ-join efficiency.

Theorem 6 *For all (R,Σ) and all $\ell \in \mathbb{N}^{\infty}_{\geq 1}$, the following are equivalent: (1) (R,Σ) is in ℓ-RFNF (ℓ-UINF), (2) (R,Σ) is in ℓ-JENF, and (3) (R,Σ) is in ℓ-BCNF.*

Proof (1) $\Rightarrow$ (2) We proceed by contraposition, and therefore assume that (R,Σ) is not in ℓ-JENF. We will show that (R,Σ) is not in ℓ-RFNF. Since (R,Σ) is not in ℓ-JENF, the join efficiency of (R,Σ) is larger than ℓ. That means there is some relation r over R, some tuple $t_0 \in r$, some non-trivial FD $X \to Y \in \Sigma^+_{\mathfrak{C}}$, and some pairwise different tuples $t_0, t_1, \ldots, t_\ell \in r$ such that $t_0(X) = t_1(X) \cdots t_\ell(X)$ holds.

We show that there is some $A \in R$ such that the data value occurrence $t_1(A) \in r(A)$ is ℓ-redundant with respect to $X \to A \in \Sigma^+_{\mathfrak{C}}$. Indeed, since $X \to Y$ is non-trivial we have $Y - X \neq \emptyset$, and we let A denote an arbitrary attribute from $Y - X$. Now, for every ℓ-transaction $\{t'_1,\ldots,t'_\ell\}$ of $\{t_1,\ldots,t_\ell\}$ for A, the resulting relation $(r - \{t_1,\ldots,t_\ell\}) \cup \{t'_1,\ldots,t'_\ell\}$ will violate $X \to A$. Indeed, for every ℓ-transaction $\{t'_1,\ldots,t'_\ell\}$ of $\{t_1,\ldots,t_\ell\}$ for A, there is some $i \in \{1,\ldots,\ell\}$ such that $t'_i(A) \neq t_i(A)$. Hence, $t_0(X) = t'_i(X)$ since $A \notin X$, and $t_0(A) = t_i(A) \neq t'_i(A)$. This shows, that (R,Σ) is not in ℓ-RFNF.

(2) $\Rightarrow$ (3) We proceed by contraposition, and therefore assume that (R,Σ) is not in ℓ-BCNF. We will show that (R,Σ) is not in ℓ-JENF. Since (R,Σ) is not in ℓ-BCNF, there is some $X \to Y \in \Sigma^+_{\mathfrak{C}}$ where $Y \not\subseteq X$ such that $\mathit{card}(X) \leq \ell \notin \Sigma^+_{\mathfrak{C}}$. In particular, Σ does not imply $\mathit{card}(X) \leq \ell$.

We construct a relation $r := \{t_0,\ldots,t_\ell\}$ over R for $i = 0\ldots,\ell$ and all $A \in R$ as follows:

$$t_i(A) := \begin{cases} 0 & \text{, if } A \in X^+_{\Sigma[\mathrm{FD}]} \\ i & \text{, else} \end{cases} .$$

We show that r satisfies Σ. Let $U \to V \in \Sigma$ and let $t, t' \in r$. If $t(U) = t'(U)$, then $U \subseteq X^+_{\Sigma[\mathrm{FD}]}$. Hence, $X \to U \in \Sigma^+_{\mathfrak{C}}$. Since $U \to V \in \Sigma$, we obtain $X \to V \in \Sigma^+_{\mathfrak{C}}$ by application of the transitivity rule. Consequently, $V \subseteq X^+_{\Sigma[\mathrm{FD}]}$. By construction, $t(V) = t'(V)$, which means that r satisfies $U \to V$. Let $card(U) \leq \ell' \in \Sigma$. Since $card(X) \leq \ell \notin \Sigma^+_{\mathfrak{C}}$ we know that i) $\ell' > \ell$ or ii) $U \not\subseteq X^+_{\Sigma[\mathrm{FD}]}$ by Theorem 1. In case i), the $\ell + 1$-tuple relation r satisfies $card(U) \leq \ell'$. In case ii), the relation satisfies $card(U) \leq 1$ by construction, and therefore also $card(U) \leq \ell'$. We have shown that r satisfies Σ. However, r does not satisfy $card(X) \leq \ell$ since there are $\ell + 1$ different tuples in r that have the same projection on $X \subseteq X^+_{\Sigma[\mathrm{FD}]}$. Indeed, there are $\ell + 1$ different tuples because there is some attribute in $R - X^+_{\Sigma[\mathrm{FD}]}$, since otherwise $card(X) \leq 1$ would be in $\Sigma^+_{\mathfrak{C}}$ and therefore also $card(X) \leq \ell$, which would contradict our assumption.

In summary, we have generated a relation r over R that satisfies Σ such that for tuple $t_0 \in r$ and FD $X \to Y \in \Sigma^+_{\mathfrak{C}}$ there are some tuples $t_1, \ldots, t_\ell \in r$ such that $t_0(X) = t_1(X) \cdots = t_\ell(X)$ holds and $t_0, \ldots, t_\ell$ are pairwise different tuples. Consequently, the join strength of t_0 is at least $\ell + 1$, which means that the join efficiency of (R, Σ) is larger than ℓ. That means that (R, Σ) is not in ℓ-JENF.

(3) $\Rightarrow$ (1) We proceed by contraposition, and therefore assume that (R, Σ) is not in ℓ-RFNF. We will show that (R, Σ) is not in ℓ-BCNF. Since (R, Σ) is not in ℓ-RFNF, there is some relation r over R that satisfies Σ, some $\sigma \in \Sigma$, and some attribute $A \in R$ for which there is some data value occurrence $v \in r(A)$ that is ℓ-redundant with respect to σ. Hence, there are ℓ distinct tuples $t_1, \ldots, t_\ell \in r$ such that $t_1(A) = \ldots = t_\ell(A) = v$ and every ℓ-transaction $t'_1, \ldots, t'_\ell$ of $t_1, \ldots, t_\ell$ for A results in a relation $r' = (r - \{t_1, \ldots, t_\ell\}) \cup \{t'_1, \ldots, t'_\ell\}$ that violates σ. It follows that $\sigma = X \to Y \in \Sigma$ such that $A \in Y - X$. Indeed, σ cannot be a cardinality constraint since we can easily pick an ℓ-transaction that would result in a relation that satisfies σ otherwise (by simply introducing a fresh domain value for one of the tuples). Hence, there must be at least $\ell+1$ different $t_0, t_1, \ldots, t_\ell \in r$ such that $t_i(XA) = t_0(XA)$ for $i = 1, \ldots, \ell$. Indeed, if there were at most ℓ different $t_0, t_1, \ldots, t_{\ell-1} \in r$ such that $t_i(XA) = t_0(XA)$ for $i = 1, \ldots, \ell-1$, then the ℓ-transaction $\{t'_0, t'_1, \ldots, t'_{\ell-1}\}$ of $\{t_0, t_1, \ldots, t_{\ell-1}\}$ for A where $t'_0(A) = t'_1(A) = \ldots = t'_{\ell-1}(A)$ would result in a relation $(r - \{t_0, \ldots, t_{\ell-1}\}) \cup \{t'_0, t'_1, \ldots, t'_{\ell-1}\}$ that satisfies $X \to A$. Consequently, r does not satisfy $card(X) \leq \ell$ since there are $\ell + 1$ different tuples in r that have the same projection on X. It follows that there is some non-trivial $X \to Y \in \Sigma^+_{\mathfrak{C}}$ and $card(X) \leq \ell \notin \Sigma^+_{\mathfrak{C}}$. Hence, (R, Σ) is not in ℓ-BCNF. ∎

The proof of Theorem 6 - in particular - constructs for every schema that is not in ℓ-BCNF an instance that features some ℓ-redundant data value occurrence (ℓ-update inefficiency), and a level $\ell + 1$-join strength. Such a construction can be used in practice to automatically generate relations which exemplify the properties of a schema.

$(\textsc{Hap}, \Sigma)$ from Example 1 is in ℓ_2-BCNF but not in $\ell_2 - 1$-BCNF. The CC $card(V) \leq \ell_2 - 1$ is not implied by Σ but $V \to C$ is an FD in Σ. Based on this FD we construct a relation with ℓ_2 tuples $t'_1, \ldots, t'_{\ell_2}$ that all have matching values on columns in $V^+_{\Sigma[\mathrm{FD}]} = VC$, and unique values on all other columns. The relation may consist of the last ℓ_2 tuples in r of Table 1. It prevents level ℓ_2 data redundancy and update inefficiency, and permits level ℓ_2 join efficiency.

6 Schema Design Algorithms

After defining the update inefficiency and join efficiency of schema decompositions we propose three algorithms for logical schema design. We also illustrate how our concepts inform view selection already during logical design.

6.1 Assessing Decompositions

We can assess the quality of a schema decomposition $\mathcal{D}$ for both updates and joins. The join efficiency of $\mathcal{D}$ results from only those FDs preserved by $\mathcal{D}$. These may have been transformed into keys or not. We define the set of *join-supportive attribute subsets* as $JS_{\mathcal{D}}^{(R,\Sigma)} = \{S : X_k \mid \exists S \in \mathcal{D} \exists X \to Y \in \Sigma[S], \Sigma \models X \to S\} \cup \{S : X \mid \exists S \in \mathcal{D} \exists X \to Y \in \Sigma[S], \Sigma \not\models X \to S\}$. Next, the join strength of a join-supportive attribute set is given by the minimal upper bounds for any CC that applies to it. Hence, for (R, Σ) and $X \subseteq R$, $\ell_X := \min\{\ell \mid \Sigma \models \mathit{card}(X) \leq \ell\}$ is the minimum level of data redundancy for X. Now we define the level of join efficiency for $\mathcal{D}$ as the maximum among all minimum levels of data redundancy for a join-supportive attribute set, that is, $\ell_{\mathcal{D}}^J := \sup\{\ell_X \mid S : X \in JS_{\mathcal{D}}^{(R,\Sigma)}\} \cup \{1 \mid S : X_k \in JS_{\mathcal{D}}^{(R,\Sigma)}\}$. In essence, FDs $X \to Y$ transformed into keys X contribute level 1, and other FDs contribute level ℓ_X. Defining $\ell_{\mathcal{D}}^J$ as supremum means it is 1 when Σ contains no FDs. In addition, the *total level of join efficiency for* $\mathcal{D}$ is the sum of the levels: $\ell_{\mathcal{D}}^{J,\text{total}} = \sum_{S:X \in JS_{\mathcal{D}}^{(R,\Sigma)}} \ell_X + \sum_{S:X_k \in JS_{\mathcal{D}}^{(R,\Sigma)}} 1$.

The update inefficiency of $\mathcal{D}$ is determined by all FDs of the input. In particular, any lost FD will need to be enforced by joining elements of $\mathcal{D}$ whenever updates occur. A BCNF decomposition of our running example into R_2 and R_3 results in a loss of the FD $E \to C$. As the instances r_2 and r_3 in Table 2 illustrate, any update of a C-value (such as *Kilo*) in r_2 needs to be propagated consistently for all its occurrences in r_2 with the same event (eg. *Party*), so the FD $E \to C$ still holds (on the join of R_2 and R_3). The example illustrates how lost FDs cause data redundancy on the join of schemata. This is the reason why they need to be enforced. Hence, we define the set of *update-critical attribute subsets* as $UC_{\mathcal{D}}^{(R,\Sigma)} = JS_{\mathcal{D}}^{(R,\Sigma)} \cup \{X \subseteq R \mid \exists X \to Y \in (\Sigma - (\cup_{S \in \mathcal{D}} \Sigma[S])^+)\}$. Now we define the level of update inefficiency for $\mathcal{D}$ as the maximum among all minimum levels of data redundancy for an update-critical attribute set, that is, $\ell_{\mathcal{D}}^U := \sup\{\ell_X \mid S : X \in UC_{\mathcal{D}}^{(R,\Sigma)}\} \cup \{1 \mid S : X_k \in UC_{\mathcal{D}}^{(R,\Sigma)}\}$. Alternatively, we can also sum up to derive the *total level of update inefficiency for* $\mathcal{D}$ as follows: $\ell_{\mathcal{D}}^{U,\text{total}} = \sum_{S:X \in UC_{\mathcal{D}}^{(R,\Sigma)}} \ell_X + \sum_{S:X_k \in UC_{\mathcal{D}}^{(R,\Sigma)}} 1$.

With only FDs, BCNF decomposition or 3NF synthesis may lead to the optimal case of an FD-preserving BCNF decomposition with a join efficiency and update inefficiency of 1. Otherwise, some FD is lost or not a key. Hence, every non-optimal case results in unbounded levels of update inefficiency and join efficiency. With CCs, our framework can measure update inefficiency and join efficiency in all cases.

We will now propose different algorithms for schema design, and illustrate their achievements later by experiments.

Algorithm 2 $\textsc{Opt}(R, \Sigma)$

Require: (R, Σ) with CC/FD set Σ over schema R
Ensure: Lossless, FD-preserving $\ell_{\mathcal{D}}$-BCNF decomposition $\mathcal{D}$ of (R, Σ) with minimum $\ell_{\mathcal{D}}$ $(= \ell_{\mathcal{D}}^U = \ell_{\mathcal{D}}^J)$

1: Choose an atomic cover Σ_a of Σ [25];
2: **for all** $X \to A \in \Sigma_a$ **do**
3: $\quad \Sigma[X \to A] \leftarrow \emptyset$
4: $\quad$ **for all** $Y \to B \in \Sigma_a (YB \subseteq XA \wedge XA \not\subseteq Y^+_{\Sigma[FD]})$ **do**
5: $\quad\quad \ell_Y \leftarrow \min\{\ell \mid \Sigma \models \mathit{card}(Y) \leq \ell\}$
6: $\quad\quad \Sigma[X \to A] \leftarrow \Sigma[X \to A] \cup \{(Y \to B, \ell_Y)\}$
7: $\quad \ell_{X \to A} \leftarrow \sup\{\ell_Y \mid Y \to B \in \Sigma[X \to A]\}$
8: $\mathcal{D} \leftarrow \emptyset$;
9: **for all** $X \to A \in \Sigma_a$ in decreasing order of $\ell_{X \to A}$ **do**
10: $\quad$ **if** $\Sigma_a - \{X \to A\} \models X \to A$ **then**
11: $\quad\quad \Sigma_a \leftarrow \Sigma_a - \{X \to A\}$
12: $\quad$ **else**
13: $\quad\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(XA, \Sigma_a[XA])\}$
14: Remove all $(S, \Sigma_a[S]) \in \mathcal{D}$ if $\exists (S', \Sigma_a[S']) \in \mathcal{D} (S \subseteq S')$
15: **if** there is no $(R', \Sigma') \in \mathcal{D}$ where $R' \to R \in \Sigma^+_{\mathfrak{L}}$ **then**
16: $\quad$ Choose a minimal key K for R with respect to Σ
17: $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{(K, \Sigma_a[K])\}$
18: **Return**$(\mathcal{D}, \ell_{\mathcal{D}})$

6.2 FD-preservation

Preserving all FDs guarantees that the levels of join efficiency and update inefficiency coincide. Otherwise, the level of update inefficiency may exceed that of join efficiency (more pain than gain). Consequently, we aim at minimizing the level of update inefficiency across all FD-preserving decompositions. Hence, we start with the unique *atomic cover* [25] (all L-reduced FDs with singleton attribute on the right-hand side), then compute for all FDs σ the maximum ℓ_σ that is associated with all the non-key FDs subsumed by σ, and then synthesize the final decomposition with schemata generated by FDs σ in decreasing order of their ℓ_σ, unless such a σ is implied by the remaining FDs. Due to Theorem 5, Algorithm 2 is worst-case exponential.

$(\textsc{Hap}, \Sigma)$ from Example 1 is already in 3NF. As no FD-preserving BCNF decomposition exists, classical normalization stops here. Using $(\textsc{Hap}, \Sigma)$ as input for Algorithm 2, we obtain $\Sigma_a = \Sigma \cup \{CT \to E\}$ (line 1). Hence, $\ell_{CT \to V} = 1m$ and $\ell_{CT \to E} = 1k$, and for other $X \to A \in \Sigma_a$, $\ell_{X \to A} = 1$ (line 7). However, $CT \to V$ is implied by $\Sigma_a - \{CT \to V\}$ (line 10). This step is critical, since $CT \to V$ has effectively been replaced by $CT \to E$, which also means that the *1m*-level data redundancy caused by $CT \to V$ has been reduced to the *1k*-level data redundancy caused by $CT \to E$. Furthermore, $R_1 \subseteq R_4$ (line 13), so Algorithm 2 returns the $1k$-BCNF schema $\mathcal{D}_1 = \{(R_2, \Sigma_2), (R_3, \Sigma_3), (R_4, \Sigma_4)\}$ from the intro. Classical normalization would generate neither $\mathcal{D}_1$ nor $\mathcal{D}_2$. Even if it did, $\mathcal{D}_1$, $\mathcal{D}_2$ and $(\textsc{Hap}, \Sigma)$ would be assessed as equal in quality.

Algorithm 3 $\textsc{Greed}(R, \Sigma)$

Require: (R, Σ) with CC/FD set Σ over R, $\ell \in \mathbb{N}_{\geq 1}$
Ensure: Lossless $\ell^U_{\mathcal{D}}$-BCNF decomposition $\mathcal{D}$ of (R, Σ)
1: **if** (R, Σ) is in ℓ-BCNF **then**
2: $\quad \mathcal{D} \leftarrow \{(R, \Sigma)\}$
3: **else**
4: $\quad \ell_{\max} \leftarrow \max\{\ell_X \mid X \rightarrow Y \in \Sigma, Y \not\subseteq X, \Sigma \not\models X \rightarrow R\}$
5: $\quad$ Pick $X \rightarrow Y \in \Sigma$ $(Y \not\subseteq X \wedge \Sigma \not\models X \rightarrow R \wedge \ell_X = \ell_{\max})$
6: $\quad R_1 \leftarrow XY$; $R_2 \leftarrow X(R - XY)$
7: $\quad \mathcal{D} \leftarrow \textsc{Greed}(R_1, \Sigma[R_1]) \cup \textsc{Greed}(R_2, \Sigma[R_2])$
8: **Return**$(\mathcal{D}, \ell^U_{\mathcal{D}}, \ell^J_{\mathcal{D}})$

We may apply Algorithm 2 to a partial input of Σ. For example, excluding FDs which potentially cause high update inefficiency but are unlikely violated by actual updates, may result in higher join efficiency. Similarly, one may include all FDs $X \rightarrow Y$ where ℓ_X meets a given threshold.

6.3 Greedy Decomposition

Typically, FD-preservation requires many output tables, which is linked to low join efficiency. We reposition classical BCNF decomposition as a greedy algorithm for reducing $\ell^U_{\mathcal{D}}$ using fewer tables. Selecting non-key FDs that drive classical BCNF decomposition is arbitrary. With CCs at hand, we select FDs whose level of data redundancy is maximal (for example, ∞). Algorithm 3 shows this strategy. Targeting higher join efficiency, we may use a higher level ℓ as input. If unspecified, the default target is $\ell = 1$. The algorithm proceeds with some best remaining FD (lines 4-5), as long as some local schema is not in ℓ-BCNF (line 1). The output returns schema $\mathcal{D}$, the level $\ell^U_{\mathcal{D}}$ of update inefficiency it prevents, and the level $\ell^J_{\mathcal{D}}$ of join efficiency it permits. The penalty for fewer output tables is the lack of control over these levels due to lost FDs. FDs $X \rightarrow A$ where $\ell_X = \infty$ have high priority. With no CCs given, we have the classical BCNF decomposition case where $\ell^U_{\mathcal{D}} = \infty = \ell^J_{\mathcal{D}}$, unless $\mathcal{D}$ is FD-preserving and $\ell^U_{\mathcal{D}} = 1 = \ell^J_{\mathcal{D}}$.

We apply Algorithm 3 to $(\textsc{Hap}, \Sigma)$ from Example 1. As $V \rightarrow C \in \Sigma$ causes $\ell_{\max} = 1m$ (line 4), we obtain $\mathcal{D}_g = \{(R_2 = VC, \Sigma_2 = \{V \rightarrow C\}), (R_3 = EVT, \Sigma_3 = \{TV \rightarrow E, ET \rightarrow V\})\}$ (line 7). We lost $E \rightarrow C$ and $CT \rightarrow V$, so $\ell^U_{\mathcal{D}_g} = 1k$. We create assertion checks to enforce the lost FDs $E \rightarrow C$ and $CT \rightarrow V$ on the join $R_2 \bowtie R_3$:

CREATE ASSERTION LostFDEtoC CHECK(NOT EXISTS (
SELECT $R_3.E$ FROM R_2, R_3 WHERE $R_2.V = R_3.V$
GROUP BY $R_3.E$ HAVING COUNT($R_2.C$)> 1));
CREATE ASSERTION LostFDCTtoV CHECK(NOT EXISTS (
SELECT $R_2.C, R_3.T$ FROM R_2, R_3 WHERE $R_2.V = R_3.V$
GROUP BY $R_2.C, R_3.T$ HAVING COUNT($R_2.V$)> 1));

$\mathcal{D}_g$ achieves the same level of update inefficiency as $\mathcal{D}_1$ with fewer schemata. This

Algorithm 4 HYBRID(R, Σ)

Require: (R, Σ) with CC/FD set Σ over schema R
Ensure: Lossless $\ell^U_{\mathcal{D}}$-BCNF decomposition $\mathcal{D}$ of (R, Σ)

```
1: D ← Opt(R, Σ)
2: for all S ∈ D do
3:     S = XA for some X → A ∈ Σ_a
4:     for all Y → B ∈ Σ_a(YB ⊆ XA ∧ XA ⊄ Y^+_{Σ[FD]}) do
5:         if ℓ_Y > ℓ_X then
6:             S_1 = YB; S_2 = Y(XA − B)
7:             D ← (D − {(S, Σ_a[S])}) ∪ {(S_1, Σ_a[S_1]), (S_2, Σ_a[S_2])}
8: Remove any non-maximal schema from D
9: Return(D, ℓ^U_D, ℓ^J_D)
```

comes at the price of lost FDs that require assertion checks and smaller join efficiency $\ell^J_{\mathcal{D}_g} = 1$.

Our algorithms can be adapted when more information becomes available. For example, while $\mathcal{D}_g$ was driven by the FD $V \to C$ due to *card*$(V) \leq 1m$, there may not be updates of C-values based on V-values but frequent updates of C-values based on E-values. Hence, we could prioritize the FD $E \to C$ instead to obtain $\mathcal{D}'_g = \{(R_1 = EC, \Sigma_1 = \{E \to C\}), (R_3, \Sigma_3)\}$ with lost FDs $V \to C$ and $CT \to V$.

Just like FD-preserving BCNF decompositions cannot always be achieved, ℓ-BCNF cannot always be achieved for a given ℓ. Adapting a classical example [3], consider $R = \{c(ity), s(treet), z(ip), A\}$ where Σ consist of $sc \to z$ and $z \to c$, and the CCs *card*$(c, s) \leq \ell + 1$ and *card*$(z) \leq \ell + 1$. If $\{c, s, z\}$ is not in the output $\mathcal{D}$, we lose $sc \to z$, and $\ell^U_{\mathcal{D}} = \ell + 1$. If $\{c, s, z\}$ is in $\mathcal{D}$, we will have $z \to c$ on $\{c, s, z\}$, and hence $\ell^U_{\mathcal{D}} = \ell + 1$. Thus, no output can be in ℓ-BCNF.

6.4 Hybrid algorithm

We combine the two previous strategies to the hybrid Algorithm 4. Here, Algorithm 2 is applied first (line 1) to obtain the smallest possible $\ell_{\mathcal{D}}$ among all FD-preserving decompositions. Next we check if for any of the output schemata XA (lines 2-3) that are in 3NF but not in BCNF (line 4), any non-key FD $Y \to B$ on XA satisfies $\ell_Y > \ell_X$ (line 5). In this case, we trade the preservation of $X \to A$ for the lower level ℓ_X of update inefficiency by decomposing with the FD $Y \to B$ (lines 6/7).

On input (HAP, Σ) from Example 1, Algorithm 4 returns decomposition $\mathcal{D}_1$ as output of Algorithm 2 (line 1). In particular, $S = R_4 = CET$ is in 3NF but not in BCNF for $\Sigma_a[S] = \Sigma_4 = \{CT \to E, E \to C\}$. That is, the FD $E \to C \in \Sigma_a$ is critical on the schema $S = R_4 = CET$ (lines 3 and 4). Since $\ell_E = 1k > 1 = \ell_{CT}$ (line 5), we decompose along the critical FD $E \to C$ to replace R_4 by $R_6 = EC$ and $R_7 = ET$ (lines 6 and 7). However, $R_6 = R_1$ and $R_7 \subseteq R_3$. Hence, the output is $\mathcal{D}_h = \{(R_1, \Sigma_1), (R_2, \Sigma_2), (R_3, \Sigma_3)\}$ with $\ell^U_{\mathcal{D}_h} = 1 = \ell^J_{\mathcal{D}_h}$ and lost FD $CT \to V$. The assertion check `LostFDCTtoV` enforces the FD $CT \to V$ on the join $R_2 \bowtie R_3$.

Table 4: Schema Designs for Running Example

Method	*Schema*	*Lost FDs*	ℓ^U	ℓ^J
Opt	$\mathcal{D}_1$: $R_2 = CV$ with $V \to C$	none	1k	1k
	$R_3 = ETV$ with $TV \to E$, $ET \to V$	Materialized view:		
	$R_4 = CET$ with $CT \to E$, $E \to C$	$\mathcal{V}_{\mathcal{D}_1} = \pi_{VCT}(R_2 \bowtie R_3)$		
N/A	$\mathcal{D}_2$: $R_1 = EC$ with $E \to C$	none	1m	1m
	$R_3 = ETV$ with $TV \to E$, $ET \to V$	Materialized view:		
	$R_5 = CTV$ with $CT \to V$, $V \to C$	$\mathcal{V}_{\mathcal{D}_2} = \pi_{ECT}(R_1 \bowtie R_3)$		
Greed	$\mathcal{D}_g$: $R_2 = CV$ with $V \to C$	$E \to C$	1k	1
	$R_3 = ETV$ with $TV \to E$, $ET \to V$	$CT \to V$		
		$\mathcal{V}_{\mathcal{D}_g} = \pi_{ECT}(R_2 \bowtie R_3)$		
Hybrid	$\mathcal{D}_h$: $R_1 = EC$ with $E \to C$	$CT \to V$	1	1
	$R_2 = CV$ with $V \to C$	Materialized view:		
	$R_3 = ETV$ with $TV \to E$, $ET \to V$	$\mathcal{V}_{\mathcal{D}_h} = \pi_{VCT}(R_1 \bowtie R_3)$		

Table 4 summarizes the schemata that our algorithms return for our example, and also $\mathcal{D}_2$, including their properties.

6.5 Materialized Views

When operational, frequent access patterns emerge on the database. Adding materialized views can help accelerate queries but will occupy additional storage space and time to maintain data consistency. Hence, view selection should be informed by quantifying join support and maintenance costs. We define the (total) level of join efficiency and update inefficiency of a given view $\mathcal{V}$ (namely, a set of attributes) as $\ell_{\mathcal{V}}^{J(\text{, total})}$ and $\ell_{\mathcal{V}}^{U(\text{, total})}$, respectively.

For our hybrid schema $\mathcal{D}_h$ as example, a frequent query may ask at what times companies work on a venue. Hence, we may introduce the following materialized view $\mathcal{V}$.

CREATE MATERIALIZED VIEW $\mathcal{V}$ AS (
SELECT $R_2.V, R_2.C, R_3.T$ FROM R_2, R_3 WHERE $R_2.V = R_3.V$);

In this case, the assertion check `LostFDCTtoV` can directly be enforced on the view instead.

CREATE ASSERTION LostFDCTtoV CHECK(NOT EXISTS (SELECT
C, T FROM $\mathcal{V}$ GROUP BY C, T HAVING COUNT(V)>1));

Due to $V \to C$, $CT \to V$ and $card(V) \leq 1m$, the view has level $\ell_{\mathcal{V}}^J = 1m = \ell_{\mathcal{V}}^U$ update inefficiency and join efficiency, respectively. Join support is effective, but requires the propagation of a C-update on base table R_2 to up to $1m$ updates on $\mathcal{V}$. Hence, the trade-off we quantify between update inefficiency and join efficiency is also intrinsic to materialized views and should be part of the information for helping with their selection. Table 4 also lists some views for the various schema designs of our running example.

Table 5: Synthetic Experiment 1 (in seconds)

Op	Hap	$\mathcal{D}_1$	$\mathcal{D}_1 + \mathcal{V}_{\mathcal{D}_1}$	$\mathcal{D}_2$	$\mathcal{D}_2 + \mathcal{V}_{\mathcal{D}_2}$	$\mathcal{D}_g$	$\mathcal{D}_g + \mathcal{V}_{\mathcal{D}_g}$	$\mathcal{D}_h$	$\mathcal{D}_h + \mathcal{V}_{\mathcal{D}_h}$
u_1	0.93	0.91	0.91	0.59	1.03	1.62	1.81	0.59	0.59
u_2	45.75	0.23	45.33	44.81	44.81	0.20	0.20	0.21	83.82
q_1	0.011	0.009	0.009	0.17	0.012	1.32	0.009	0.14	0.14
q_2	3.39	4.11	3.47	2.76	2.76	4.09	4.09	4.32	3.44

7 Experiments

Our experiments analyze the properties of our normal forms and how these translate into the performance of updates and joins over instances of the schema. We implemented our algorithms in Visual C++. Experiments were done on an Intel Xeon W-2123, 3.6 GHz, 256GB, Windows 10 PC, with the 2017 SQL Server Community Edition.

Qualitative study. Firstly, we analyze the performance of two updates and two joins on synthetic instances over the schema designs for our running example.

Our first instance over (Hap, Σ) consists of 1,001,000 tuples. There are $1k$ tuples with matching values on E and C, and $1m$ tuples with matching values on V and C. The other values are unique in their columns. The instance is isomorphic to instance r in Table 1 with $\ell_1 = 1k$ and $\ell_2 = 1m$. To illustrate the impact of FDs on updates and joins, the numbers of redundant values they cause in the instance coincides with their levels of data redundancy on the schema (ℓ_1 and ℓ_2).

We consider four operations affected by our FDs. Update u_1 (u_2) replaces all occurrences of a C-value associated with a given E-value (V-value, respectively). Query q_1 (q_2) returns all CTE (CTV, respectively) combinations. Each operation is run 100 times on each design, and the average value reported.

Updates are always propagated to views from base tables. As $E \to C$ is lost on $\mathcal{D}_g$, u_1 updates C-values on $R_2 \bowtie R_3$ and projects them onto R_2. Unless queries q_1 or q_2 are equivalent to a view, the table below lists their final projection in the left column q (same on every design), and the indices of tables joined for each design before the projection (no join for single index), such as $q_1 = \pi_{ECT}(R_1 \bowtie R_3)$ over $\mathcal{D}_2$.

q	$\mathcal{D}_1$	$\mathcal{D}_1 + \mathcal{V}_{\mathcal{D}_1}$	$\mathcal{D}_2$	$\mathcal{D}_2 + \mathcal{V}_{\mathcal{D}_2}$	$\mathcal{D}_g$	$\mathcal{D}_g + \mathcal{V}_{\mathcal{D}_g}$	$\mathcal{D}_h$	$\mathcal{D}_h + \mathcal{V}_{\mathcal{D}_h}$
$q_1 = \pi_{ECT}$	4	4	1,3	$\mathcal{V}_{\mathcal{D}_2}$	2,3	$\mathcal{V}_{\mathcal{D}_g}$	1,3	1,3
$q_2 = \pi_{CTV}$	2,3	$\mathcal{V}_{\mathcal{D}_1}$	5	5	2,3	2,3	1,3	$\mathcal{V}_{\mathcal{D}_h}$

On the instance over 3NF schema (Hap, Σ), q_1 and q_2 are simple projections and do not require joins. For u_1, $1k$ occurrences of a redundant C-value are updated, and $1m$ occurrences of a redundant C-value are updated for u_2, due to the non-key FDs $E \to C$ and $V \to C$, respectively.

Table 5 shows the times (in seconds) for the operations on the instance projected onto the schemata of Table 4.

Specific updates and queries are best supported by different designs. Notably, the level of data redundancy for an FD translates proportionally into the performance for updates and joins affected by the FD. This also applies to the views.

Table 6: Synthetic Experiment 2 (in milliseconds)

Op	HAP	$\mathcal{D}_1$	$\mathcal{D}_1 + \mathcal{V}_{\mathcal{D}_1}$	$\mathcal{D}_2$	$\mathcal{D}_2 + \mathcal{V}_{\mathcal{D}_2}$	$\mathcal{D}_g$	$\mathcal{D}_g + \mathcal{V}_{\mathcal{D}_g}$	$\mathcal{D}_h$	$\mathcal{D}_h + \mathcal{V}_{\mathcal{D}_h}$
u_1	58.2	55.8	55.8	3.7	60.7	82.3	170.2	3.7	3.7
u_2	10	4.4	12.7	9.8	9.8	4.9	4.9	4.7	18.9
q_1	8.3	7.5	7.5	11.1	10.6	236	4	11.6	11.6
q_2	5.6	14.6	5.7	5.3	5.3	14.5	14.5	14.1	7.4

Operations u_2 and q_1 are fast on $\mathcal{D}_1$ with R_4 and key V on R_2, but slow on $\mathcal{D}_2$. Symmetrically, u_1 and q_2 are fast on $\mathcal{D}_2$ with R_5 and key E on R_1, but slow on $\mathcal{D}_1$. For each operation, the difference in performance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is proportional to the level of data redundancy caused by the FD that affects the operation. For example, the FD $V \to C$ with $\ell_V = 1m$ causes a significant difference between performing u_2 (44.58s difference) on $\mathcal{D}_1$ and $\mathcal{D}_2$, and between performing q_2 (1.35s difference) on $\mathcal{D}_1$ and $\mathcal{D}_2$. In comparison, the FD $E \to C$ with $\ell_V = 1k$ causes a much smaller difference between performing u_1 (0.32s) on $\mathcal{D}_1$ and $\mathcal{D}_2$, and between performing q_1 (0.161s) on $\mathcal{D}_1$ and $\mathcal{D}_2$.

Relatively to $\ell_{\mathcal{V}_{\mathcal{D}_1}} = 1m$, the speed up of q_2 and slow down of u_2 by $\mathcal{V}_{\mathcal{D}_1}$ are large. Relatively to $\ell_{\mathcal{V}_{\mathcal{D}_2}} = 1k$, the speed up of q_1 and slow down of u_1 by $\mathcal{V}_{\mathcal{D}_2}$ are smaller.

As expected, $\mathcal{D}_g$ performs well on u_2 but not so much on the other operations. $\mathcal{V}_{\mathcal{D}_g}$ speeds up q_1 with small penalty for u_1, but which had poor performance already.

$\mathcal{D}_h$ performs well for both u_1 and u_2, and quite well for q_1. $\mathcal{V}_{\mathcal{D}_h}$ speeds up q_2 but slows down u_2 significantly.

Table 6 shows our results for the second synthetic instance with 1,100 tuples, $\ell_1 = 1k$, and $\ell'_2 = 100 < 1m$. The experiment illustrates that the same combinations of operations are best supported by the same designs, when compared to the first instance. Since the actual cardinality (100) is much smaller than what is permitted ($1m$), processing times for operations with this cardinality are much smaller (in ms).

The choice of a logical design depends on how well it is believed to support the future workload of operations. Our framework quantifies this support by the level of data redundancy that FDs exhibit as part of the design or materialized view, which measures their impact on the performance of specific updates and joins. In contrast, classical logical design may not bring forward some designs and cannot quantify the support for workloads.

Quantitative study. Secondly, we analyze update and join performance over schema designs derived by our algorithms for FDs and CCs we mined from the real-world data sets in the following table. It shows the number of FDs discovered (#FD), the total number of null marker occurrences (#$\perp$), and the number of rows and columns (#R,#C).

Data set	#FDs	#$\perp$	#R	#C
china	918	418580	262920	18
ncvoter	568	3079822	1024000	19

The data sets benchmark algorithms that discover FDs $X \to A$ from data and rank them in decreasing order by the lowest bound ℓ_X such that $card(X) \leq \ell_X$ holds on the data [42, 43, 50]. We regard different occurrences of $\perp$ in the same column as matching

Table 7: Algorithmic Achievements on Real Schemata

measures	Original	Opt	Greed	Hybrid
ℓ^U	$4,097$	15	512	7
ℓ^J	$4,097$	15	1	7
$\ell^{U,\text{total}}$	$5,043$	87	$3,035$	79
$\ell^{J,\text{total}}$	$5,043$	87	6	73
$\vert\mathcal{D}\vert$	1	65	6	66
time (ms)	-	< 1	< 1	< 1

(a) Measures for top-20% FDs of **ncvoter**

measures	Original	Opt	Greed	Hybrid
ℓ^U	$4,889$	56	$4,881$	55
ℓ^J	$4,889$	56	1	50
$\ell^{U,\text{total}}$	$62,492$	327	$54,247$	257
$\ell^{J,\text{total}}$	$62,492$	327	4	165
$\vert\mathcal{D}\vert$	1	145	4	157
time (ms)	-	< 1	< 1	< 1

(b) Measures for top-20% FDs of **china**

domain values ($\bot=\bot$). Our results are very similar otherwise ($\bot\neq\bot$). The discovered CCs and FDs represent patterns on the data, but not necessarily on the domain. Input to our algorithms are only more relevant FDs, namely those ranked in the top-20%. Adding further FDs causes hardly any differences, indicating that FDs with higher levels of data redundancy determine schema designs.

Analysis of update and join efficiency levels. We apply Algorithms 2 (Opt), 3 (Greed), and 4 (Hybrid) to the schemata for *ncvoter* and *china*.

Table 7 shows the levels ℓ^U; ℓ^J for the decompositions $\mathcal{D}$ and the original input, their total levels $\ell^{U,\text{total}}$; $\ell^{J,\text{total}}$, output size $|\mathcal{D}|$, and the runtime.

The reduction in update inefficiency levels is significant for Opt and Hybrid. Greed achieves small reductions due to lost FDs. The join efficiency level for Greed equals 1 since every FD becomes either a key or is lost. Under FD-preservation, Opt always achieves the optimum level ℓ^U, which always equals ℓ^J as all FDs are preserved. Hybrid achieves even lower levels ℓ^U by loss of some FDs and lowering ℓ^J.

The reductions of ℓ^U are achieved by different output sizes: Greed requires few schemata, while Opt achieves the optimum by adding schemata to preserve all FDs, and Hybrid achieves further reductions by even more schemata. All algorithms are fast as the input is an atomic cover [25].

Physical performance. We ran updates and join queries over our original data sets and the outputs of Algorithms 2 and 3 on input sets that included all FDs $X \to Y$ where ℓ_X met a given target level ℓ. Figures 3 and 5 show the results of Algorithm 2, and Figures 4 and 6 those of Algorithm 3. The x-axes always show the target levels ℓ, the number n of output tables and the number of input FDs. The y-axes in Figures 3 and 4 show the runtimes (in *sec*) for a join query, and the runtimes (in *sec*) for updates

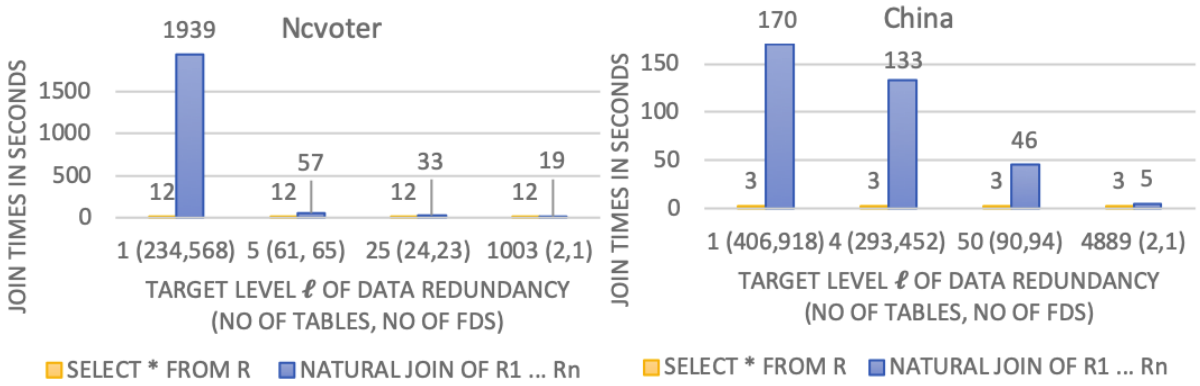


Figure 3: Join performance after using Opt

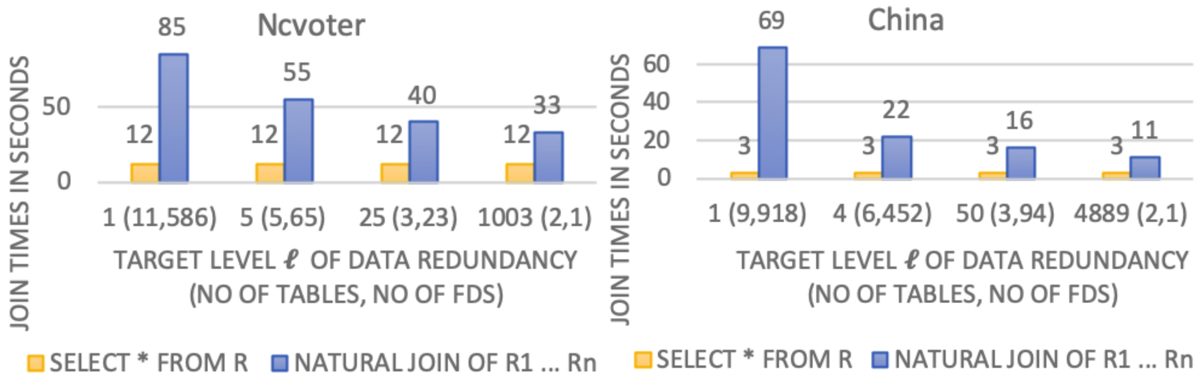


Figure 4: Join performance after using Greed

in Figures 5 and 6.

The join queries are i) a simple SELECT $*$ FROM Hap over the original schema Hap, and ii) a SELECT $*$ FROM R_1 NATURAL JOIN ... NATURAL JOIN R_n over $\{R_1, \ldots, R_n\}$.

Join times increase with lower target levels ℓ. For Opt, join performance becomes poor due to the many output tables required for FD preservation. This is compensated by Greed, but the trade-off is poor update performance for lost FDs. The level ℓ, solely determined by the input integrity constraints, translates into different physical performance degrees for updates and joins on the output of our algorithms. Making these options available is the point of our new framework.

Updates involve changes to all occurrences of some redundant value (recall the update scenario from Section 5). After an FD is transformed into a key, every value occurs at most once, so an update is required for at most one occurrence.

Average update times for keys are about two orders of magnitude lower than those for input FDs on the original data. For preserved, non-key FDs (called critical FDs) the gain is about one order of magnitude (grey bars in Figures 5 and 6, with no data point when all FDs become keys). Lost FDs require more update time than on the original data set, due to necessary joins. With no lost FDs, only keys need to be enforced (in Figure 6 for the largest target level). Figures 5 and 6 show that, by lowering the level

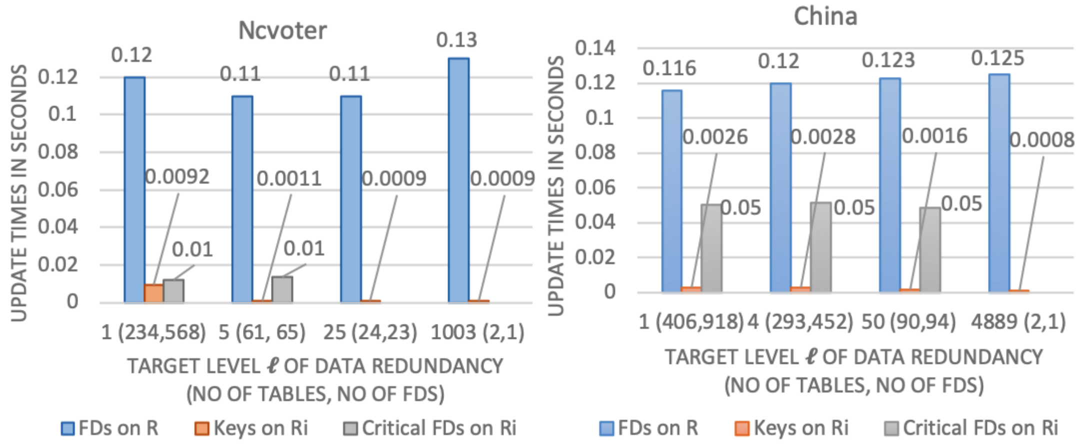


Figure 5: Update performance after using Opt

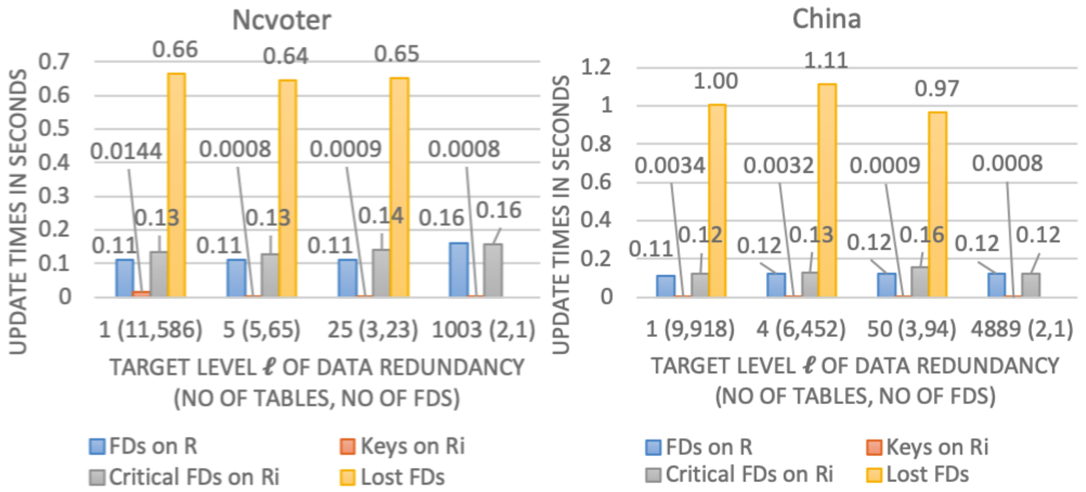


Figure 6: Update performance after using Greed

ℓ, more FDs are converted into keys (for Opt), which improves update efficiency at the expense of introducing more tables. A main message is that schema design is primarily driven by high-ranked FDs, but inclusion of low-ranked FDs means that more FDs need to be preserved during synthesis to lower their update time (by transformation into keys or critical FDs with Opt), or more FDs will be lost during decomposition (Greed).

Comments. Logical schema design is driven by business rules which constrain instances to those that are meaningful to the underlying domain. We have seen how the upper bounds of CCs inform schema design and quantify the support for query and update operations. Our experiments illustrate that smaller bounds mean smaller differences in performance between designs. During logical design the choice of a schema is thus determined by the upper bounds, as these represent the current requirements. When requirements change over time, physical database tuning, such as de-normalization, can lift performance. However, this is only possible once the database is operational, when actual instances can be observed and stable patterns of workloads have emerged. If constraints evolve to the point that a different design is more suitable, then this may

Table 8: Benchmark schemata and data

Data set	#FDs	#⊥	#R	#C
abalone	137	0	4177	9
adult	78	0	48842	14
balance	1	0	625	5
chess	1	0	280560	7
iris	4	0	28056	7
letter	61	0	20000	17
lineitem	3984	0	6001215	16
nursery	1	0	12960	9
breast	46	16	699	11
bridges	142	77	108	13
echo	527	132	132	13
ncvoter	758	2863	1000	19
hepatitis	8250	167	155	20
horse	128727	1605	368	28
plista	178152	23317	1000	63
flight	982631	51938	1000	109
china	918	418580	262920	18
diabetic	40195	192849	101766	30
uniprot512k	3703	3759296	512000	30
pdb	68	2035242	17305799	13
ncvoter1024k	568	3079822	1024000	19

justify migration to a new schema. Ultimately, logical design informs the choice of a schema based on application requirements (evolved or not evolved).

Ranking designs informs the search for a relevant schema, just as ranking websites informs the search for relevant websites. Blindly picking a design with minimum ℓ is similar to using the "I'm feeling lucky" button. This also extends to the selection of materialized views. Our experiments suggest to carefully examine which operations are affected by which FDs. The level of data redundancy caused by the FDs quantifies how much they affect the operations. In particular, if the level of data redundancy caused by some FD is a priori unbounded ($\ell = \infty$), our framework alerts analysts to the fact that requirements analysis may still be incomplete.

8 Additional Experiments

Besides *china* and *ncvoter1024k* (previously called *ncvoter*) we also applied our framework to the real-world data sets and schemata in Table 8. The data set are available for download[2].

We demonstrate the following. 1) Classical Boyce-Codd Normal Form decomposition

[2] https://bit.ly/3c0fE9k

and 3NF synthesis result in schemata that still exhibit significant data redundancy. 2) Legacy data can be profiled using ℓ-BCNF to provide insight into the distribution of data redundancy patterns 3) The logical notion of the level of data redundancy is an indicator for the physical number of redundant values in the data. 4) We illustrate how our decomposition algorithms reduce the level and total levels of update inefficiency and join efficiency. We show the trade-offs between the levels, the number of tables in the decomposition, FD-preservation and the computation time. 5) We illustrate that the top-ranked 30-40% of FDs cause the majority of redundant data values, and that limiting normalization to those results in decompositions with half the number of tables compared to including all FDs. 6) The bounds of CCs guide us in finding fully normalized schema designs that process updates quickly and joins slowly, or fully de-normalized schema designs that process updates slowly and joins quickly, or schemata with different degrees of (de-)normalization that provide a better balance of update and query efficiency.
Set up. We implemented our algorithms in Visual C++. Experiments were done on an Intel Xeon W-2123, 3.6 GHz, 256GB, Windows 10 PC, with the 2017 SQL Server Community Edition. Our experiments are based on FDs mined from the given data sets, and ranked by the number of redundant value occurrences they cause. This was achieved by FD discovery algorithms [43, 50]. For each LHS X of some discovered FD $X \to Y$, we computed the smallest positive integer ℓ_X such that $card(X) \leq \ell_X$ holds on the data set. The discovered CCs and FDs represent exactly the patterns that hold on the given data, but not necessarily of the underlying domain. Since we only illustrate our concepts based on the data we have and not based on the underlying domain, the use of these constraints is fine for our purposes.

8.1 Redundant Values Resulting From Classical Normalization

We use the data sets from Table 8 to illustrate shortcomings of the well-known normalization frameworks of Boyce-Codd and Third normal form.

For each data set we performed 100 Boyce-Codd Normal Form decompositions using different orders of the given FDs. The blue bars of Figure 7 show for each data set the average ratio of the number of redundant data value occurrences caused by FDs that were not preserved during the decomposition relative to the number caused by all given FDs. The orange bars indicate the average ratio of the number of redundant data values caused by FDs that were preserved relative to the number caused by all given FDs. The two ratios are not complements of one another as some occurrences are caused by preserved FDs and by lost FDs. In a strict sense such occurrences are not eliminated, but it is just interesting to observe.

Figure 8 shows the ratio of redundant data value occurrences exhibited after 3NF synthesis using canonical covers. It illustrates that 3NF synthesis removes all or nearly all data redundancies on some data sets (adult, breast, iris, letter, pdb), removes reasonable numbers of data redundancies (abalone, bridges, echo, ncvoter), but also introduces many additional data redundancies on other data sets (china, diabetic, flight, hepatitis, horse, plista).

The experiments illustrate that significant numbers of redundant data value occurrences remain after classical normalization approaches are applied. However, the main

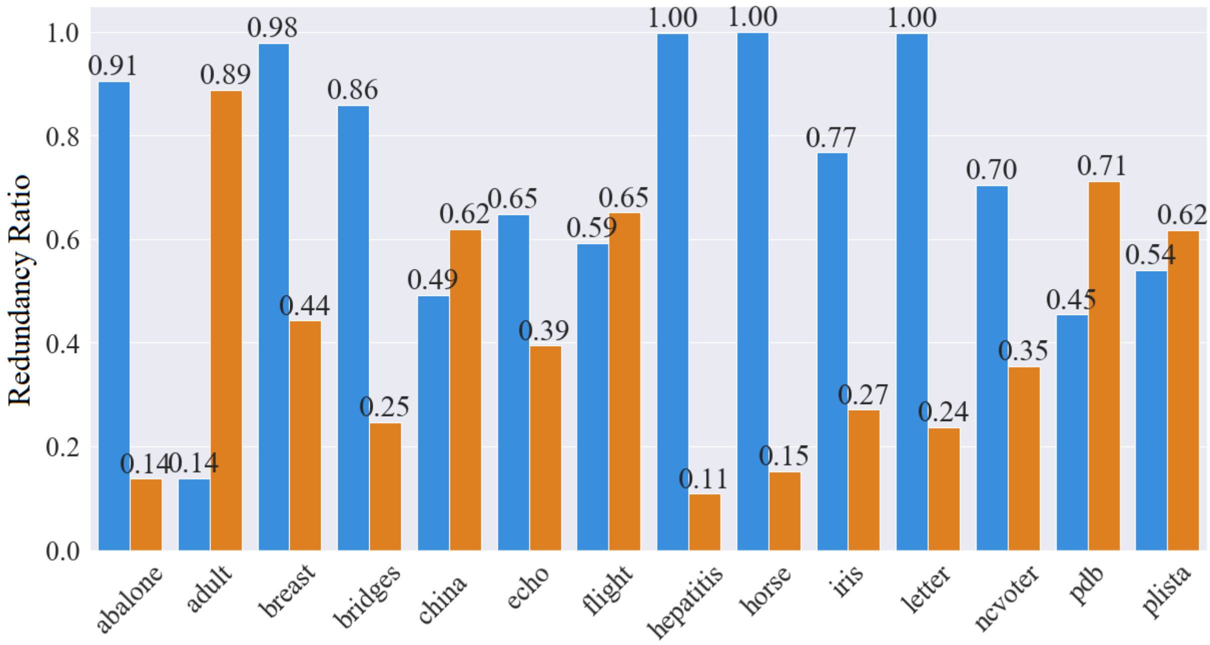


Figure 7: Average proportion of redundant data value occurrences caused (blue) and eliminated (orange) by FDs not preserved during Boyce-Codd decomposition

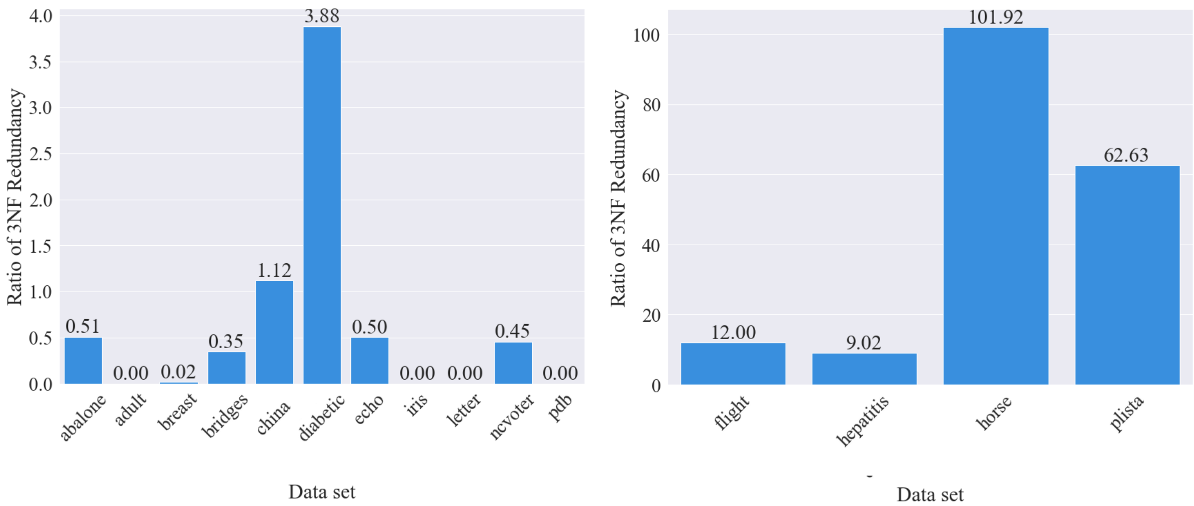


Figure 8: Data redundancy ratio after 3NF synthesis

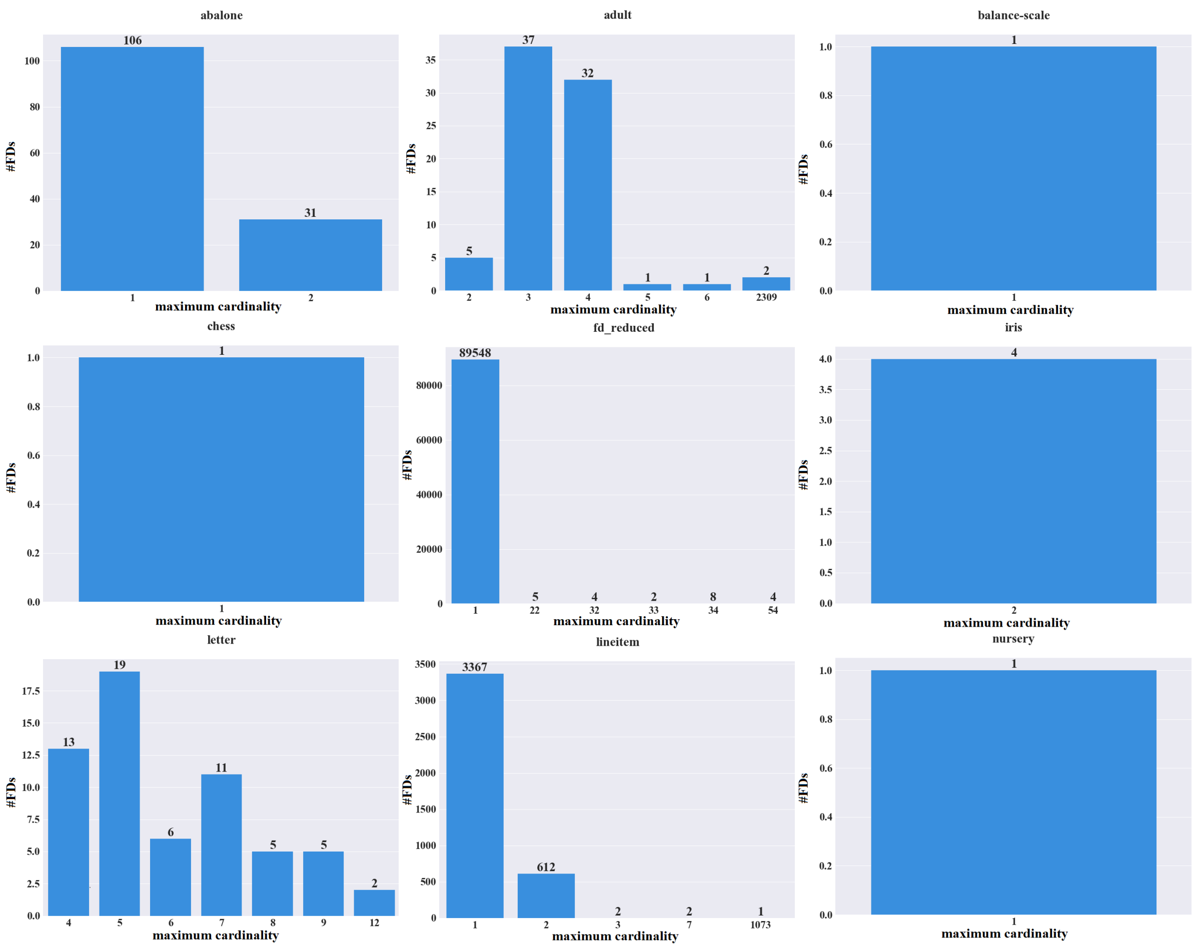


Figure 9: Distribution of redundancy levels for individual FDs on complete data sets

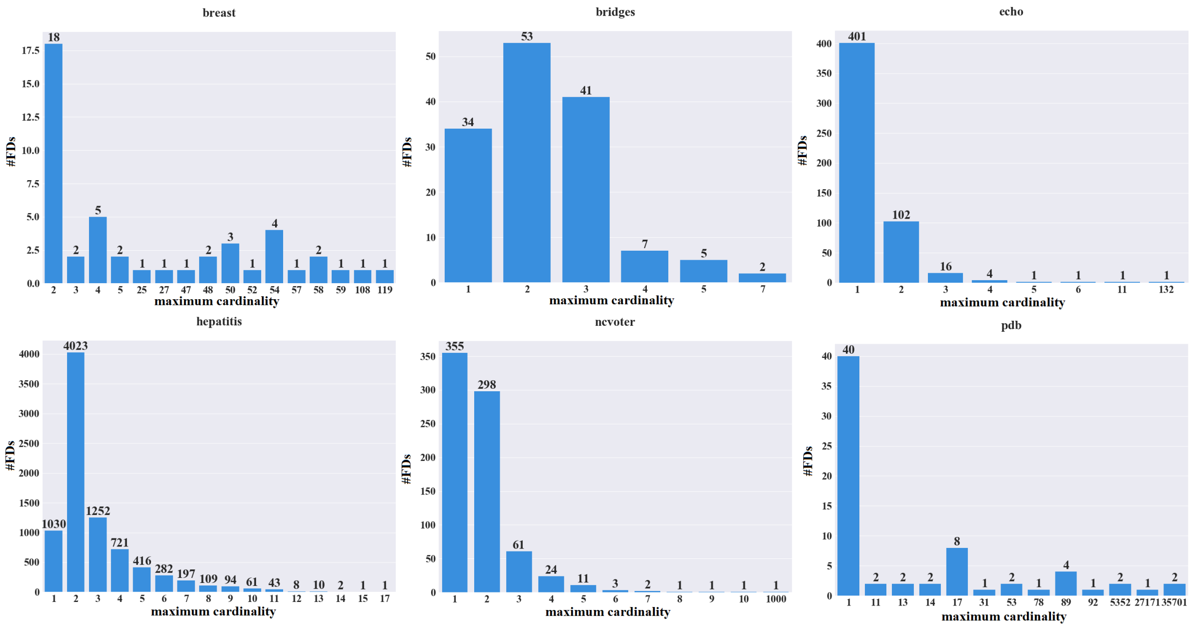


Figure 10: Distribution of redundancy levels for individual FDs on incomplete data sets

point is that these normal forms cannot guarantee a priori bounds on the level of data redundancy, the level of update inefficiency nor the level of join efficiency.

8.2 Data Profiling with ℓ-BCNF

Here we use the benchmark schemata and data to illustrate the insight that our normal forms can provide.

Firstly, we can use any legacy data for a given schema to compute a purely data-driven view for which positive integer ℓ the schema is in ℓ-BCNF, simply by computing ℓ_X for each left-hand side X of a non-trivial, non-key FD we previously mined. Note that this is also a very useful application of the FD discovery algorithms that have been developed over the last 40 years. In fact, we can take this illustration further and show the distribution of the ℓ_X for the data set. The results are shown in Figure 9, Figure 10 and Figure 11. Here, the labels on x-axis indicate the various levels ℓ_X of data redundancy, with the maximum level ℓ being the smallest positive integer for which the given data set is in ℓ-BCNF. The y-axis indicates how many of the discovered FDs $X \to Y$ have the associated cardinality bound ℓ_X on the x-axis.

One may also show which specific FDs have which associated levels of data redundancy, and later on use this information for other purposes, such as normalization. Indeed, the level of data redundancy - as introduced in this article - provides us with information that should drive the schema design process.

8.3 Indicator of Data Redundancy

Indeed, the left-hand columns of Figures 12 and 13 show for the bottom ten percent of our FDs which cardinality bounds are associated with them, while the right-hand columns

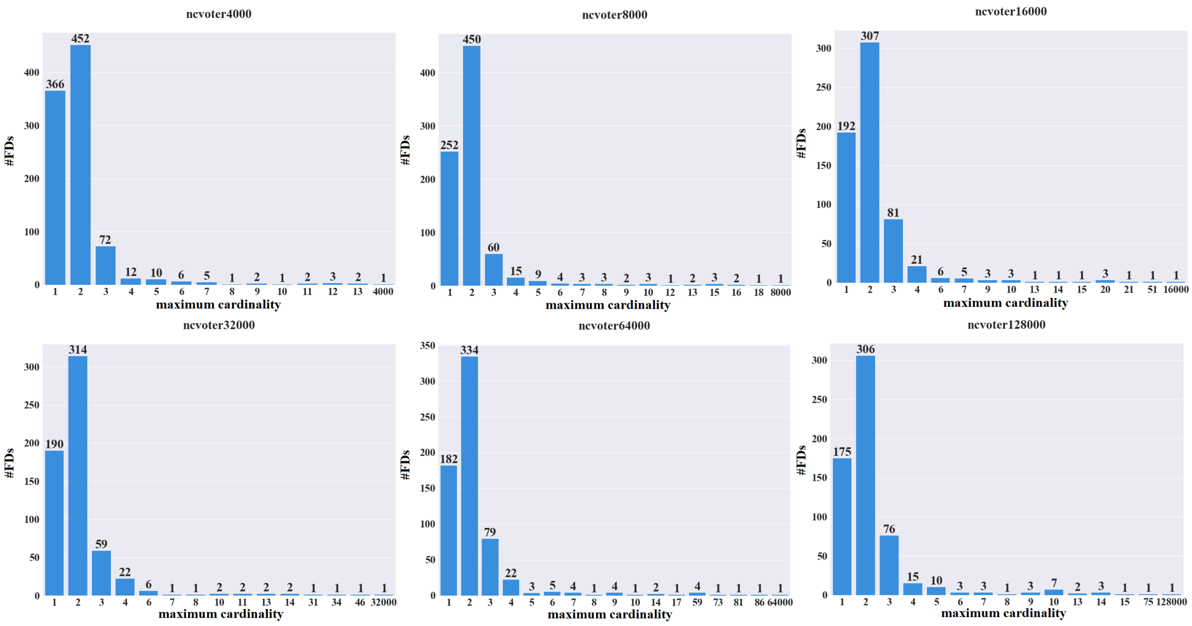


Figure 11: Distribution of redundancy levels for individual FDs on NCVoter fragments

show the bounds associated with the top ten percent of our FDs.

Figures 12 and 13 show that the level ℓ_X of data redundancy (termed maximum cardinality in the figures) is a strong indicator at design time which FDs will cause many redundant values in instances. Indeed, the left column shows that the bottom 10%-numbers of redundant values were caused by FDs $X \to Y$ where ℓ_X is small. The right column shows that the top 10%-numbers of redundant values were caused by FDs $X \to Y$ where ℓ_X is high. Hence, FDs that cause higher numbers of redundant values have typically higher levels of data redundancy.

8.4 Update and Join Efficiency Levels

We apply Algorithms 2 (Opt), 3 (Greed), and 4 (Hybrid) to our benchmark schemata, using FDs in the top-k percentages for $k = 20$ and $k = 80$ as input.

Our analysis visualizes the reduction of the update inefficiency levels $\ell^U_{\mathcal{D}}$ for the decomposition $\mathcal{D}$ compared to the input schema (Figure 14), and also the total levels, $\ell^{U,\text{total}}_{\mathcal{D}}$ (Figure 15), the size $|\mathcal{D}|$ of the output (Figure 16), and the runtime (Figure 17).

The reduction in the levels is significant for all three algorithms. As Figures 14 and 15 show, Greed achieves the least reduction, due to lost FDs. This loss could not be measured by the classical BCNF decomposition based on FDs alone. The join efficiency levels $\ell^J_{\mathcal{D}}$ and $\ell^{J,\text{total}}_{\mathcal{D}}$ for Greed all equal 1 since an FD is either transformed into a key or lost. Opt preserves all FDs, meaning that the (total) levels of update inefficiency and join efficiency always coincide. For FD-preserving decompositions, Opt always achieves the optimum level $\ell^U_{\mathcal{D}}$ of update inefficiency. Hybrid trades even lower levels of update inefficiency for lower levels of join efficiency. The various levels achieved by selecting the FDs from the top-20% and top-80%, respectively, are very similar. This shows that achievements depend primarily on those FDs that cause higher numbers of

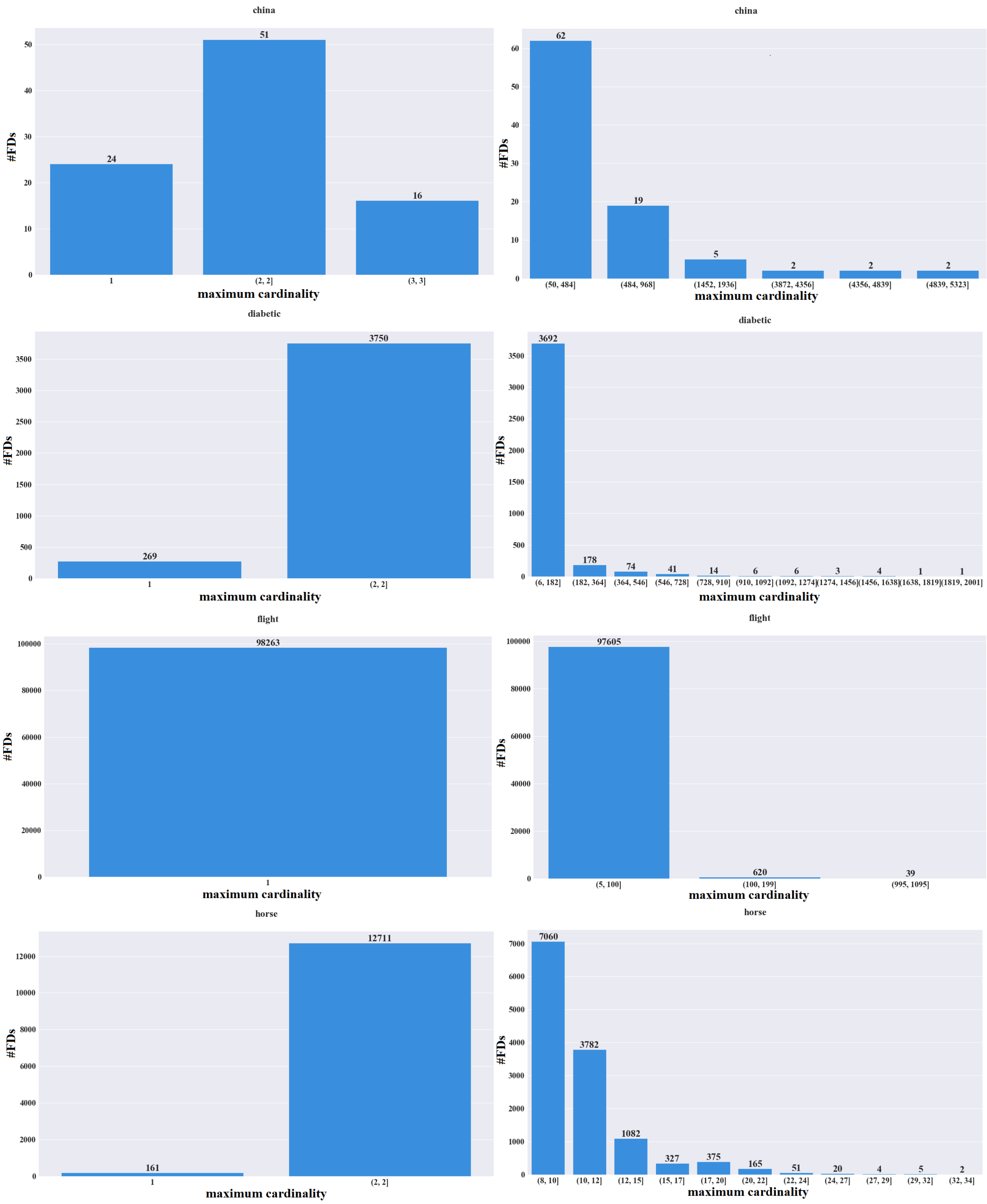


Figure 12: Left (Right): The x-axis shows the maximum cardinality (= level of data redundancy) for FDs in the bottom (top) 10% for the number of redundant values, the y-axis shows how many FDs have these levels - Part I

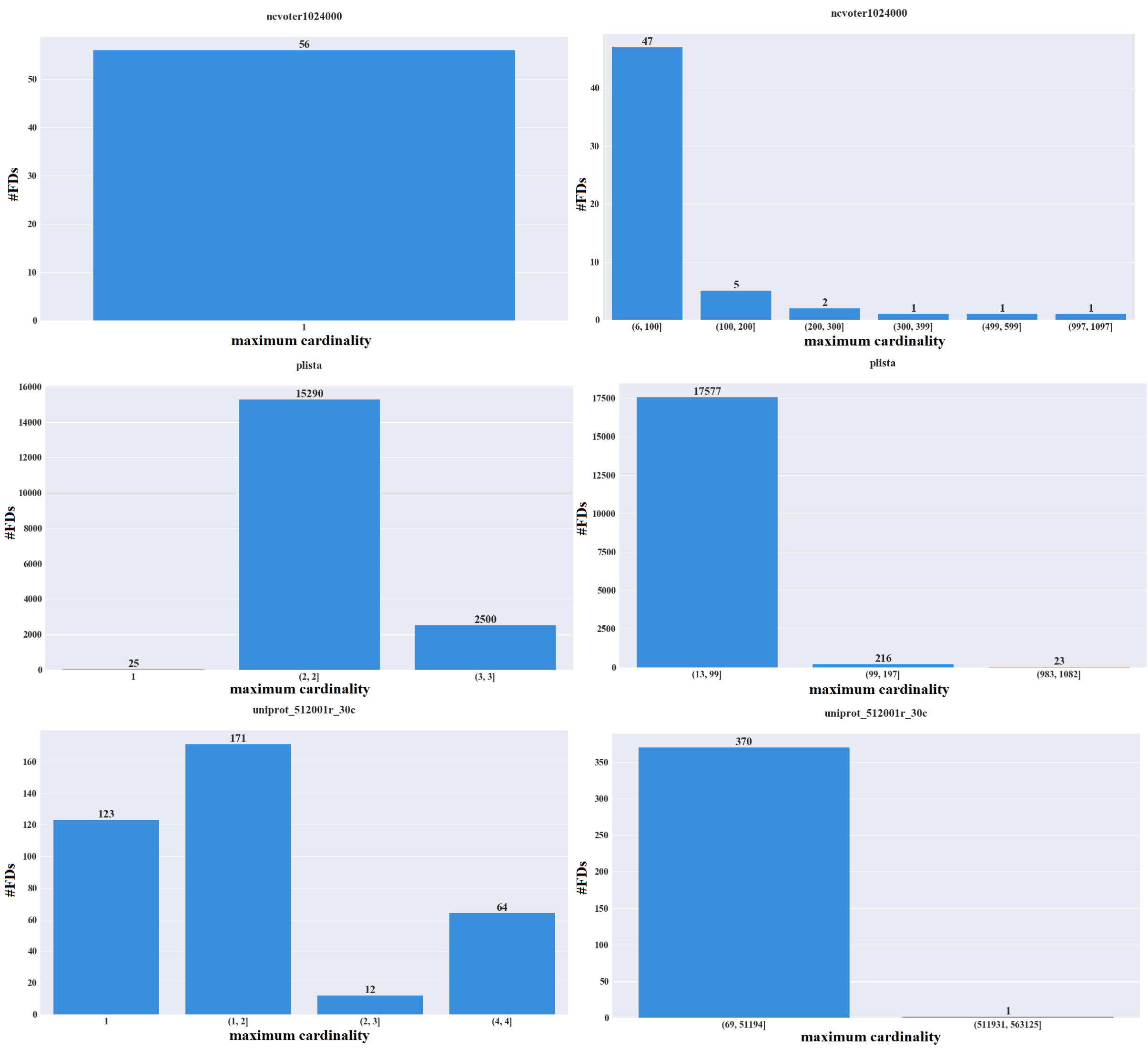


Figure 13: Left (Right): The x-axis shows the maximum cardinality (= level of data redundancy) for FDs in the bottom (top) 10% for the number of redundant values, the y-axis shows how many FDs have these levels- Part II

redundant values (indicated by higher levels of data redundancy according to our first experiment).

Figure 16 explains how the algorithms achieve different reductions in update inefficiency: Greed generally requires few schemata, while Opt achieves its minimization of update inefficiency by adding many schemata, and Hybrid achieves further reductions by even more schemata. An analysis of these trade-offs is important since additional schemata require additional joins when processing queries. Our results stress that supporting fewer joins in common queries may require the enforcement of many constraints on joins of schemata, and the local enforcement of constraints may require the processing of more joins during query evaluation. Figure 17 shows that all three algorithms work fast. Due to the importance of identifying the right database schema, the time spent on this crucial task will be time worth spent.

8.5 Top-k normalization

FDs that cause higher numbers of redundant data values appear to be more relevant for database normalization. We use Algorithm 2 to visualize the impact of using only the top-k percent of FDs mined from our data sets for our synthesis algorithms. The main takeaway from Figure 18 is that the top-30 to top-40% of FDs capture the majority of all redundant data values, around 75% of those. In fact, as Figure 18 shows, considering too many FDs can easily reverse the reduction in data redundancy (china, ncvoter on the left), or even introduce additional redundant data values due to their duplication across multiple schemata (flight, plista on the right). In fact, Figures 19 and 20 show that normalization with the top-30 to top-40% of FDs leads to decompositions of approximately half the size of those decompositions that are generated when all FDs are included, while both achieve similar total levels of data redundancy and eliminate similar numbers of redundant data values. This reconfirms that higher ranked FDs are more meaningful for normalization, which facilitates ideas such as data-driven normalization [44].

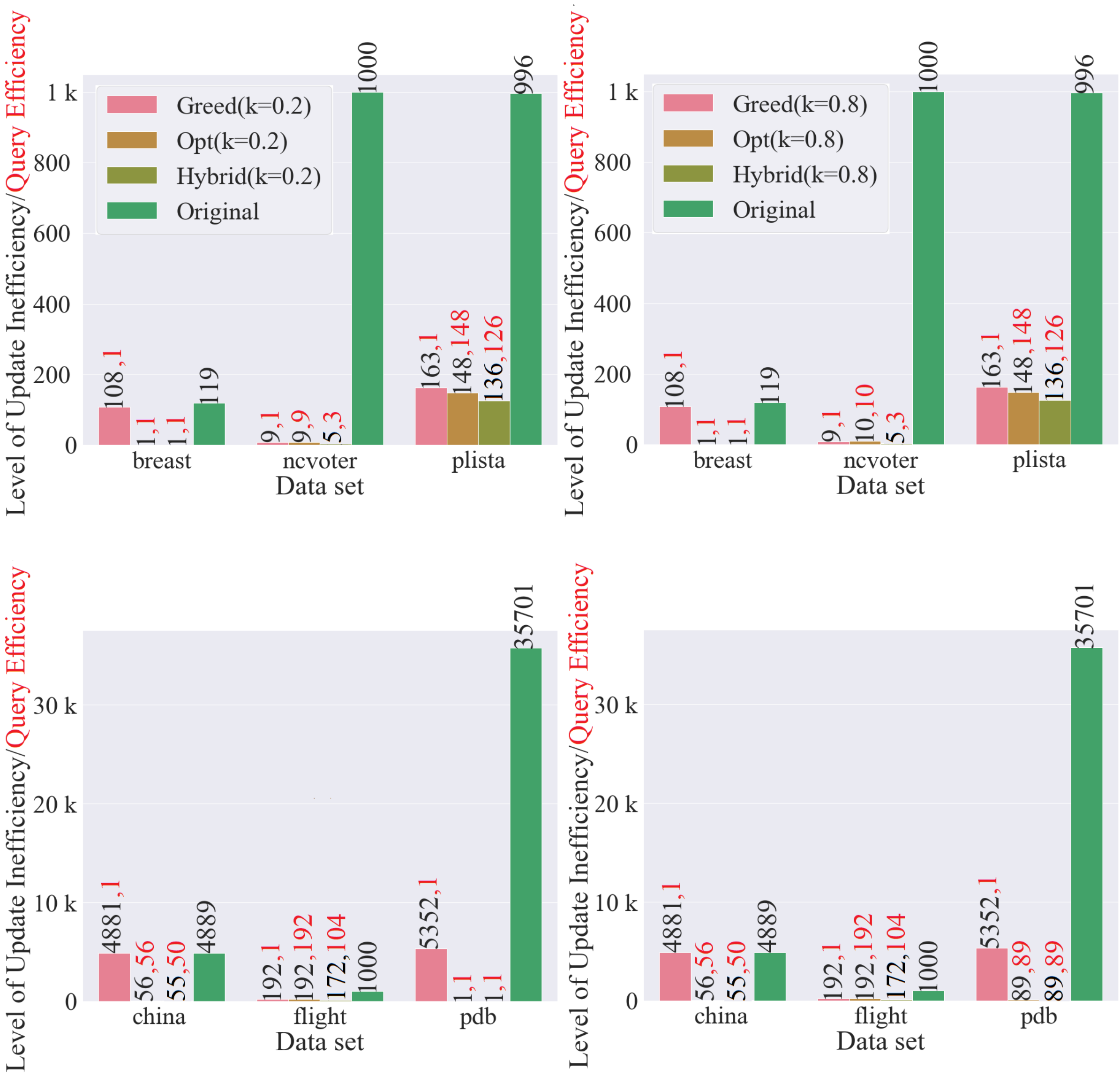


Figure 14: Reducing the level of redundancy using the top-20% (left) and top-80% (right) of FDs

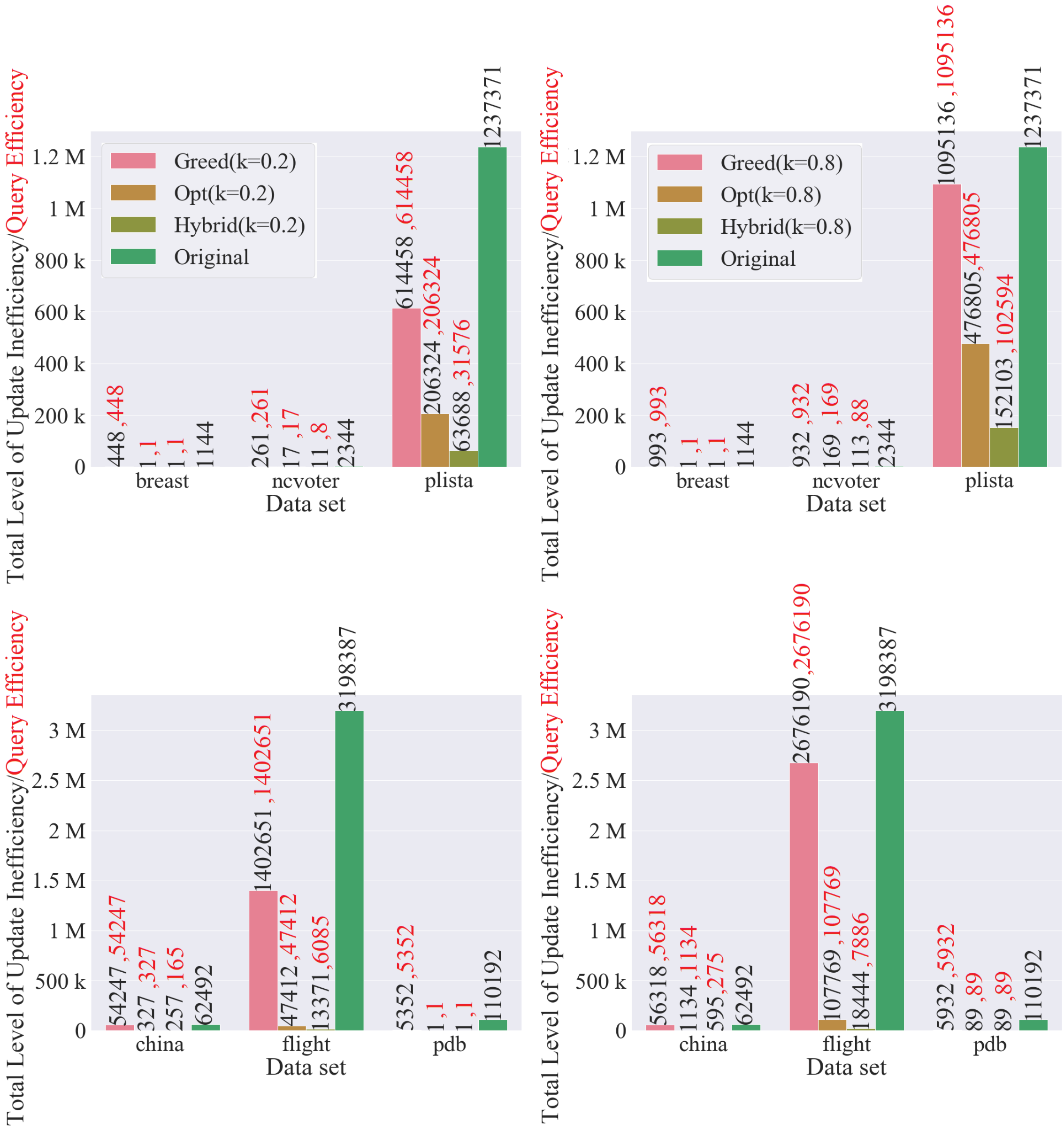


Figure 15: Reducing the total level of redundancy using the top-20% (left) and top-80% (right) of FDs

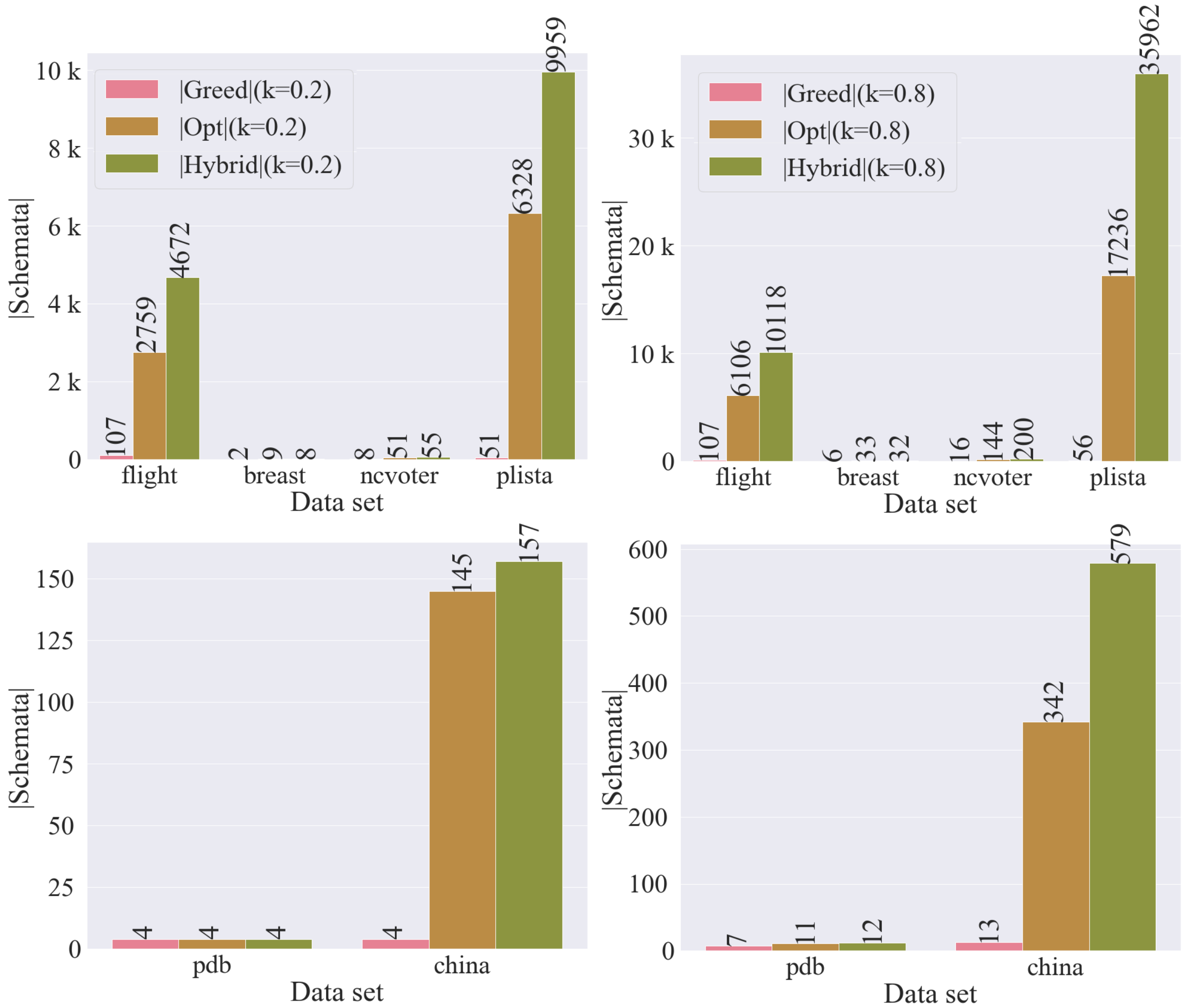


Figure 16: Number of schemata in output with top-20% (left) and top-80% (right) of FDs

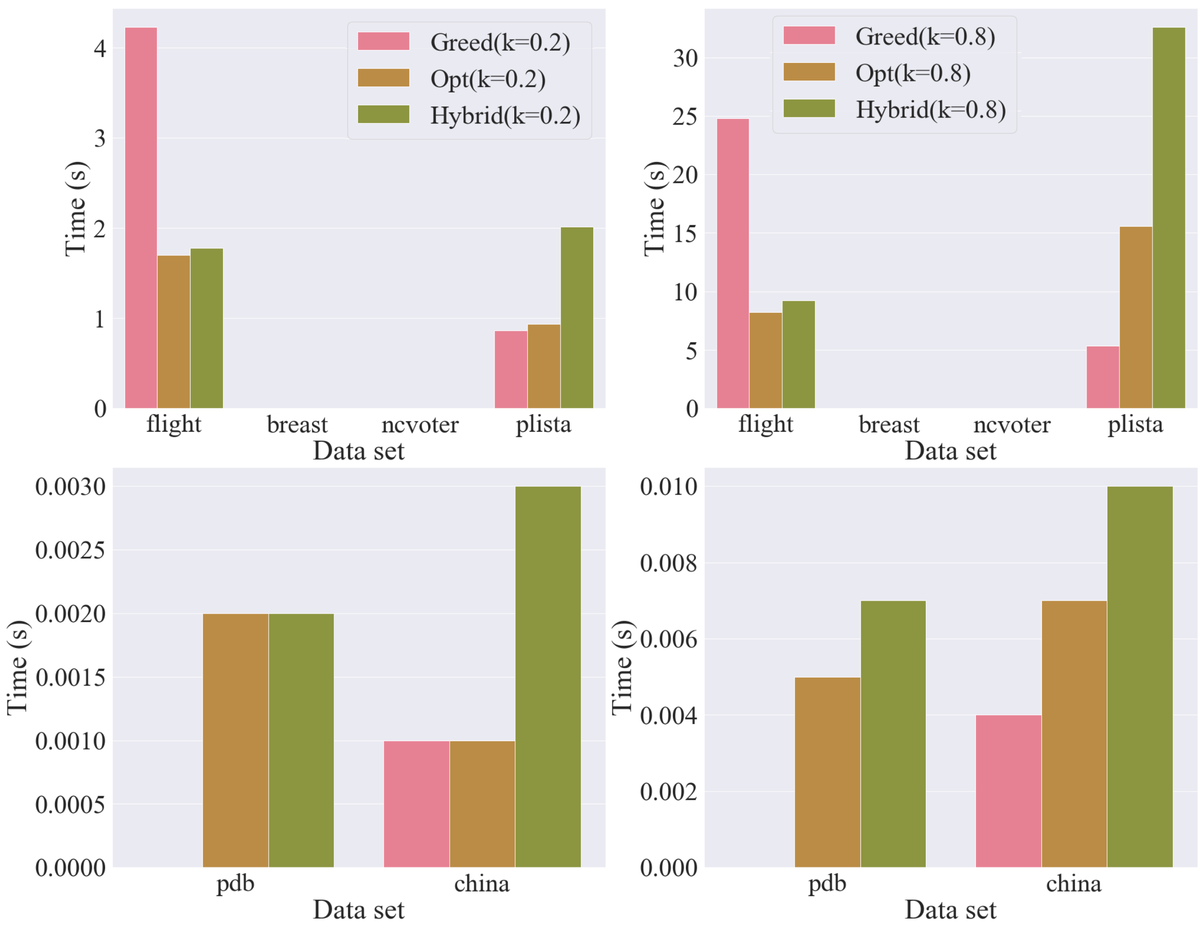


Figure 17: Time for algorithms with top-20% (left) and top-80% (right) of FDs

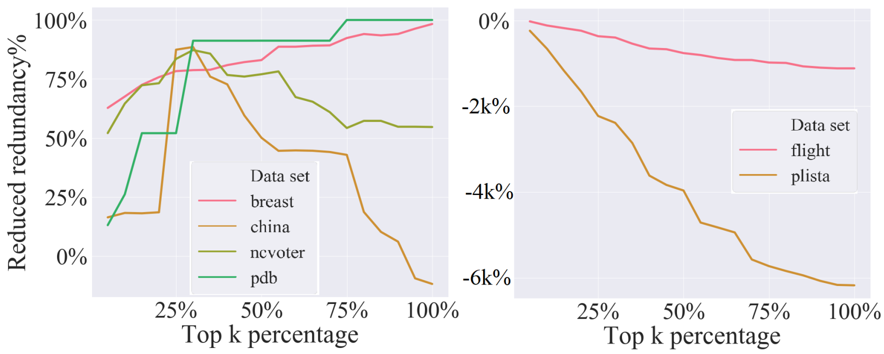


Figure 18: Ratio of data redundancy after applying Algorithm 2 to the top-k FDs

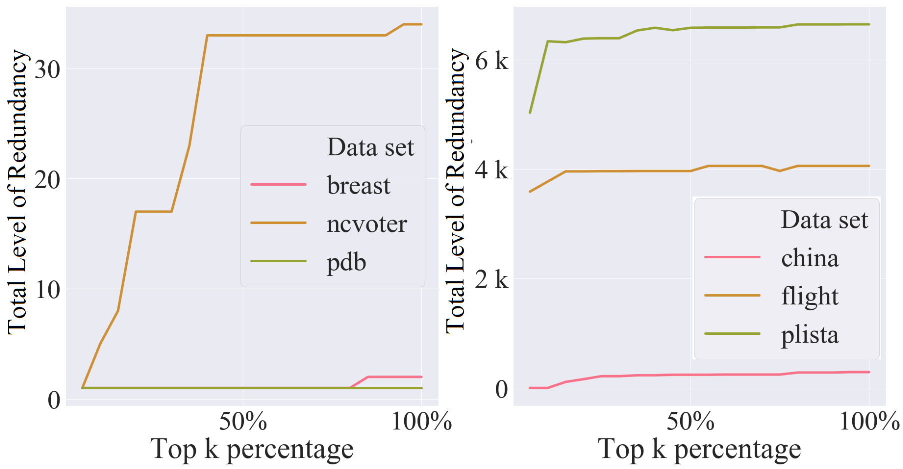


Figure 19: Total level of data redundancy after applying Algorithm 2 to top-k FDs

8.6 Physical Performance

We ran updates and join queries over our original data sets and the decompositions resulting from Algorithms 2 and 3 on input sets that included all FDs $X \to Y$ where ℓ_X met a given target level ℓ. Figure 21 shows the results of Algorithm 2 on diabetic, and Figure 22 shows the results of Algorithm 3. Updates involve changes to all occurrences of some redundant value. Indeed, a change for one occurrence of the redundant data value means that the other redundant occurrences of the same value need to be updated, too. After an FD is transformed into a key, only one value requires an update. The join queries are i) a simple SELECT $*$ FROM HAP over the original schema HAP, and ii) a SELECT $*$ FROM R_1 NATURAL JOIN ... NATURAL JOIN R_n over the decomposition $\{R_1, \ldots, R_n\}$. The x-axis shows the target level ℓ, the number n of tables in the decomposition and the number of FDs in the input. The left y-axis shows the time in seconds required for the join query, while the right y-axis (logarithmic scale) shows the average time in seconds required for the updates. Average update times for keys of the decomposition are consistently two orders of magnitude lower than those for the input FDs on the original data. For preserved non-key FDs the gain is approximately one order of magnitude (grey lines in Figure 21, with no data point when all FDs were transformed

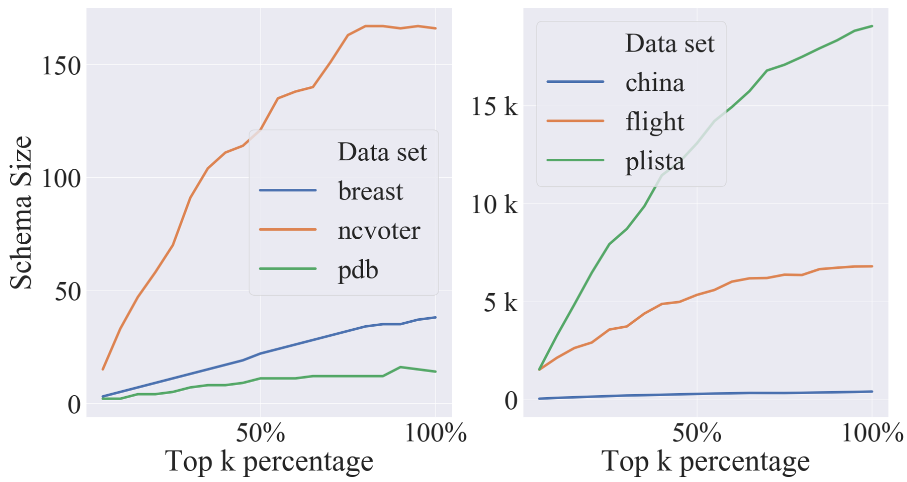


Figure 20: Size of decompositions after applying Algorithm 2 to top-k FDs

into keys). However, lost FDs require more update time on the decomposition than on the original data set. This is because lost FDs can only be updated on the join of schemata that contain all their attributes. If there are no lost FDs, only keys need to be enforced (in Figure 22 for the largest target level). Join query times increase with lower target levels ℓ. For Opt, join performance becomes prohibitively poor due to the many output tables that guarantee FD preservation. This is compensated by Greed, but the trade-off is poor update performance for lost FDs. The level ℓ, solely determined by the input integrity constraints, translates into different physical performance degrees for updates and joins in the output of our algorithms. Making these options available is precisely the point of our new framework.

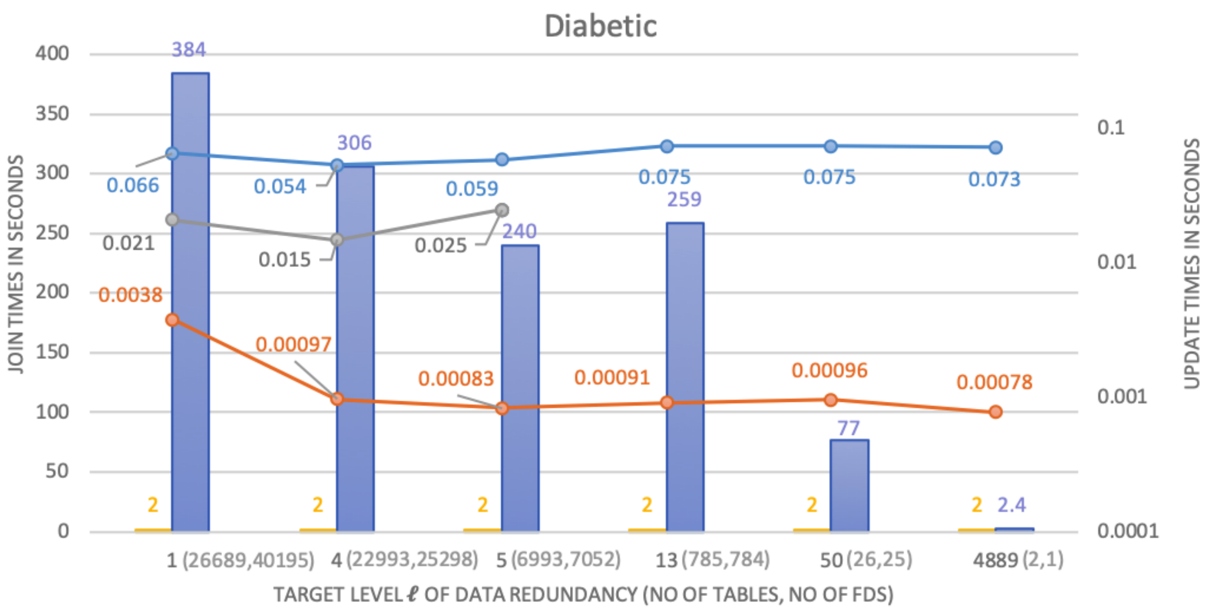


Figure 21: Update and join performance on Opt

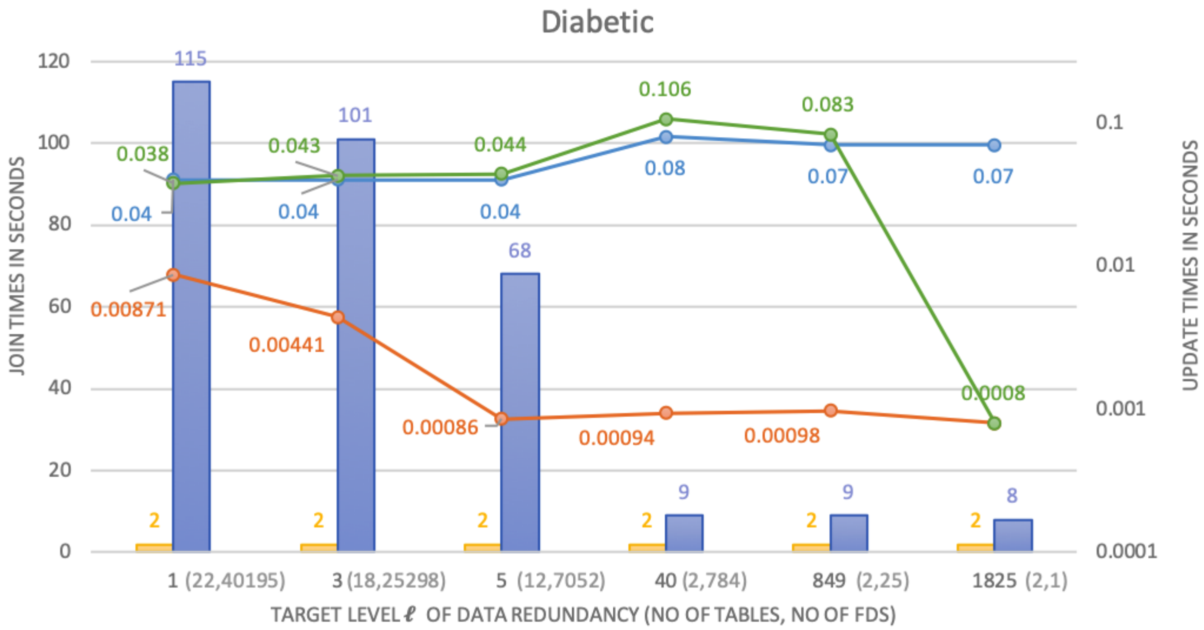


Figure 22: Update and join performance on Greed

9 Related Work

Schema design has been studied in data models such as SQL [26], nested relations [40], data warehouses [30], object-orientation [24], semantics [9], temporality [23], the Web [1], uncertainty [35], and graphs [16]. Similar to classical design, our results may be extended to richer data models and higher normal forms such as 4NF [14], Project-Join Normal Form [15], or Inclusion-Dependency Normal Form [33].

Unlike BCNF/3NF [4, 6], CCs measure the effort for achieving data consistency. Our algorithms return designs that quantify worst-case update inefficiency and best-case join efficiency. Our concepts also quantify incremental maintenance and join support of views during logical design. In contrast, numerical dependencies (NDs) apply classical normalization after horizontal decomposition into blocks where FDs hold [19]. NDs are not finitely axiomatizable [18].

Schema design is not expected to optimize the layout for all instances. Instead, physical tuning kicks in after deploying the logical schema and once reliable data retrieval patterns emerge. These include virtual de-normalization in main memory databases [36], migration to NoSQL [51], and data warehouse designs [30]. Guidelines have been devised to recover 3NF from de-normalised schema [45], and de-normalizing normalised schemata [7]. Information-theoretic justifications exist for fact tables in snowflake schemata [32] and for 3NF [28]. Update inefficiency and join efficiency inform the difficult problem of selecting materialized views [11].

Kojić and Milićev de-normalize by maximizing read benefits while write costs remain under a threshold [27]. Their technique tunes a logical schema (for example, in 3NF) based on specific updates and queries. Hence, their approach is applicable once the database is operational and a mature workload model available. This additional input makes it possible to derive a schema optimized for the input workloadfor [27]. The work in [27] does not use CCs. Our approach is for logical schema normalization where a workload model in terms of frequent update and query operations is not available yet. We use CCs to explore various normal forms with different properties in terms of update inefficiency and join efficiency on the schemata. Hence, the design team can assess different designs based on their properties. We only require a set of attributes, FDs, and CCs as input. As our work brings forward a logical schema, it may provide a different starting point for applying the work in [27] than classical normalization does. For example, once frequent updates and queries have emerged, we may apply [27] to the output of Algorithm 2 rather than the 3NF schema $(\textsc{Hap}, \Sigma)$.

Recent approaches to schema design and evolution for NoSQL databases [47] are driven by a query workload, and NoSQL schemata are thus de-normalized. We are unaware of work for logical NoSQL schema design for CCs and FDs.

"Cardinality constraints are one of the most important kinds of constraint in conceptual modeling" [41]. "CCs correspond to very common semantic rules on relationships and their formal definition at the conceptual level improves significantly the completeness of data description" [31]. Surprisingly, our way of applying CCs to logical schema design has not been observed before. CCs were introduced in Chen's seminal ER paper [9], and have been studied in data models such as semantic [34], Web [17], spatial and temporal [13], and uncertain models [46]. They are part of major languages for data and knowledge modeling, including UML, EER, ORM, XSD (such as `maxOccurs`), or OWL (such as `owl:maxCardinality`) [20]. They have also been used in data cleaning [10], database testing [8], query answering [12] and reverse engineering [48].

Violations of dependencies by dirty data motivate relaxed notions such as approximate keys and FDs [22, 29, 39]. When mining from (dirty) data, approximate notions may improve the recall but worsen the precision of identifying meaningful rules. The CC $card(X) \leq \ell$ may be seen as an approximate key permitting up to ℓ duplicates. For small ℓ, we may therefore view our work as extending classical schema design to approximate keys, offering some robustness for dirty data.

10 Conclusion and Future Work

We introduced the first logical schema design framework that measures update inefficiency and join efficiency, based on integrity constraints alone. This is possible by the new notion of level-ℓ data redundancy, which is determined by the upper bounds of CCs at schema design time. Our infinite family of ℓ-Bounded Cardinality Normal Forms characterizes instances that are free from level ℓ data redundancy and update inefficiency, and permit level ℓ join efficiency. We developed algorithms for schema design, and illustrated experimentally how they reduce the levels of update inefficiency and join efficiency with trade-offs in the size of output designs. We also showed experimentally how these levels quantify the suitability of schema designs and materialized views for the performance of specific updates and joins on instances of the designs. Our framework uses domain knowledge about CCs to advance logical schema design.

Future work will address more constraints and data models. We expect the interaction

of CCs and join dependencies to challenge the development of higher normal forms.
Acknowledgements. The authors thank the anonymous reviewers for their valuable suggestions. The first author thanks Joachim Biskup, Bernhard Thalheim, and Jef Wijsen for insightful discussions during Dagstuhl Seminar 19031.

References

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

[2] W. W. Armstrong. Dependency structures of data base relationships. In *IFIP congress*, volume 74, pages 580–583. Geneva, Switzerland, 1974.

[3] 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.

[4] C. Beeri, P. A. Bernstein, and N. Goodman. A sophisticate's introduction to database normalization theory. In *VLDB*, pages 113–124, 1978.

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

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

[7] D. B. Bock and J. F. Schrage. Denormalization guidelines for base and transaction tables. *ACM SIGCSE Bull.*, 34(4):129–133, 2002.

[8] N. Bruno, S. Chaudhuri, and D. Thomas. Generating queries with cardinality constraints for DBMS testing. *IEEE Trans. Knowl. Data Eng.*, 18(12):1721–1725, 2006.

[9] P. Chen. The entity-relationship model-toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, 1976.

[10] W. Chen, W. Fan, and S. Ma. Incorporating cardinality constraints and synonym rules into conditional functional dependencies. *Inf. Process. Lett.*, 109(14):783–789, 2009.

[11] R. Chirkova and J. Yang. Materialized views. *Found. Trends Databases*, 4(4):295–405, 2012.

[12] G. Cormode, D. Srivastava, E. Shen, and T. Yu. Aggregate query answering on possibilistic data with cardinality constraints. In *ICDE*, pages 258–269, 2012.

[13] F. Currim and S. Ram. Conceptually modeling windows and bounds for space and time in database constraints. *Commun. ACM*, 51(11):125–129, 2008.

[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. Normal forms and relational database operators. In *SIGMOD*, pages 153–160, 1979.

[16] W. Fan, Y. Wu, and J. Xu. Functional dependencies for graphs. In *SIGMOD*, pages 1843–1857, 2016.

[17] F. Ferrarotti, S. Hartmann, and S. Link. Efficiency frontiers of XML cardinality constraints. *Data Knowl. Eng.*, 87:297–319, 2013.

[18] J. Grant and J. Minker. Inferences for numerical dependencies. *Theor. Comput. Sci.*, 41:271–287, 1985.

[19] J. Grant and J. Minker. Normalization and axiomatization for numerical dependencies. *Inf. Control.*, 65(1):1–17, 1985.

[20] N. Hall, H. Köhler, S. Link, H. Prade, and X. Zhou. Cardinality constraints on qualitatively uncertain data. *Data Knowl. Eng.*, 99:126–150, 2015.

[21] S. Hartmann. Reasoning about participation constraints and Chen's constraints. In *ADC*, pages 105–113, 2003.

[22] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. TANE: an efficient algorithm for discovering functional and approximate dependencies. *Comput. J.*, 42(2):100–111, 1999.

[23] C. S. Jensen, R. T. Snodgrass, and M. D. Soo. Extending existing dependency theory to temporal databases. *IEEE Trans. Knowl. Data Eng.*, 8(4):563–582, 1996.

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

[25] H. Köhler. Finding faithful boyce-codd normal form decompositions. In *AAIM*, pages 102–113, 2006.

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

[27] N. Kojic and D. Milicev. Equilibrium of redundancy in relational model for optimized data retrieval. *IEEE Trans. Knowl. Data Eng.*, 32(9):1707–1721, 2020.

[28] 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.

[29] S. Kruse and F. Naumann. Efficient discovery of approximate dependencies. *Proc. VLDB Endow.*, 11(7):759–772, 2018.

[30] J. Lechtenbörger and G. Vossen. Multidimensional normal forms for data warehouse design. *Inf. Syst.*, 28(5):415–434, 2003.

[31] M. Lenzerini and G. Santucci. Cardinality constraints in the entity-relationship model. In *ER*, pages 529–549, 1983.

[32] M. Levene and G. Loizou. Why is the snowflake schema a good data warehouse design? *Inf. Syst.*, 28(3):225–240, 2003.

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

[34] S. W. Liddle, D. W. Embley, and S. N. Woodfield. Cardinality constraints in semantic data models. *Data Knowl. Eng.*, 11(3):235–270, 1993.

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

[36] Z. Liu and S. Idreos. Main memory adaptive denormalization. In *SIGMOD*, pages 2253–2254, 2016.

[37] D. Maier. Minimum covers in relational database model. *J. ACM*, 27(4):664–674, 1980.

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

[39] P. Mandros, M. Boley, and J. Vreeken. Discovering reliable approximate functional dependencies. In *SIGKDD*, pages 355–363, 2017.

[40] W. Y. Mok, Y. Ng, and D. W. Embley. A normal form for precisely characterizing redundancy in nested relations. *ACM Trans. Database Syst.*, 21(1):77–106, 1996.

[41] A. Olivé. Cardinality constraints. In *Conceptual Modeling of Information Systems*, pages 83–102. Springer, 2007.

[42] T. Papenbrock, J. Ehrlich, J. Marten, T. Neubert, J. Rudolph, M. Schönberg, J. Zwiener, and F. Naumann. Functional dependency discovery: An experimental evaluation of seven algorithms. *PVLDB*, 8(10):1082–1093, 2015.

[43] T. Papenbrock and F. Naumann. A hybrid approach to functional dependency discovery. In *SIGMOD*, pages 821–833, 2016.

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

[45] J. Petit, F. Toumani, J. Boulicaut, and J. Kouloumdjian. Towards the reverse engineering of denormalized relational databases. In *ICDE*, pages 218–227, 1996.

[46] T. Roblot, M. Hannula, and S. Link. Probabilistic cardinality constraints. *VLDB J.*, 27(6):771–795, 2018.

[47] S. Scherzinger and S. Sidortschuck. An empirical study on the design and evolution of nosql database schemas. In *ER*, pages 441–455, 2020.

[48] C. Soutou. Relational database reverse engineering: Algorithms to extract cardinality constraints. *Data Knowl. Eng.*, 28(2):161–207, 1998.

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

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

[51] J. Yoo, K. Lee, and Y. Jeon. Migration from RDBMS to nosql using column-level denormalization and atomic aggregates. *J. Inf. Sci. Eng.*, 34(1):243–259, 2018.