# Synthesizing Third Normal Form Schemata that Minimize Integrity Maintenance and Update Overheads

Parameterizing 3NF by the Numbers of Minimal Keys and Functional Dependencies

Author names surpressed due to double-blind reviewing

## ABSTRACT

State-of-the-art relational schema design generates a lossless, dependency-preserving decomposition into Third Normal Form (3NF), that is in Boyce-Codd Normal Form (BCNF) whenever possible. In particular, dependency-preservation ensures that data integrity can be maintained on individual relation schemata without having to join them, but may need to tolerate a priori unbounded levels of data redundancy and integrity faults. As our main contribution we parameterize 3NF schemata by the numbers of minimal keys and functional dependencies they exhibit. Conceptually, these parameters quantify, already at schema design time, the effort necessary to maintain data integrity, and allow us to break ties between 3NF schemata. Computationally, the parameters enable us to optimize normalization into 3NF according to different strategies. Operationally, we show through experiments that our optimizations translate from the logical level into significantly smaller update overheads during integrity maintenance. Hence, our framework provides access to parameters that guide the computation of logical schema designs which reduce operational overheads.

**ACM Reference Format:**

Author names surpressed due to double-blind reviewing. 2025. Synthesizing Third Normal Form Schemata that Minimize Integrity Maintenance and Update Overheads: Parameterizing 3NF by the Numbers of Minimal Keys and Functional Dependencies. In *Proceedings of the 2025 International Conference on Management of Data (SIGMOD '25), June 22–27, 2025, Berlin, Germany.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3448016.3459238

## 1 INTRODUCTION

We will revisit classical normalization in the relational model of data, in particular Third and Boyce-Codd Normal Form (3NF, BCNF) that are based on functional dependencies (FDs) [7, 12, 18]. These topics are fundamental and taught in introductory database courses [1, 20, 26]. We will not discuss higher normal forms [9, 21, 32].

Arguably, 3NF is the most popular normal form in database practice. While BCNF guarantees that no relation can ever exhibit any redundant data value [7], a lossless, dependency-preserving decomposition into BCNF is not always achievable [6]. In contrast, a lossless, dependency-preserving decomposition into 3NF is always possible [7, 8, 36], but may need to tolerate unbounded levels of data redundancy and potential integrity faults [7, 16, 23]. Without dependency-preservation, maintaining data integrity is regarded as prohibitively expensive since relation schemata need to be joined before FDs can be validated [8, 23]. The use of 3NF is further promoted as it admits the fewest sources of data redundancy among all lossless, dependency-preserving decompositions [17].

State-of-the-art normalization computes a lossless, dependency-preserving decomposition into 3NF that is in BCNF whenever possible [28, 37]. Intuitively, the algorithms checks for every critical schema (a relation schema that is in 3NF but not in BCNF) whether it is redundant. If any critical schema is non-redundant, no lossless, dependency-preserving decomposition can be in BCNF. Despite 3NF being one of the most fundamental topics in database education and practice for close to 50 years of research, state-of-the-art algorithms make arbitrary choices between critical schemata that redundant. This is illustrated as follows.

*Example 1.1.* As running example, we consider schema $R = \{E(vent), M(anager), S(tatus), V(enue), T(ime)\}$ which records the name of events, their managers, status, venue and time they are held. The set $\mathcal{D}$ of FDs consists of: $VSE \rightarrow T$, $SET \rightarrow V$, $SME \rightarrow V$, $VS \rightarrow M$, $SME \rightarrow T$, $MT \rightarrow E$, and $ET \rightarrow M$. Based on arbitrary choices, state-of-the-art normalization [28, 37] we may return different decompositions, such as $\mathbb{D}_1$ of $(R, \mathcal{D})$:

- $R_1 = ESTV$ and $\mathcal{D}_1$ with 3 minimal keys $EST$, $ESV$, and $STV$
- $R_2 = EMT$ and $\mathcal{D}_2$ with 2 minimal keys $ET$ and $MT$
- $R_3 = EMSV$ and $\mathcal{D}_3$ with FD $VS \rightarrow M$ and 2 minimal keys $ESV$ and $EMS$.

or the decomposition $\mathbb{D}_2$ of $(R, \mathcal{D})$:

- $R_1 = ESTV$ and $\mathcal{D}_1$ with 3 minimal keys $EST$, $ESV$, and $STV$
- $R_4 = EMST$ and $\mathcal{D}_4$ with two FDs $MT \rightarrow E$, $ET \rightarrow M$ and three minimal keys $EST$, $EMS$, and $MST$
- $R_5 = MSV$ and $\mathcal{D}_5$ with 1 minimal key $VS$.

Both decompositions contain $R_1$, while $\mathbb{D}_1$ has BCNF-schema $R_2$ with 2 minimal keys, $\mathbb{D}_2$ has BCNF-schema $R_5$ with just 1 minimal key, and while $\mathbb{D}_1$ has 3NF-schema $R_3$ with 1 FD, $\mathbb{D}_2$ has 3NF-schema $R_4$ with 2 FDs. □

Example 1.1 illustrates challenges with current state-of-the-art. Firstly, there are no systematic means to break ties between 3NF schemata. Hence, we ask (Q1) *What constitutes better 3NF schemata?* During schema design, no future workload of the target database is known yet, and the only input available to 3NF synthesis consists of some schema and set $\mathcal{D}$ of FDs. Secondly, the current 3NF condition does not provide information sufficient for separating better from worse 3NF schemata. This leads to (Q2) *How can we refine the existing Third Normal Form condition to identify better 3NF schemata?* Thirdly, even the ability to separate better and worse 3NF schemata does not yet tell us how to optimize 3NF synthesis. Hence, we ask (Q3) *In which sense can we optimize 3NF synthesis?*



*SIGMOD '25, June 22–27, 2024, Berlin, Germany*



ACM ISBN 978-1-4503-8343-1/21/06…$15.00
https://doi.org/10.1145/3448016.3459238

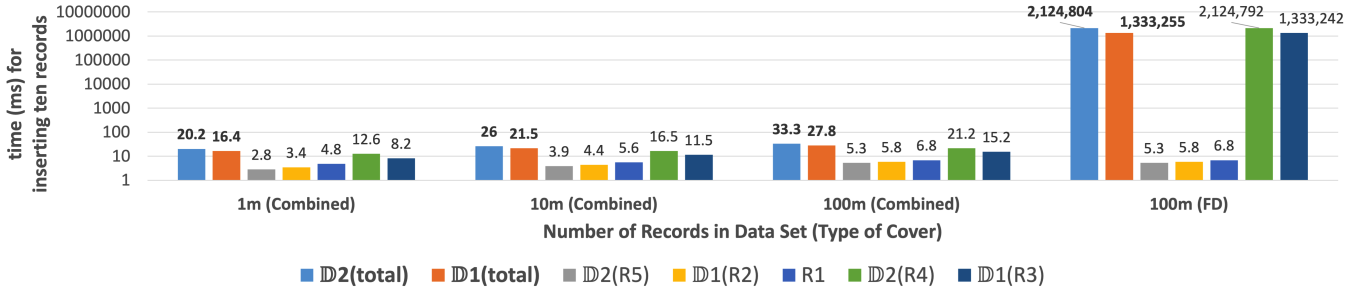



**Figure 1: Update Overhead (ms) for Insertions of 10 Records on 1, 10 and 100 Million Records over decomposition $\mathbb{D}_1 = \{R_1, R_2, R_3\}$ and decomposition $\mathbb{D}_2 = \{R_1, R_4, R_5\}$**

Finally, we want to see better outcomes at the operational level in terms of minimizing integrity maintenance and update overheads. Consequently, we wonder (Q4) *How does optimization at the logical level transcend to the operational level?* Answers to these questions will advance the rich knowledge about relational databases but also modern data models, such as incomplete [19, 34], temporal [14], Web [3, 10, 35], uncertain [22] and graph data [2, 31].

We will address the research questions as follows. We will use Maier's seminal notions of minimal-reduced (and optimal) covers [25] to address (Q1). Our main idea is to measure the operational effort of FD maintenance already at the logical level, namely by using the magnitude required for representing FDs. We separate an FD set $\mathcal{D}$ into the set $\mathcal{K}$ of minimal keys it implies, and a minimal-reduced cover $\mathcal{F}$ [25] for the set of non-key FDs it implies, that is, FDs whose left-hand side is not a key. Hence, the number $k$ of elements in $\mathcal{K}$ measures how much maintenance is shifted into minimal keys, while the number $f$ of elements in $\mathcal{F}$ measures the effort of maintaining non-key FDs. As a result, we can compare 3NF schemata based on $k$ and $f$.

In addressing (Q2), we will refine the well-known 3NF condition, gaining access to the parameters $k$ and $f$. Hence, we can break ties between 3NF schemata based on a strategy we target. In our example, if our strategy prefers schemata with fewer non-key FDs (S1), then $\mathbb{D}_1$ is preferred over $\mathbb{D}_2$. However, if schemata with more minimal keys are targeted (S2), then $\mathbb{D}_2$ is preferred over $\mathbb{D}_1$.

Instead of choosing an arbitrary redundant critical schema during state-of-the-art 3NF synthesis, we can now choose a schema that is preferred by our strategy. Simply put, the output of our new 3NF synthesis will be optimized for our target strategy. In our example, $\mathbb{D}_1$ is optimized for (S1), while $\mathbb{D}_2$ is optimized for (S2), illustrating our approach to (Q3).

In answering (Q4), we observe that non-key FDs cause the biggest bottleneck for maintaining data integrity: their validation is orders of magnitude slower than that of minimal keys. Hence, biggest performance improvements are to be expected from speeding up the maintenance of non-key FDs. However, this can be formulated as a target strategy, namely (S1) in our example. As we will show, (S1) translates optimizations from logical to the operational level where integrity maintenance and update overheads are minimized, thereby answering (Q4). For our running example, Figure 1 shows a study we conducted to measure the update overhead incurred by the decompositions in Example 1.1. As underlying data sets we used 1, 10, and 100 million records of synthetic data, respectively, each of which satisfies the given set of FDs but violates every FD not implied by the set. Hence, the data set is a perfect representation of the given constraint set. These data sets are then projected onto each element of the decompositions $\mathbb{D}_1$ and $\mathbb{D}_2$, respectively. The blue (orange) bars show times taken to insert 10 records into the projected data sets of $\mathbb{D}_1$ and $\mathbb{D}_2$, which measures the effort of maintaining integrity used by outputs of the algorithms. Times reported are averaged over 30 runs. The two bars on the right use an FD cover that enforces all constraints (non-key FDs and minimal keys) uniformly as FDs using triggers, while a combined cover of non-key FDs and minimal keys (and their `UNIQUE` indices) is used elsewhere. The remaining bars break down these total times on relation schemata of the two decompositions (both share the schema $R_1$, so it is only reported once). There are two main observations: (1) Combined covers facilitate integrity maintenance that is orders of magnitude faster than FD covers. (2) Update overheads on $\mathbb{D}_1$, resulting from strategy (S1), are significantly smaller than those on $\mathbb{D}_2$, resulting from strategy (S2), and this is true across all sizes of the data set and both types of covers. This provides quantitative insight how our normalization strategies at the logical level transcend to operational level, reducing update overheads based on strategy (S1) in our example. Record deletions cannot result in violations of FDs, so updates are a sequence of record deletion and insertion.

Our **contributions** can be summarized as follows.

(1) Fundamentally, we introduce parameters that quantify the effort of maintaining data integrity for FDs, making it possible to break ties between 3NF schemata. For that purpose, we parameterize classical relational normalization by separating non-key FDs from minimal keys in minimal-reduced and optimal covers [25].

(2) Algorithmically, our framework enables us to optimize lossless, dependency-preserving decompositions into 3NF based on strategies set by our parameters. This replaces arbitrary by strategic choices to improve state-of-the-art since we can declare how ties between redundant 3NF schemata are broken. Experiments with real and synthetic FD sets showcase the quality advancement of schema designs returned by our algorithms over previous work.

(3) Operationally, we evaluate how much update overhead is reduced by our optimizations. We reveal which strategy works best in minimizing update overheads, illustrating performance gains over previous work, and how well optimizations at logical level transcend to integrity maintenance at operational level.

Overall, we establish the first parameterized framework for relational schema design with FDs, intrinsically linking schema optimizations to performance gains at operational level.

**Organization.** We summarize previous work in Section 2 before introducing a parameterized schema design based on the set of minimal keys and covers of non-key FDs in Section 3. Section 4 develops algorithms for parameterized 3NF normalization, while Section **??** presents and analyzes the results of our experiments. We conclude in Section 5 where we also comment on future work. Artifacts are available at *https://github.com/zzxhelloworld/3NF*.

## 2 FUNDAMENTALS AND PREVIOUS WORK

We summarize the background necessary for our framework, starting with concepts from relational databases design [1, 20, 26]. Example 1.1 serves as illustration for the majority of these concepts.

**FDs, Normal Forms, and Normalization.** A *relation schema* is a finite set $R$ of attributes $A$ that have an associated domain $dom(A)$ that contains the set of possible values for the attribute. A *relation* over $R$ is a finite subset $r$ of tuples from the Cartesian product $\prod_{A \in R} dom(A)$. For a tuple $t$ and subset $S \subseteq R$ we use $t[S]$ to denote the projection of $t$ onto $S$. Intuitively, relations formalize tables of records over column names.

A *functional dependency* (FD) is an expression $X \rightarrow Y$ with attribute sets $X, Y \subseteq R$. A relation *satisfies* $X \rightarrow Y$ iff every pair of records with values matching on all attributes of $X$ have also values matching on all attributes of $Y$. An FD $X \rightarrow Y$ is *trivial* iff $Y \subseteq X$.

For an FD set $\mathcal{D} \cup \{d\}$, $\mathcal{D}$ *implies* $d$ if every relation that satisfies all FDs in $\mathcal{D}$ also satisfies $d$. The *semantic closure* of $\mathcal{D}$ is the set $\mathcal{D}^*$ that contains all FDs implied by $\mathcal{D}$, which equals the *syntactic closure* $\mathcal{D}^+$ of $\mathcal{D}$ that contains the FDs that can be inferred from $\mathcal{D}$ by using an axiomatization such as Armstrong's axioms [4]. FD sets $\mathcal{D}$ and $\mathcal{T}$ are *covers* of one another if $\mathcal{D}^+ = \mathcal{T}^+$.

For an FD set $\mathcal{D}$ over $R$, $X \rightarrow Y \in \mathcal{D}^+$ is *minimal* if there is no proper subset $Z \subset X$ such that $Z \rightarrow Y \in \mathcal{D}^+$ holds. $X$ is a *key* of $R$ iff the FD $X \rightarrow R$ is implied by $\mathcal{D}$. In this case, $X \rightarrow R$ is a *key dependency*. An FD $X \rightarrow Y$ is *non-key* if $X \rightarrow R$ is not implied by $\mathcal{D}$. A key $X$ of $R$ is *minimal* if $X \rightarrow R$ is a minimal key dependency, that is, there is no proper subset $Y \subset X$ that is also a key of $R$. $\mathcal{K}$ denotes the set of minimal keys implied by $\mathcal{D}$. An attribute $A \in R$ is *prime* for $\mathcal{D}$ when it is contained in some minimal key of $R$.

FDs encode business rules of the underlying domain, but may also cause data value occurrences that are redundant. Indeed, for a tuple $t$ of a relation $r$ that satisfies an FD set $\mathcal{D}$, the data value occurrence $v = t[A]$ is *redundant* whenever every change of $v$ to a different value $v' \neq v$ results in a relation that violates some FD in $\mathcal{D}$. For a non-trivial FD there is some relation with a redundant data value occurrence if and only if it is not a key dependency.

For an FD set $\mathcal{D}$ over $R$, $(R, \mathcal{D})$ is in *Boyce-Codd Normal Form* (BCNF) iff for every non-trivial FD $X \rightarrow A \in \mathcal{D}^+$, $X$ is a key of $R$. Hence, BCNF characterizes the absence of redundant data values caused by FDs. Data redundancy causes update inefficiency as updates to data values that occur redundantly need be applied to every redundant occurrence. If such values are not updated consistently, integrity faults will occur as violations of FDs.

A *decomposition* of $R$ is a set $\mathbb{D}$ of subsets $S \subseteq R$ such that $\bigcup_{S \in \mathbb{D}} S = R$. $\mathbb{D}$ is *lossless* iff for every relation $r$ over $R$ that satisfies $\mathcal{D}$, it is true that $r = \bowtie_{S \in \mathbb{D}} r[S]$, that is, $r$ is the lossless join over its projections $r[S] = \{t[S] \mid t \in r\}$. $\mathbb{D}$ is *dependency-preserving* iff $\mathcal{D}^+ = (\bigcup_{S \in \mathbb{D}} \mathcal{D}[S])^+$ where $\mathcal{D}[S] = \{X \rightarrow Y \mid X \cup Y \subseteq S \wedge X \rightarrow Y \in \mathcal{D}^+\}$. $\mathbb{D}$ is in BCNF (3NF, see next paragraph) if for every $S \in \mathbb{D}$, $(S, \mathcal{D}[S])$ is in BCNF (3NF). If $\mathbb{D}$ is not dependency-preserving, integrity of some FDs can only be validated on expensive joins of some relations. While lossless decompositions into BCNF always exist, there are some $(R, \mathcal{D})$ for which no lossless, dependency-preserving decomposition into BCNF exists.

Hence, $(R, \mathcal{D})$ is in *Third Normal Form* (3NF) iff for every non-trivial FD $X \rightarrow A \in \mathcal{D}^+$, $X$ is a key of $R$ or $A$ is prime. Indeed, lossless, dependency-preserving decompositions into 3NF are always possible. However, unless it is a BCNF decomposition, data redundancy needs to be tolerated. Classical algorithms [28] compute for input $(R, \mathcal{D})$ a lossless, dependency-preserving decomposition into 3NF that is in BCNF whenever possible. 3NF was shown to guarantee the fewest sources of data redundancy among all lossless, dependency-preserving decompositions [17]. However, they did not aim at breaking ties between 3NF schemata.

**Complexity Results.** It is important to highlight lower complexity bounds associated with computational problems in normalization. For example, the problem of deciding if a given schema $(R, \mathcal{D})$ satisfies 3NF is *NP*-complete, as is the problem of deciding if a given attribute $A \in R$ is prime for $(R, \mathcal{D})$ [5].

The problem of finding the number of minimal keys for $(R, \mathcal{D})$ is #*P*-complete [13], but there is an algorithm for computing the set of minimal keys in time linear in the output [24]. While the number of minimal keys can be exponential in the number of given FDs, this case occurs rarely in practice, so the set of minimal keys can often be computed efficiently [24].

**Parameterizing BCNF.** Recent work has parameterized BCNF by the number $k$ of minimal keys [37]. Here, $k$ represents a measure of both update and query complexity. The larger $k$, the more `UNIQUE` indices need to be maintained during updates but the more queries may benefit from these indices.

Fundamentally, the set of $k$ minimal keys forms a *composite object* of level $k$, characterized by $k$-uniqueness and $k$-dependence [37]. A schema $(R, \mathcal{D})$ is in *Composite Object Normal Form* (CONF) of level $k$ ($k$-CONF) iff every left-hand side of a minimal FD in $\mathcal{D}^+$ belongs to a composite object. In BCNF, all constraints are enforced by minimal keys, and in $k$-CONF, all constraints are enforced by $k$ minimal keys. Hence, $k$ breaks ties between BCNF schemata [37].

Computationally, the best classical algorithm [28] was optimized by eliminating redundant BCNF schemata with larger (smaller, respectively) numbers of minimal keys first, with the aim of minimizing update complexity (maximizing query efficiency, respectively) on those schemata of the decomposition that are in BCNF [37].

However, non-key FDs, which cause the biggest bottleneck for integrity maintenance, have not been considered. This is the contribution of our current work. Indeed, we will parameterize 3NF using $k$ and the number $f$ of FDs in a suitable cover, resulting in $(k, f)$-3NF, where the special case of $f = 0$ captures $k$-CONF.

## 3 FOUNDATIONS FOR PARAMETERIZED 3NF

We will refine the 3NF framework with access to parameters that can optimize normalization. As a byproduct, the difference between

3NF and BCNF becomes quantifiable, too. We will refine the classical 3NF definition by the effort necessary to maintain minimal keys and non-key FDs during updates. Isolating non-key FDs enables us to minimize their overhead during normalization. This minimization transcends from logical to operational level where overheads for integrity maintenance are reduced.

We outline this section intuitively. 3NF is naturally tied to the set $\mathcal{K}$ of minimal keys as the left-hand side $X$ of each non-trivial FD $X \to A$ either contains some minimal key, or the right-hand side $A$ is element of some minimal key (that is, $A$ is prime). Hence, $\mathcal{K}$ is useful for at least two reasons: 1) Each minimal key gives rise to a `UNIQUE` index, and 2) $\mathcal{K}$ enables us to isolate those FDs that cannot be enforced by keys. Partitioning an FD set $\mathcal{D}$ into $\mathcal{K}$ and a suitable cover $\mathcal{F}$ for the set of non-key FDs will formally lead us to defining *the 3NF-core* of $\mathcal{D}$. If that core forms a cover of $\mathcal{D}$, then it is in 3NF. Based on the cardinalities $k$ of $\mathcal{K}$ and $f$ of a minimal-reduced cover for $\mathcal{F}$ we may then compare schemata in 3NF. If $f = 0$, the schema will be in BCNF, and more precisely in $k$-CONF. Hence, $f$ quantifies the overhead of a schema in $(k, f)$-3NF over that in $k$-CONF.

## 3.1 Intransitive Composite Objects

In the special case of BCNF, the set $\mathcal{K}$ forms a composite object of level $k$. Hence, BCNF schemata only require $k$ minimal keys for integrity maintenance during updates. In the general case of 3NF, non-key FDs are also required, but only for RHS attributes that are prime. Hence, we will generalize composite objects to intransitive composite objects, which we define as 3NF-substructures sufficient for maintaining the integrity of their input FD set under updates.

Formally, let $(R, \mathcal{D})$ denote a relation schema $R$ with a set $\mathcal{D}$ of FDs over $R$. Let $\mathcal{T}$ denote a set of FDs over $R$ such that

$$\begin{aligned}\mathcal{T} \quad \subseteq \quad & \{X \subseteq R \mid X \to R \in \mathcal{D}^+\} \cup \\ & \{X \to Y \in \mathcal{D}^+ \mid (X \to R \notin \mathcal{D}^+) \wedge (Y - X \subseteq \mathcal{P})\}\end{aligned}$$

where

$$\mathcal{P} = \{A \in R \mid \exists K \to R \in \mathcal{D}^+ \wedge \forall K' \subset K (K' \to R \notin \mathcal{D}^+) \wedge A \in K\}$$

denotes the set of prime attributes for $\mathcal{D}$. We call $\mathcal{T}$ a *3NF-substructure* of $(R, \mathcal{D})$. While 3NF-substructures meet the requirements of 3NF, they may not enforce all FDs of the input set. This additional feature is special and defined as follows.

*Definition 3.1 (intransitive composite object).* Let $(R, \mathcal{D})$ denote a relation schema $R$ with a set $\mathcal{D}$ of FDs over $R$. Let $\mathcal{T}$ denote a 3NF-substructure of $(R, \mathcal{D})$. We call $\mathcal{T}$ an *intransitive composite object* for $\mathcal{D}$ if and only if the following holds:

- (3NF update completeness)
  For all relations $r$ over $R$ that satisfy $\mathcal{D}$, for all $t \in dom(R)$, if $r \cup \{t\}$ satisfies $\mathcal{T}$, then $r \cup \{t\}$ satisfies $\mathcal{D}$. □

Hence, integrity for $\mathcal{D}$ is retained when all FDs in an intransitive composite object for $\mathcal{D}$ are valid. Next, we illustrate the definitions.

*Example 3.2.* Consider $R = \{E, M, S, T\}$ and $\mathcal{D} = \{ET \to MS, M \to E\}$. The set of minimal keys is $\mathcal{K} = \{ET, MT\}$, $\mathcal{P} = \{E, M, T\}$, and the only non-prime attribute is $S$. Hence, $(R, \mathcal{D})$ is in 3NF. However, $\mathcal{T}' = \{ET, MT, MS \to E\}$ is not an intransitive composite object for $\mathcal{D}$ (because the FD $M \to E$ is not implied by $\mathcal{T}'$). However, $\mathcal{T} = \mathcal{T}' \cup \{M \to E\}$ is an intransitive composite object for $\mathcal{D}$. Indeed, every relation that satisfies $ET$ must also satisfy $ET \to MS$. However, $\{ET, MT\}$ is not a composite object. This can be observed on the following records.

| | *Event* | *Time* | *Manager* | *Status* |
|---|---|---|---|---|
| $t'$ : | Workshop | 21/11/2024 | Sophie | approved |
| $t$ : | Workshop | 19/12/2025 | Sophie | approved |

Indeed, $r = \{t'\}$ satisfies $\mathcal{D}$, and $r \cup \{t\}$ satisfies $ET$ and $MT$, but not $M \to E$. Hence, $\{ET, MT\}$ is not a composite object. □

## 3.2 Intransitive CONF

Intransitive composite objects are not unique. In fact, 3NF-substructures may contain non-minimal keys or non-minimal FDs. Similar to BCNF where the unique composite object is given by $\mathcal{K}$, we will now define a unique 3NF-substructure by partitioning input FD set $\mathcal{D}$ into $\mathcal{K}$ and the set $\mathcal{F}$ of all non-key FDs $X \to A \in \mathcal{D}^+$ where the RHS $A$ is prime and $X$ is minimal such that no proper subset $Y \subset X$ exists where $Y \to A \in \mathcal{D}^+$ holds.

*Definition 3.3.* (3NF-core) For an FD set $\mathcal{D}$ over relation schema $R$, we use $\mathcal{K} = \{X \subseteq R \mid X \to R \in \mathcal{D}^+ \wedge \forall Z \subset X (Z \to R \notin \mathcal{D}^+)\}$ to denote the set of minimal keys implied by $\mathcal{D}$, and

$$\begin{aligned}\mathcal{F} \quad = \quad \{Z \to A \in \mathcal{D}^+ \mid \quad & (Z \to R \notin \mathcal{D}^+) \wedge (A \in \mathcal{P} - Z) \wedge \\ & (\forall Y \subset Z (Y \to A \notin \mathcal{D}^+))\}\end{aligned}$$

to denote the set of non-key minimal FDs with RHS prime attribute implied by $\mathcal{D}$. We call $\mathcal{K} \cup \mathcal{F}$ the *3NF-core* of $\mathcal{D}$. □

The following continues our previous example by removing a non-minimal non-key FD.

*Example 3.4.* For $R = \{E, M, S, T\}$ and $\mathcal{D} = \{ET \to MS, M \to E\}$ from Example 3.2, $\mathcal{T}_c = \mathcal{K} \cup \mathcal{F}$ forms the 3NF-core of $\mathcal{D}$ where $\mathcal{K} = \{ET, MT\}$ and $\mathcal{F} = \{M \to E\}$. □

The 3NF-core is unique and by choosing some minimal-reduced cover for $\mathcal{F}$ we can minimize the number $f$ of non-key FDs required to represent $\mathcal{F}$. This will be used in the next section to optimize 3NF synthesis. For now, however, we will generalize a recent characterization of BCNF by CONF [37] to a characterization of 3NF by 3NF-cores. We have already done all the work since 3NF-cores only need to satisfy 3NF update completeness to capture 3NF.

*Definition 3.5.* (intransitive composite object normal form)
Let $\mathcal{D}$ denote an FD set over relation schema $R$. Then $(R, \mathcal{D})$ is in *intransitive Composite Object Normal Form* (*iCONF*) if and only if the 3NF-core of $\mathcal{D}$ is an intransitive composite object for $\mathcal{D}$. □

Definition 3.5 is independent of how $\mathcal{D}$ is represented. In fact, for every cover $\mathcal{T}$ of $\mathcal{D}$, $(R, \mathcal{D})$ is in iCONF iff $(R, \mathcal{T})$ is in iCONF. We will now illustrate the definition of iCONF on our running example.

*Example 3.6.* For $R = \{E, M, S, T\}$ and $\mathcal{D} = \{ET \to MS, M \to E\}$ from Example 3.4, the 3NF-core $\mathcal{T}_c$ satisfies 3NF update completeness, so $(R, \mathcal{D})$ is in iCONF. □

## 3.3 3NF and iCONF

We will now establish the equivalence between 3NF and iCONF. The main idea is realizing that a 3NF-substructure $\mathcal{T}$ is 3NF update complete if and only if the input FD set is in 3NF and covered by $\mathcal{T}$.

Lemma 3.7. *(Main Lemma)*
*Let $(R, \mathcal{D})$ denote a set $\mathcal{D}$ of FDs over relation schema $R$. Let $\mathcal{T}$ denote a 3NF-substructure of $(R, \mathcal{D})$. Then $\mathcal{T}$ is 3NF update complete if and only if all of the following hold:*

*(1) $\mathcal{D} \subseteq \mathcal{T}^+$*
*(2) $(R, \mathcal{D})$ is in Third Normal Form.* □

We now conclude that the 3NF-core of a schema in 3NF must necessarily be an intransitive composite object.

Corollary 3.8. *If $(R, \mathcal{D})$ is in 3NF, then the 3NF-core $\mathcal{K} \cup \mathcal{F}$ is an intransitive composite object for $(R, \mathcal{D})$.* □

As targeted we can characterize 3NF by iCONF.

Theorem 3.9. *For all relation schemata $R$ and sets $\mathcal{D}$ of FDs over $R$, $(R, \mathcal{D})$ is in 3NF if and only if $(R, \mathcal{D})$ is in iCONF.* □

State-of-the-art results from the literature [37] now emerge as special cases of our new framework. In particular, $k$-CONF is the special case of iCONF where the given FD set $\mathcal{D}$ is covered by the set $\mathcal{K}$ of $k$ minimal keys.

Corollary 3.10. *Let $\mathcal{D}$ denote a set of FDs over relation schema $R$. Then the following statements are equivalent:*

*(1) The 3NF-core of $\mathcal{D}$ over $R$ is covered by $\mathcal{K}$*
*(2) $(R, \mathcal{D})$ is in BCNF with $k$ minimal keys*
*(3) $(R, \mathcal{D})$ is in CONF of level $k$.* □

Next we illustrate how to break ties between different 3NF schemata, which was not possible with previous work.

*Example 3.11.* Consider the following two schemata that belong to the decompositions of $(R, \mathcal{D})$ from Example 1.1:

- $R_3 = EMSV$ and $\mathcal{D}_3$ with non-key FD $VS \rightarrow M$ and two minimal keys $ESV$ and $EMS$
- $R_4 = EMST$ and $\mathcal{D}_4$ with two non-key FDs $MT \rightarrow E$, $ET \rightarrow M$ and three minimal keys $EST$, $EMS$, and $MST$.

Either $(R_3, \mathcal{D}_3)$ or $(R_4, \mathcal{D}_4)$ is redundant, so we can choose which one to include in a decomposition. $\mathcal{D}_3$ contains one minimal non-key FD and two minimal keys, while $\mathcal{D}_4$ contains two minimal non-key FDs and three minimal keys. If we prefer to have fewer non-key FDs, we pick $(R_3, \mathcal{D}_3)$ over $(R_4, \mathcal{D}_4)$, but if we prefer to have more minimal keys, then we pick $(R_4, \mathcal{D}_4)$ over $(R_3, \mathcal{D}_3)$.

The last example illustrates how classical normalization can be optimized by using parameters, such as the numbers of non-key FDs or minimal keys, or even a combination of them to break further ties. Hence, the resulting algorithms target a specific strategy rather than making arbitrary choices between redundant schemata. The next section establishes our framework of parameterized normalization.

# 4 PARAMETERIZED 3NF NORMALIZATION

3NF admits the fewest sources of data redundancy among all lossless, dependency-preserving decompositions [17]. However, not all schemata in 3NF are the same and non-key FDs cause serious update overheads. This presents a great opportunity for normalization, and it is surprising that no previous work has addressed it yet. Recently, BCNF schemata have been classified by the number of minimal keys they exhibit, but 3NF schemata have not received attention yet despite non-key FDs causing the biggest overheads.

We have already shifted as much application semantics of the given FD set $\mathcal{D}$ into the set $\mathcal{K}$ of minimal keys, leaving us with the set $\mathcal{F}$ of minimal, non-key FDs. We will use minimal-reduced (optimal) covers to minimize $\mathcal{F}$ for its cardinality (size, respectively). This enables us to introduce parameterized variants of 3NF, study the computational complexity of parameterized normalization, introduce an algorithm suite and illustrate it on our example.

## 4.1 Parameterized 3NF

The first step is minimizing the set $\mathcal{F}$ of non-key FDs. Maier [25] introduced *minimal-reduced* and *optimal covers*, minimizing the cardinality and size, respectively. While optimal covers provide the smallest total number of attributes occurring in any representation possible for any given FD set, their underlying decision problem is *NP*-complete [25]. In contrast, minimal-reduced covers provide the fewest FDs with no superfluous attribute occurrences in any representation and their computation is quadratic [25].

For a schema $(R, \mathcal{D})$ in 3NF, the set $\mathcal{K}$ of minimal keys for $\mathcal{D}$, and a minimal-reduced cover $\mathcal{F}_c$ for the set of non-key FDs implied by $\mathcal{D}$, we call $(\mathcal{K}, \mathcal{F}_c)$ a *minimal-reduced cover* for the 3NF-core of $(R, \mathcal{D})$. For an optimal cover $\mathcal{F}_s$ for the set of non-key FDs implied by $\mathcal{D}$, we call $(\mathcal{K}, \mathcal{F}_s)$ an *optimal cover* for the 3NF-core of $(R, \mathcal{D})$.

*Definition 4.1.* Let $\mathcal{D}$ be an FD set over $R$, and $k_c, k_s$ be positive integers, and $f_c, f_s$ be non-negative integers. $(R, \mathcal{D})$ is in $(k_c, f_c)$-*3NF* iff $(R, \mathcal{D})$ is in 3NF and there is some minimal-reduced cover $(\mathcal{K}, \mathcal{F}_c)$ for the 3NF-core of $(R, \mathcal{D})$ where $\mathcal{K}$ has cardinality $k_c$, and $\mathcal{F}_c$ has cardinality $f_c$. $(R, \mathcal{D})$ is in $(k_s, f_s)$-*3NF* iff $(R, \mathcal{D})$ is in 3NF and there is some optimal cover $(\mathcal{K}, \mathcal{F}_s)$ for the 3NF-core of $(R, \mathcal{D})$ where $\mathcal{K}$ has size $k_s$, and $\mathcal{F}_s$ has size $f_s$. □

When we write $(k, f)$ we mean $(k, f) \in \{(k_c, f_c), (k_s, f_s)\}$, so we treat the two cases of minimal-reduced and optimal covers simultaneously. The property of $(R, \mathcal{D})$ being in $(k, f)$-3NF is independent of how the semantic constraints are represented. That is, if $\mathcal{D}'$ and $\mathcal{D}$ are covers of one another, then $(R, \mathcal{D})$ is in $(k, f)$-3NF if and only if $(R, \mathcal{D}')$ is in $(k, f)$-3NF. All minimal-reduced covers have the same cardinality, and all optimal covers have the same size. Hence, finding some minimal-reduced or optimal cover, respectively, gives assurance that $f$ is the best achievable.

## 4.2 Breaking Ties between 3NF Schemata

We can now utilize the parameters to articulate strategies, enabling us to break ties between 3NF schemata, and later optimize 3NF synthesis with specific targets. Enabling diverse strategies, we use the parameters $k \in \{k_c, k_s\}$ and $f \in \{f_c, f_s\}$ to define different finite orders $O$ such as i) minimizing or maximizing the ii) cardinality or size of the iii) set of minimal keys or set of non-key FDs. Since we want to minimize $f$, we will not consider $>_f$.

Based on a given order $O$, we fix a ranking $<_O$ that ranks the elements in order of preference by $O$. In $<_O$, the least element is always regarded "best" since our normalization framework aims at minimizing integrity maintenance. For example, if $O = <_f$, then $<_O$ is the natural order $0 < \ldots < n$ on the non-negative integers $0, \ldots, n$; and, if $O = >_k$, then $<_O$ is the reverse natural order $n < n-1 < \ldots < 1 < 0$ on $0, \ldots, n$.

We may choose a primary parameter $l \in \{k, f\}$, and secondary parameter $m \in \{k, f\} - \{l\}$. We then have the following possible orders combining primary and secondary parameters $O = (O_l, O_m) \in \{(<_f, <_k), (<_f, >_k), (<_k, <_f), (>_k, <_f)\}$. The next example makes the ranking $<_O$ explicit for such orders $O$.

*Example 4.2.* The rankings $<_O$ we obtain from different orders $O$ with primary and secondary parameters are:

| $O$ | $<_O$ |
|---|---|
| $(<_f, >_k)$: | $(0, k_0) < \ldots < (0, 1) < \ldots < (f, k_f) < \ldots < (f, 1)$ |
| $(<_f, <_k)$: | $(0, 1) < \ldots < (0, k_0) < \ldots < (f, 1) < \ldots < (f, k_f)$ |
| $(<_k, <_f)$: | $(1, 0) < \ldots < (1, f_1) < \ldots < (k, 0) < \ldots < (k, f_k)$ |
| $(>_k, <_f)$: | $(k, 0) < \ldots < (k, f_k) < \ldots < (1, 0) < \ldots < (1, f_1)$ |

For the first ranking, a schema with FD set of cardinality/size 0, and set of minimal keys with cardinality/size $k_0$ is ranked first. The primary order $<_f$ formalizes that schemata with fewer FDs are prioritized first, and remaining ties are broken by the secondary order $>_k$ that prioritizes schemata with more keys. For the second ranking, a schema with FD set of cardinality/size 0, and set of minimal keys with cardinality/size 1 is ranked first. □

We further apply different orders for parameter $k$ regarding critical ($f > 0$) and non-critical ($f = 0$) schemata. For instance, we may minimize $k$ on BCNF schemata using $<_k$, while maximizing $k$ as secondary parameter on critical schemata by using $>_k$. We write $<_{O'\text{-}BCNF}$ for the ranking of elements from order *O'-BCNF* where $f = 0$, and $<_{O''\text{-}3NF}$ for the ranking of elements from order *O"-3NF* where $f > 0$. We merge $<_{O'\text{-}BCNF}$ and $<_{O''\text{-}3NF}$ into the ranking $<_{O\text{-}BCNF/3NF}$ by defining the largest element (the least preferred) of the former to precede the smallest element (the most preferred) of the latter. For *O'-BCNF* $= <_k$ and *O"-3NF* $= (<_f, >_k)$ we have the following merged ranking $<_{O\text{-}BCNF/3NF}$:

$$1 < \ldots < k < (1, k_1) < \ldots < (1, 2) < \ldots < (f, k_f) < \ldots < (f, 2)$$

where the $k$ smallest elements represent ranking $<_{O'\text{-}BCNF}$ and the remaining elements represent ranking $<_{O''\text{-}3NF}$. Critical schemata have at least two different minimal keys. We now lift any ranking $<_O$ to a ranking $<_O^R$ of critical schemata, thereby breaking ties.

*Definition 4.3.* For a schema $(R, \mathcal{D})$ in 3NF and a ranking $<_O$ for a finite order $O$, we define the *3NF-rank* $r_O^R$ of $(R, \mathcal{D})$ as the smallest rank of any $(k, f)$ in the ranking $<_O$ for which $(R, \mathcal{D})$ is in $(k, f)$-3NF. We further define the order $<_O^R$ on 3NF schemata by $(R, \mathcal{D}) <_O^R (R', \mathcal{D}')$ if and only if $r_O^R <_O r_O^{R'}$. □

Next we illustrate on our running example how different 3NF schemata can be compared with respect to given orders.

*Example 4.4.* Consider the two 3NF schemata from Example 3.11: $R_3 = \{E, M, S, V\}$ and $\mathcal{D}_3$ with non-key FD $VS \rightarrow M$ and 2 minimal keys *ESV* and *EMS*; $R_4 = \{E, M, S, T\}$ and $\mathcal{D}_4$ with 2 non-key FDs $MT \rightarrow E$, $ET \rightarrow M$ and 3 minimal keys *EST*, *EMS*, and *MST*. First, we select the order $O = <_{f_c}$ with ranking $1 < 2$ as a strategy to minimize the number of non-key FDs. Since we have $r_O^{R_3} = 1$ and $r_O^{R_4} = 2$, we obtain $(R_3, \mathcal{D}_3) <_O^R (R_4, \mathcal{D}_4)$. For $O = (<_{f_c}, >_{k_c})$, we obtain $(1, 2) < (2, 3)$ and the same result. Second, we select the order $O' = >_{k_c}$ with ranking $3 < 2$ as a strategy to maximize the number of minimal keys. Since the 3NF-rank of $R_3$ for that order is $r_{O'}^{R_3} = 2$ and that of $R_4$ is $r_{O'}^{R_4} = 1$, we obtain $(R_4, \mathcal{D}_4) <_{O'}^R (R_3, \mathcal{D}_3)$. For $O' = (>_{k_c}, <_{f_c})$, we obtain ranking $(3, 2) < (2, 1)$ and the same result. Both schemata are in 3NF but not in BCNF, so merging the rankings with any $<_{O\text{-}BCNF}$ will not change the result. □

## 4.3 Computational Complexity

We have already discussed that classical 3NF synthesis suffers from likely computational intractability in general. However, it needs to be stressed that the underlying variables are defined at schema level, which means they are typically small. Despite the worst-case bounds, computation in practice is typically efficient. We will illustrate this later by experiments for our parameterized framework. However, it is still important to analyze the computational complexity to understand fundamental limits and set expectations.

The first fundamental problems are to decide whether a given schema is in $(k_c, f_c)$-3NF, or in $(k_s, f_s)$-3NF, respectively.

| Parameterised 3NF | |
|---|---|
| Input: | $(R, \mathcal{D})$, non-negative integers $(k, f) \in \{(k_c, f_c), (k_s, f_s)\}$ |
| Problem: | Decide whether $(R, \mathcal{D})$ is in $(k, f)$-3NF |

As the problem of computing the set of minimal keys is output-polynomial and typically efficient in practice, we may regard this set as additional input, particularly in our framework where we separate this set from that of non-key FDs.

| Parameterised 3NF with set of minimal keys | |
|---|---|
| Input: | $(R, \mathcal{D})$, non-negative integer $f \in \{f_c, f_s\}$<br>Set $\mathcal{K}$ with positive integer $k \in \{k_c, k_s\}$ |
| Problem: | Decide whether $(R, \mathcal{D})$ is in $(k, f)$-3NF |

The previous problems validate whether a schema meets some normal form condition. However, the design problem asks for lossless, dependency-preserving decompositions of the schema. Hence, we need to consider dependencies on subsets of the schema. Consequently, we arrive at the following problem.

| Parameterised 3NF Design | |
|---|---|
| Input: | $(R, \mathcal{D})$, non-negative integers $(k, f) \in \{(k_c, f_c), (k_s, f_s)\}$<br>Set $S \subseteq R$ |
| Problem: | Decide whether $(S, \mathcal{D}[S])$ is in $(k, f)$-3NF |

Next we summarize the results associated with the computational complexity for the decision problems above. An *atomic cover* $\overline{\mathcal{D}}_a$ of FD set $\mathcal{D}$ is the unique set of minimal FDs $X \rightarrow A$ implied by $\mathcal{D}$ [15, 28].

Theorem 4.5. *The parameterised variants associated with 3NF have the following computational complexity.*

1. *Parameterised 3NF is NP-complete for each the cardinality-based and size-based variant.*
2. *Parameterised 3NF with the set of minimal keys is polynomial for the cardinality-based variant, but NP-complete for the size-based variant.*
3. *Parameterised 3NF Design is NP-complete for each the cardinality-based and size-based variant, even if the set of minimal keys on the schema is given, and even if the input is its own atomic cover.* □

These results set expectations for the following sections that will develop an algorithm suite and conduct experiments.

## 4.4 Algorithm Suite

We will now channel our ideas into an algorithm that uses a specific strategy to optimize state-of-the-art normalization.

Previous algorithms guarantee a lossless, dependency-preserving decomposition into 3NF that is in BCNF whenever possible [28]. BCNF is returned iff every critical schema is redundant. Recently [37], this was improved by parameterizing schemata in BCNF by the number of minimal keys they exhibit. This made it possible to break ties between BCNF schemata. However, no work has attempted to break ties between critical schemata. Hence, outputs are not optimized for integrity maintenance of non-key FDs.

We will address this opportunity for optimization by introducing a co-lexical order $<^{D}_{\textit{O-BCNF/3NF}}$ on lossless, dependency-perserving 3NF decompositions, using rankings $<_{\textit{O-BCNF/3NF}}$ of orders *O-BCNF* and *O-3NF* for parameters $k$ and $f$. Our algorithm will return a lossless, dependency-preserving decomposition into 3NF that is optimal for the target order *O-BCNF/3NF*.

Put simply, we minimize the number of critical schemata that are less preferred according to $<_{O-3NF}$. For example, based on $(<_{f_c}, >_{k_c})$, we first keep redundant 3NF schemata with smaller $f_c$ and break further ties by keeping those with larger $k_c$.

We will now introduce the co-lexical order. Intuitively, a decomposition is better than another when the former's worst rank (its least preferred) is better than the latter's worst rank.

*Definition 4.6.* Let $\mathbf{D}$ be a set of lossless, dependency-preserving decompositions of a schema $(R, \mathcal{D})$ into 3NF. For $\mathbb{D} \in \mathbf{D}$, ranking $<_{\textit{O-BCNF/3NF}}$ that spans all values of parameters that occur in any decomposition in $\mathbf{D}$, rank $r$ in $<_{\textit{O-BCNF/3NF}}$, let $\mathcal{S}^{\mathbb{D}}_r$ denote the set of schemata $(S, \mathcal{D}[S]) \in \mathbb{D}$ with rank $r^S_O = r$, that is,

$$\mathcal{S}^{\mathbb{D}}_r = \{(S, \mathcal{D}[S]) \in \mathbb{D} \mid (S, \mathcal{D}[S]) \text{ has rank } r^S_O = r \text{ in } <_{\textit{O-BCNF/3NF}}\}$$

and $n^{\mathbb{D}}_r$ its cardinality, that is, $n^{\mathbb{D}}_r = |\mathcal{S}^{\mathbb{D}}_r|$. For $\mathbb{D}', \mathbb{D}'' \in \mathbf{D}$, $\mathbb{D}'$ is *D-better* than $\mathbb{D}''$ ($\mathbb{D}' <^{D}_{\textit{O-BCNF/3NF}} \mathbb{D}''$) if and only if for the worst rank $r$ in $<_{\textit{O-BCNF/3NF}}$ where $\mathcal{S}^{\mathbb{D}'}_r \neq \mathcal{S}^{\mathbb{D}''}_r$, $\mathcal{S}^{\mathbb{D}'}_r = \emptyset$. □

Alternatively, $\mathbb{D}'$ is *n-better* than $\mathbb{D}''$ ($\mathbb{D}' <^{n}_{\textit{O-BCNF/3NF}} \mathbb{D}''$) if and only if for the worst rank $r$ in $<_{\textit{O-BCNF/3NF}}$ where $n^{\mathbb{D}'}_r \neq 0$ or $n^{\mathbb{D}''}_r \neq 0$, $n^{\mathbb{D}'}_r = 0$. However, if $\mathbb{D}'$ is *D*-better than $\mathbb{D}''$, then $\mathbb{D}'$ is also *n*-better than $\mathbb{D}''$, but not vice versa. As our algorithm will ensure $<^{D}_{\textit{O-BCNF/3NF}}$-optimality, it will also be optimal for $<^{n}_{\textit{O-BCNF/3NF}}$.

For illustration of our definitions, we compare the decompositions of our running example with respect to different orders.

*Example 4.7.* Consider $\mathbf{D} = \{\mathbb{D}_1, \mathbb{D}_2\}$ with $\mathbb{D}_1$ and $\mathbb{D}_2$ from Example 1.1. First, let *O-BCNF* $= <_{k_c}$ and *O-3NF* $= <_{f_c}$, resulting in the following ranking $<_{\textit{O-BCNF/3NF}}$ where *O-BCNF* and *O-3NF* are separated by <<: 1 < 2 < 3 << 1 < 2. The table below shows $\mathcal{S}^{\mathbb{D}}_r$ for every $\mathbb{D} \in \mathbf{D}$ and every rank $r$.

| $\mathbb{D}$ | $\mathcal{S}^{\mathbb{D}}_1$ | $\mathcal{S}^{\mathbb{D}}_2$ | $\mathcal{S}^{\mathbb{D}}_3$ | $\mathcal{S}^{\mathbb{D}}_4$ | $\mathcal{S}^{\mathbb{D}}_5$ |
|---|---|---|---|---|---|
| $\mathbb{D}_1$ | $\emptyset$ | $\{R_2\}$ | $\{R_1\}$ | $\{R_3\}$ | $\emptyset$ |
| $\mathbb{D}_2$ | $\{R_5\}$ | $\emptyset$ | $\{R_1\}$ | $\emptyset$ | $\{R_4\}$ |

The worst rank on which $\mathbb{D}_1$ and $\mathbb{D}_2$ have different schemata is rank 5. As $n^{\mathbb{D}_1}_5 = 0 < 1 = n^{\mathbb{D}_2}_5$, we have $\mathbb{D}_1 <^{D}_{\textit{O-BCNF/3NF}} \mathbb{D}_2$. Second, let *O'-BCNF* $= <_{k_c}$ and *O'-3NF* $= >_{k_c}$, resulting in the following ranking $<_{\textit{O'-BCNF/3NF}}$ where *O-BCNF* and *O-3NF* are separated by <<: 1 < 2 < 3 << 3 < 2. The table below shows $\mathcal{S}^{\mathbb{D}}_r$ for every $\mathbb{D} \in \mathbf{D}$ and every rank $r$.

| $\mathbb{D}$ | $\mathcal{S}^{\mathbb{D}}_1$ | $\mathcal{S}^{\mathbb{D}}_2$ | $\mathcal{S}^{\mathbb{D}}_3$ | $\mathcal{S}^{\mathbb{D}}_4$ | $\mathcal{S}^{\mathbb{D}}_5$ |
|---|---|---|---|---|---|
| $\mathbb{D}_1$ | $\emptyset$ | $\{R_2\}$ | $\{R_1\}$ | $\emptyset$ | $\{R_3\}$ |
| $\mathbb{D}_2$ | $\{R_5\}$ | $\emptyset$ | $\{R_1\}$ | $\{R_4\}$ | $\emptyset$ |

The worst rank on which $\mathbb{D}_1$ and $\mathbb{D}_2$ have different schemata is rank 5. As $n^{\mathbb{D}_1}_5 = 1 > 0 = n^{\mathbb{D}_2}_5$, we have $\mathbb{D}_2 <^{D}_{\textit{O'-BCNF/3NF}} \mathbb{D}_1$. □

**Algorithm 1** iCONF($R$, $\mathcal{D}$, *O-3NF*, *O-BCNF*)

**Require:** $(R, \mathcal{D})$ with FD set $\mathcal{D}$ over schema $R$, 3NF-order *O-3NF*, BCNF-order *O-BCNF*
**Ensure:** Lossless, FD-preserving 3NF decomposition $\mathbb{D}$ of $(R, \mathcal{D})$ that is $<^{D}_{\textit{O-BCNF/3NF}}$-optimal

1: Compute the atomic cover $\overline{\mathcal{D}}_a$ of $\mathcal{D}$ [15, 28]
2: $\mathcal{D}_a \leftarrow \overline{\mathcal{D}}_a$, $\mathcal{D}_c \leftarrow \emptyset$
3: **for all** $X \rightarrow A \in \mathcal{D}_a$ **do**
4: **for all** $Y \rightarrow B \in \mathcal{D}_a (YB \subseteq XA \wedge XA \not\subseteq Y^+_{\mathcal{D}})$ **do**
5: Compute $k_{XA}$ and $f_{XA}$ in target 3NF-core of $\mathcal{D}_a[XA]$
6: $\mathcal{D}_c \leftarrow \mathcal{D}_c \cup \{(X \rightarrow A, k_{XA}, f_{XA})\}$
7: $\mathbb{D} \leftarrow \emptyset$;
8: **for all** $(X \rightarrow A, k_{XA}, f_{XA}) \in \mathcal{D}_c$ in reverse $<_{\textit{O-3NF}}$-ranks **do**
9: **if** $\overline{\mathcal{D}}_a \backslash \{X \rightarrow A\} \not\models X \rightarrow A$ **then**
10: $\mathbb{D} \leftarrow \mathbb{D} \cup \{(XA, \mathcal{D}_a[XA], k_{XA}, f_{XA})\}$
11: **else**
12: $\overline{\mathcal{D}}_a \leftarrow \overline{\mathcal{D}}_a \backslash \{X \rightarrow A\}$
13: **for all** $X \rightarrow A \in \overline{\mathcal{D}}_a \backslash \mathcal{D}_c$ **do**
14: $k_{XA} \leftarrow |\{K \mid K \rightarrow XA \in \mathcal{D}_a[XA])^+\}|$ using [24]
15: $\mathcal{D}_a \leftarrow (\mathcal{D}_a \backslash \{X \rightarrow A\}) \cup \{(X \rightarrow A, k_{XA})\}$
16: **for all** $(X \rightarrow A, k_{XA}) \in \overline{\mathcal{D}}_a \backslash \mathcal{D}_c$ in reverse $<_{\textit{O-BCNF}}$-ranks **do**
17: **if** $\overline{\mathcal{D}}_a \backslash \{X \rightarrow A\} \not\models X \rightarrow A$ **then**
18: $\mathbb{D} \leftarrow \mathbb{D} \cup \{(XA, \mathcal{D}_a[XA], k_{XA})\}$
19: **else**
20: $\overline{\mathcal{D}}_a \leftarrow \overline{\mathcal{D}}_a \backslash \{X \rightarrow A\}$
21: Remove all $(S, \mathcal{D}_a[S]) \in \mathbb{D}$ if $\exists (S', \mathcal{D}_a[S']) \in \mathbb{D} (S \subseteq S')$
22: **if** there is no $(R', \mathcal{D}') \in \mathbb{D}$ where $R' \rightarrow R \in \mathcal{D}^+_{\mathfrak{A}}$ **then**
23: Choose a minimal key $K$ for $R$ with respect to $\mathcal{D}$
24: $\mathbb{D} \leftarrow \mathbb{D} \cup \{(K, \mathcal{D}_a[K], 1)\}$
25: **Return**($\mathbb{D}$)

Algorithm 1 computes a lossless, dependency-preserving 3NF decomposition that is $<^{D}_{\textit{O-BCNF/3NF}}$-optimal for its input. It starts with the *atomic cover* $\overline{\mathcal{D}}_a$ of input $\mathcal{D}$ in line 1. Line 2 creates a copy of the atomic cover from which redundant FDs may be removed later, while the original closure checks for critical FDs (line 4).

The part in lines (3-6) is called *Critical*. It assembles the set $\mathcal{D}_c$ of critical schemata (line 6) and values of their parameters $k$ and $f$ (line 5) that determine the ranking $<_{\textit{O-3NF}}$ for input order *O-3NF*.

The part in lines (7-12) is called *Opt-3NF* where FDs that cause critical schemata are evaluated in reverse ranks of $<_{\textit{O-3NF}}$ (line 8). If an FD is redundant (line 12), we do not need the schema. Processing these FDs from worst to best ensures we eliminate remaining worst schemata when possible. If the FD is not redundant (line 9), the schema is added (line 10). *Opt-3NF* ensures $<_{\textit{O-3NF}}$-optimality.

The part in lines (13-15) is called *Key*. It computes the number of minimal keys for non-critical schemata. The computation is done in output-polynomial time [24].

The part in lines (16-20) is called *Opt-BCNF*. It loops through non-critical FDs in reverse ranks of $<_{O\text{-}BCNF}$ (line 16), eliminates the FD when redundant (line 20) or add its schema otherwise (lines 17/18). This eliminates redundant BCNF schemata following *O-BCNF*.

The part in line 21 is called *Subset*. Here, we remove any schema that is a subset of another one.

Finally, the part in lines (22-24) is called *Lossless*. It adds a minimal key to the output that ensures the decomposition is lossless when returned in line 25. This concludes the explanation of Algorithm 1.

The following result improves the state-of-the-art [37], which returns a lossless, dependency-preserving decomposition into 3NF that is in BCNF if possible and $<_{k_c}$-optimal in that case.

Theorem 4.8. *On input* $(R, \mathcal{D}$, *O-3NF*, *O-BCNF*$)$*, Algorithm 1 returns a lossless, dependency-preserving decomposition into 3NF that is* $<^D_{O\text{-}BCNF/3NF}$*-optimal.* □

Unlike any previous work, Algorithm 1 optimizes critical schemata and also does so with respect to any given target strategy.

## 4.5 Running Example

We illustrate Algorithm 1 by showing how the decompositions in Example 1.1 were derived. Firstly, we choose $O_{3NF} = <_{f_c}$ and $O_{BCNF} = <_{k_c}$ as input. Hence, our strategy is to minimize the number of non-key FDs on critical schemata and the number of minimal keys on BCNF schemata.

We start with the atomic cover $EMS \to V$, $SV \to M$, $EMS \to T$, $MT \to E$, $ET \to M$, $MST \to V$, $SET \to V$, $ESV \to T$, $STV \to E$.

Critical schemata are $(R_3, \mathcal{D}_3)$ with 1 non-key FD and 2 minimal keys, $(R_4, \mathcal{D}_4)$ with 2 non-key FDs and 3 minimal keys, and $(R_6 = MSTV, \mathcal{D}_6)$ with 1 non-key FD $SV \to M$ and 2 minimal keys $STV$ and $MST$. BCNF schemata are $(R_5, \mathcal{D}_5)$ with 1 minimal key, $(R_2, \mathcal{D}_2)$ with 2 minimal keys, and $(R_1, \mathcal{D}_1)$ with 3 minimal keys.

For $O_{3NF} = <_{f_c}$, the critical schemata are ordered as: $(R_3, \mathcal{D}_3) =^R_{f_c} (R_6, \mathcal{D}_6) <^R_{f_c} (R_4, \mathcal{D}_4)$. As the $R_4$-generating FD $EMS \to T$ is redundant, $(R_4, \mathcal{D}_4)$ is not required. The $R_3$-generating FD $EMS \to V$ is not redundant now, so the schema $(R_3, \mathcal{D}_3)$ is added to the decomposition. Next, the $R_6$-generating FD $MST \to V$ is still redundant, so the schema $(R_6, \mathcal{D}_6)$ is not required. For $O_{BCNF} = <_{k_c}$, the BCNF schemata are ordered as: $(R_5, \mathcal{D}_5) <^R_{k_c} (R_2, \mathcal{D}_2) <^R_{k_c} (R_1, \mathcal{D}_1)$ but the $R_1$-generating FD $EST \to V$ is not redundant, and neither the $R_2$-generating FDs $ET \to M$ or $MT \to E$, so $(R_1, \mathcal{D}_1)$ and $(R_2, \mathcal{D}_2)$ are added to the decomposition. Indeed, the same is true for the $R_5$-generating FD $SV \to M$, so $(R_5, \mathcal{D}_5)$ is added, too. However, $R_5 \subseteq R_3$, so $(R_5, \mathcal{D}_5)$ is removed from the decomposition. The final decomposition $\mathbb{D}_1 = \{(R_1, \mathcal{D}_1), (R_2, \mathcal{D}_2), (R_3, \mathcal{D}_3)\}$ is $<^D_{O\text{-}3NF/BCNF}$-optimal.

Using $O'_{3NF} = >_{k_c}$ as input, the critical schemata are ordered as: $(R_4, \mathcal{D}_4) <^R_{k_c} (R_3, \mathcal{D}_3) =^R_{k_c} (R_6, \mathcal{D}_6)$, the $R_3$-generating FD $EMS \to V$ is redundant and thus removed, and the $R_6$-generating FD $MST \to V$ is redundant and also removed. However, the $R_4$-generating FD $EMS \to T$ is not redundant, and the schema $(R_4, \mathcal{D}_4)$ is added to the decomposition. Based on $O'_{BCNF} = <_{k_c}$, the BCNF schemata are ordered as before: $(R_5, \mathcal{D}_5) <^R_{k_c} (R_2, \mathcal{D}_2) <^R_{k_c} (R_1, \mathcal{D}_1)$, but the $R_1$-generating FD $EST \to V$ is not redundant, and neither the $R_2$-generating FDs $ET \to M$ or $MT \to E$, so $(R_1, \mathcal{D}_1)$ and $(R_2, \mathcal{D}_2)$ are added to the decomposition. Indeed, the same is true for the $R_5$-generating FD $SV \to M$, so $(R_5, \mathcal{D}_5)$ is added, too. However, $R_2 \subseteq R_4$, so $(R_2, \mathcal{D}_2)$ is removed. The final decomposition $\mathbb{D}_2 = \{(R_1, \mathcal{D}_1), (R_4, \mathcal{D}_4), (R_5, \mathcal{D}_5)\}$ is $<^D_{O'\text{-}3NF/BCNF}$-optimal.

Through experiments we seek answers to the following questions.

*(E1)* How do keys and non-key FDs affect performance?
*(E2)* How much can we improve state-of-the-art algorithms?
*(E3)* How much update overheads can we save?

Answers to *(E1)* will motivate our research more and further inform the strategies we report on. *(E2)* focuses on the efficacy of our normalization algorithms over previous work at the logical level, in terms of meeting their design goals and the time they need to do that. *(E3)* investigates how well our optimizations transcend from logical to operational level. In particular, we will see which strategy works best for lowering overheads of integrity maintenance.

## 4.6 Experimental set up

Our algorithms were implemented in Java, Version 17.0.7, and run on a 12th Gen Intel(R) Core(TM) i7-12700, 2.10GHz, with 128GB RAM, 1TB SSD, and Windows 10. We used MySQL 8.0.29.

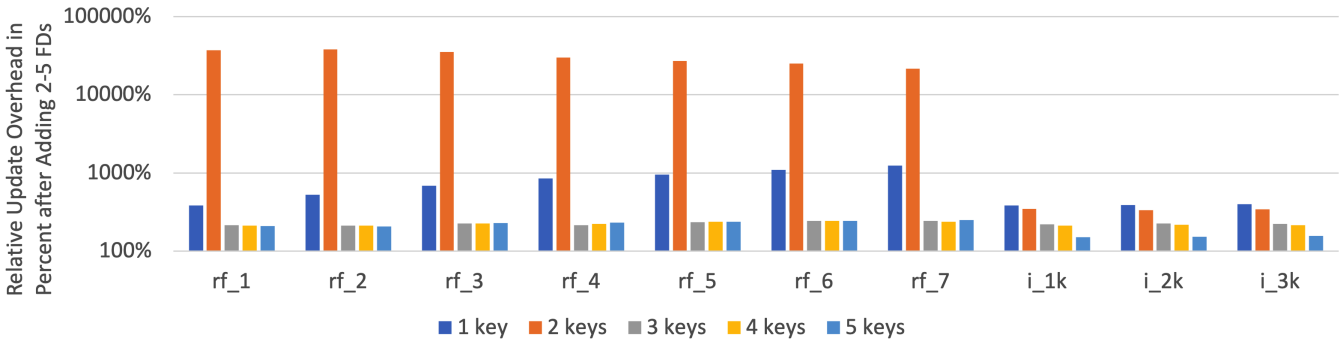
**Data sets.** Apart from perfect synthetic data for realistic schemata and sets of FDs, we used FDs mined from 12 real-world benchmark data plus TPC-H [1]. These have served as benchmarks for profiling data dependencies [29, 30, 33], but also for experiments in previous work [37]. Hence, we cannot only compare our algorithms to state-of-the-art, but also analyze them on instances with tens, hundreds, and thousands of FDs to test scalability.

**Algorithms.** We implemented our new algorithms, and used the mining of FDs [33] and generation of Armstrong relations [27]. As in previous work, we used the cardinality of key sets and FDs sets. We will denote by *iConf-fk* (A1) and *iConf-f* (A2) our Algorithm 1 where *O-3NF* is $(<_{f_c}, >_{k_c})$ and $<_{f_c}$, respectively, and where *O-BCNF=* $<_{k_c}$ in both cases. We will compare performance of these to our implementations of previous work: *Conf* (A3) [37] (which is based on *O-BCNF=* $<_{k_c}$), BC-Cover (A4) [28], and Synthesis (A5) [8].

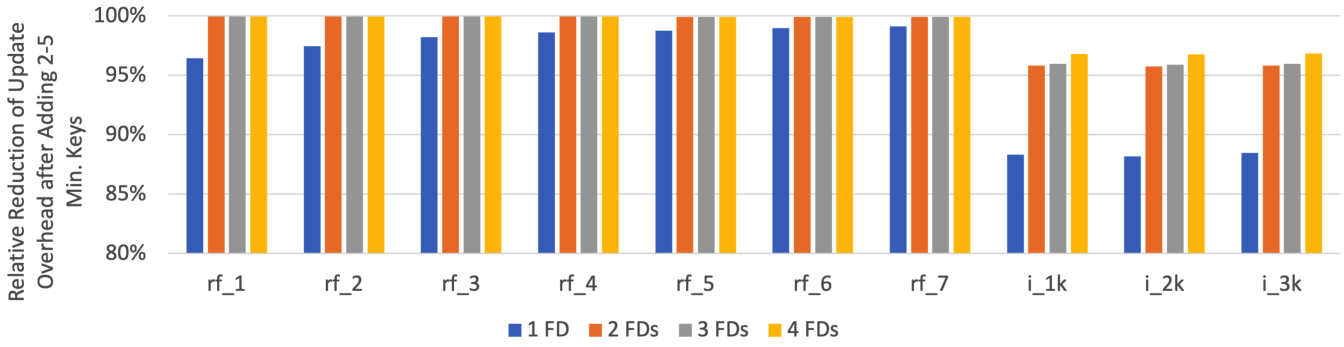
## 4.7 How do keys and FDs affect performance?

We study how adding keys or FDs affects performance of the TPC-H benchmark (scaling factor 0.1) with 22 queries, 7 refresh and 3 insert (adding 1k, 2k, and 3k of records) operations. We have chosen the five most sensible minimal keys and five most sensible non-key FDs we mined from each table [33]. For *each* table, keys are declared as `UNIQUE` constraints and FDs by triggers. Each bar in Fig. 2a and 2c displays, in percent, the relative speed of an operation, averaged over 30 runs, after adding $f = 1, \ldots, 4$ FDs, when compared to the same operation run with just 1 FD, in each case of having $k = 1, \ldots, 5$ keys present. Fig. 2b and 2d reverse the roles of keys and FDs, so we add $k = 1, \ldots, 4$ keys to each case of having $f = 1, \ldots, 5$ FDs and 1 key present.

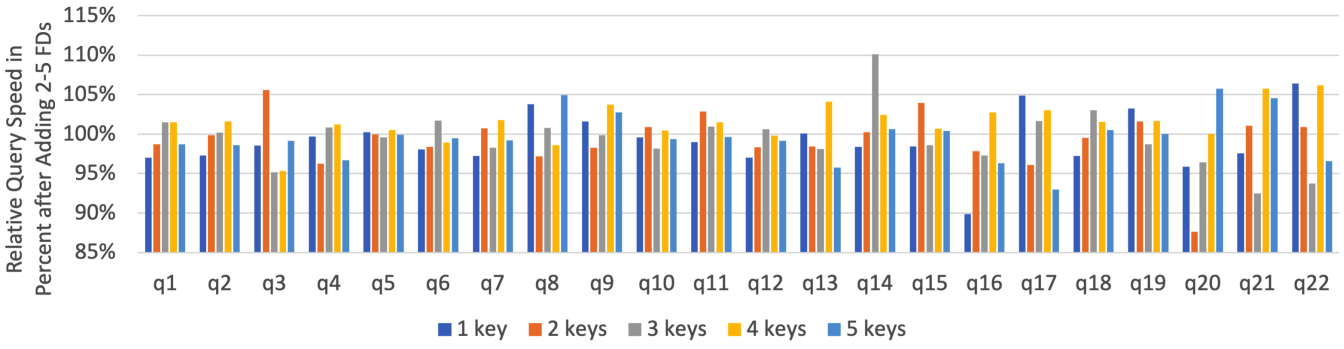
[1] hpi.de/naumann/projects/repeatability/data-profiling/fds.html



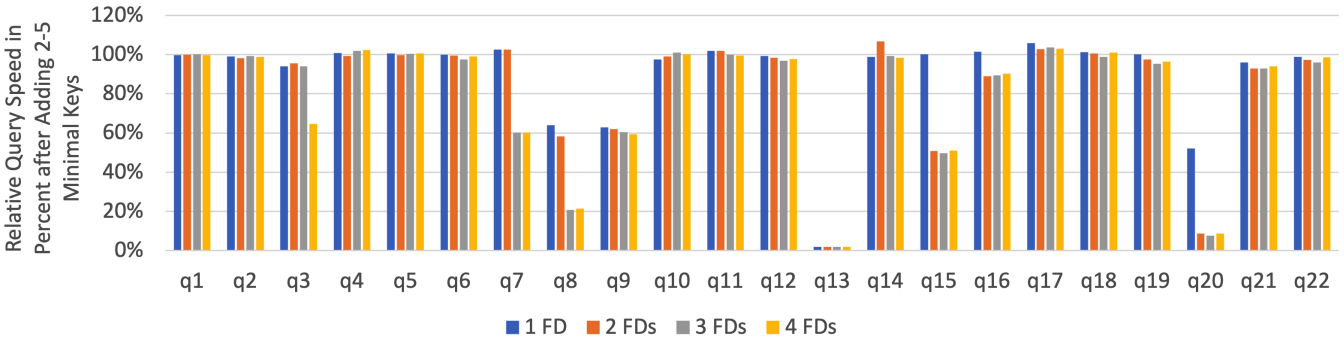

(a) Update Overheads From Adding More FDs




(b) Reducing Update Overheads by Adding More Keys




(c) Relative Query Speed after Adding More FDs




(d) Relative Query Speed after Adding More Keys

**Figure 2: Impact of More FDs and More Keys on Update and Query Performance over TPC-H**

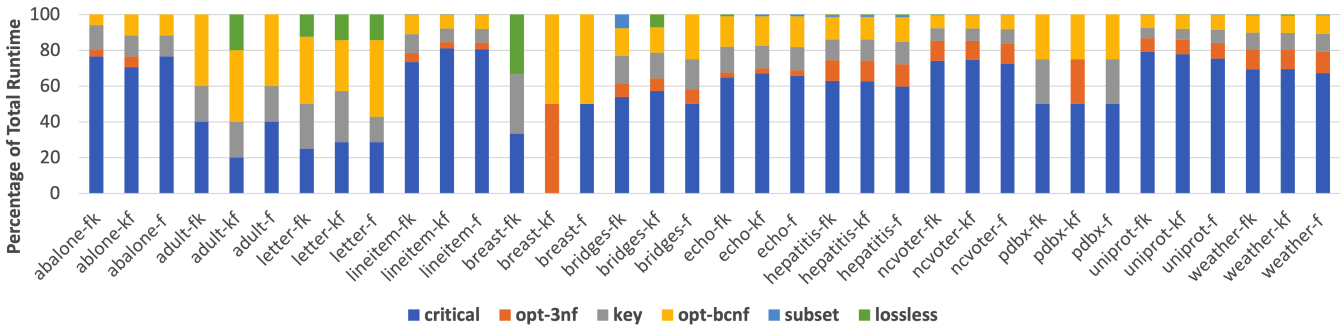
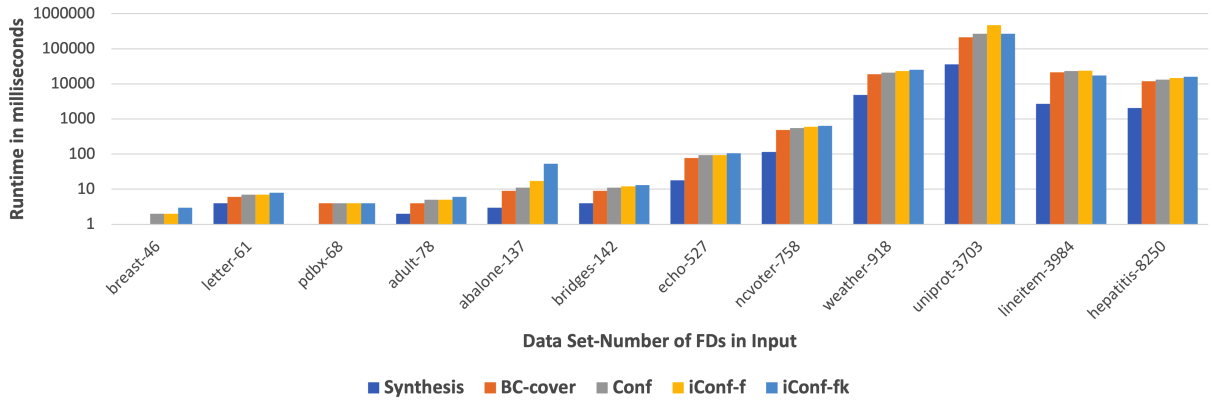
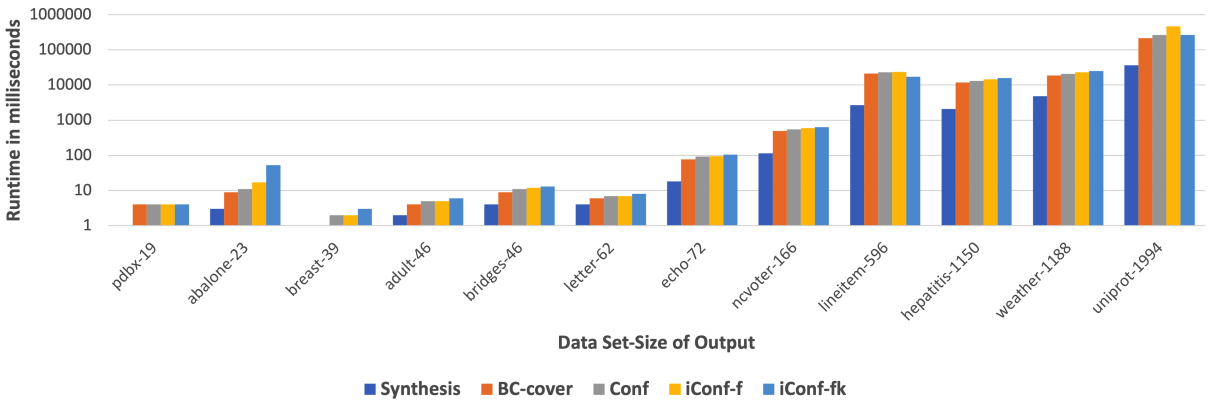



**Figure 3: Breakdown of (A1-5) in Percent of Total Runtime**




**(a) Runtime in Number of FDs (see ending of data set name)**




**(b) Runtime in Size of Output (see ending of data set name)**

**Figure 4: Runtime in Input Size and Output Size**

Figure 2a shows how adding FDs to a number of minimal keys incurs update overheads. The average overhead across the refreshs and constraint sets is more than 6400%, and across the inserts it is more than 264%. The cases where $k = 3, 4, 5$ keys are present scales update performance, no matter how many non-key FDs are added.

Figure 2b shows how adding minimal keys to a number of non-key FDs reduces update overheads. The average reduction across the refreshs and constraint sets is more than 99.4%, and across the inserts it is more than 94%. Having a few more FDs present incurs huge update overheads, but adding some minimal keys scales integrity maintenance well.

Figure 2c shows how adding FDs to a number of minimal keys affects query performance. The average speed across the 22 queries is just below 99.8%. So while some queries are affected, on average there is little impact on query performance resulting from FDs.

| | Characteristics | | | Time of Algorithms (in ms) | | | | |
|---|---|---|---|---|---|---|---|---|
| Data set | *#R* | *#C* | *#FD* | *Synthesis* | *BC-Cover* | *Conf* | *iConf-f* | *iConf-fk* |
| abalone | 4,177 | 9 | 137 | 3 | 9 | 11 | 17 | 53 |
| adult | 48,842 | 14 | 78 | 2 | 4 | 5 | 5 | 6 |
| breast | 699 | 11 | 46 | 1 | 1 | 2 | 2 | 3 |
| bridges | 108 | 13 | 142 | 4 | 9 | 11 | 12 | 13 |
| echo | 132 | 13 | 527 | 18 | 77 | 93 | 94 | 105 |
| hepatitis | 155 | 20 | 8,250 | 2064 | 11,797 | 13,134 | 14,551 | 15,865 |
| letter | 20,000 | 17 | 61 | 4 | 6 | 7 | 7 | 8 |
| lineitem | 6,001,215 | 16 | 3,984 | 2,698 | 21,269 | 23,056 | 23,696 | 17,364 |
| ncvoter | 1,000 | 19 | 758 | 115 | 489 | 547 | 595 | 640 |
| pdbx | 17,305,799 | 13 | 68 | 1 | 4 | 4 | 4 | 4 |
| uniprot | 512,000 | 30 | 3,703 | 36,238 | 213,958 | 266,825 | 468,059 | 266,257 |
| weather | 262,920 | 18 | 918 | 4,796 | 18,824 | 20,925 | 23,184 | 25,140 |

**Table 1: Data Sets and Runtimes (in ms) by Data Sets**

Figure 2d shows how adding minimal keys to a given number of non-key FDs impacts query performance. The average speed is just below 83.6%, so a speed up of over 14.4%. This quantifies our expectations that the `UNIQUE` index resulting from keys does improve query speed. Having FDs present does not affect query performance much, but adding minimal keys and their `UNIQUE` indices does have a noticeable impact.

**Conclusion.** Addressing *(E1)*, our mini and TPC-H studies quantify the need for a framework that separates non-key FDs from minimal keys to access parameters, and that aligns schema design closer with the performance at operational level. The experiments show that (1) the number of non-key FDs is a valuable parameter to minimize, (2) the number of minimal keys is a valuable parameter that affects update and query performance, and (3) relying exclusively on FDs for integrity maintenance is infeasible.

## 4.8 How good are our algorithms?

All algorithms return a lossless, dependency-preserving (LD-) decomposition into 3NF. If an LD-decomposition into BCNF exists, *BC-Cover* will find one. *Conf* improves *BC-Cover* by returning an LD-decomposition into $k$-CONF for the lowest $k$ possible. *iConf-f* guarantees that the LD-decomposition into 3NF is optimized for *O-3NF*=$<_{f_c}$ and *O-BCNF*=$<_{k_c}$. *iConf-f* guarantees that the maximum number of non-key FDs across all output schemata is minimized; and if an LD-decomposition into BCNF exists (that is, if that maximum number is zero), then *iConf-f* returns the same result as *Conf*. *iConf-fk* breaks remaining ties between 3NF schemata with the same number of non-key FDs by prioritizing larger numbers of minimal keys. Later, we will discuss other variants where *O-3NF*=$(>_{k_c}, <_{f_c})$ or *O-3NF*=$(<_{f_c}, <_{k_c})$.

*4.8.1 Runtime analysis.*

*4.8.2 Runtime analysis.* Part of Table 1 shows for each of the 12 data sets, their name, numbers of rows ($\sharp R$) and columns ($\sharp C$), and the number of FDs in the atomic cover for the set of FDs exhibited by the data sets ($\sharp FD$). In line with previous work, we uniformly regard two missing values as a match. Regarding them as no match leads to different FDs, but overall observations do not change.

In addressing *E2*, Table 1 reports the total run time of Algorithm 1 and the time spent on its steps indicated before, for each data set. Based on the different characteristics of our data sets, run times differ quite significantly, too. Foremost, all algorithms run efficiently despite some large input sizes. In fact, the longest run times were exhibited on *uniprot*, with less than 5 minutes except for *iConf-f* which took less than 8 minutes. All other FD sets did not take longer than 30 seconds. There is an order of magnitude difference between the run time of *Synthesis* and the remaining algorithms, which quantifies their effort to find an LD-decomposition into BCNF, not attempted by *Synthesis*. Unsurprisingly, there are runtime overheads for optimizations, in particular for *iConf-f* over *Conf*. While the difference is mostly insignificant, it is less than 1.5 seconds on *hepatitis*, less than a second on *lineitem*, less than 3 seconds on *weather*, and less than 3.5 minutes in the most extreme case *uniprot*. Schema design is a critical task, and our experiments quantify the runtime overheads incurred by optimizing schema design algorithms for the target strategy.

Figure 3 shows how much percent each step of the algorithm contributes to the overall run time. *Critical* consumes most of the time due to computing critical schemata, minimal covers for FDs projected on their schemata, and finding all minimal keys. The optimizations for 3NF and BCNF consume significant time due to large numbers of FDs and keys, and so does *Keys* when computing all minimal keys on the BCNF schemata.

Figure 4 quantifies the expected exponential dependence of the run time of all algorithms on the number of input FDs, and the number of schemata in the output, respectively.

*4.8.3 Output analysis.* For *E3*, Table 2 reports the results of applying (A1)-(A5) to the FDs mined from the 12 data sets. For each data set, we list the *Alg*orithms for which we report results (joining algorithms with the same results), the total number *Size* of relation schemata in the output, split into the number *BCNF* of BCNF schemata and the number *3NF* of critical schemata, and the average numbers of minimal keys *#Keys* (non-key *#FDs*) across schemata in BCNF (3NF). Under *Distribution*, $n_i$ denotes how many BCNF (3NF) schemata exhibit precisely $s_i$ minimal keys (non-key FDs).

By design, *BC-Cover* (A4) generates never more and usually fewer critical schemata than *Synthesis* (A5), in percent of 3NF schemata. By design, *Conf* and *BC-Cover* produce the same schemata in 3NF, but *Conf* optimizes BCNF schemata for $<_{k_c}$. In fact, *Conf* produces decompositions with better D-ranks than *BC-Cover* on abalone, echo, hepatitis, lineitem, ncvoter and uniprot; and fewer schemata at the lowest ranks where they differ on bridges, pdbx, and weather.

As our main target, *iConf-f* (A2) optimizes the 3NF distribution over *Conf* (A3). This is most visible on *lineitem* where (A2) has eliminated 3NF schemata with 10, 9 and 5 FDs in them. Similarly, (A2) produces decompositions that are *D*-better than those generated by (A3) on ncvoter, uniprot, and weather; and fewer schemata at the lowest rank where they differ on abalone and hepatitis.

Variant *iConf-fk* (A1) is an optimization that retains those redundant 3NF schemata that are tied using *iConf-f* (A2) but exhibit more minimal keys. Compared to (A2), (A1) eliminates a few more redundant 3NF schemata due to this strategy, such as on abalone and weather. For ncvoter, the 3NF distributions coincide but the schemata differ, making it possible to eliminate the BCNF schema with the most minimal keys compared to (A2) and (A3). On lineitem, however, (A1) uses an additional 3NF schema over (A2), but has fewer BCNF-schemata on some ranks.

**Conclusion.** Our algorithms optimize logical schema design for a target strategy. The experiments illustrate what our algorithms

| | | Decomposition | | | Schema in BCNF | | Schema in 3NF | |
|---|---|---|---|---|---|---|---|---|
| Data set | Alg | Size | BCNF | 3NF | #Keys | Distribution | #FDs | Distribution |
| abalone | A1 | 26 | 22 | 4 | 1.64 | [3:1,2:12,1:9] | 2 | [4:1,2:1,1:2] |
| | A2 | 23 | 18 | 5 | 1.61 | [2:11,1:7] | 1.8 | [4:1,2:1,1:3] |
| | A3 | 21 | 16 | 5 | 1.81 | [3:2,2:9,1:5] | 2.4 | [4:2,2:1,1:2] |
| | A4 | 20 | 15 | 5 | 2.07 | [5:1,3:2,2:8,1:4] | 2.4 | [4:2,2:1,1:2] |
| | A5 | 21 | 14 | 7 | 1.93 | [3:2,2:9,1:3] | 2.29 | [4:2,3:1,2:1,1:3] |
| adult | A1-5 | 46 | 46 | 0 | 1.02 | [2:1,1:45] | | |
| breast | A1-5 | 39 | 37 | 2 | 1.03 | [2:1,1:36] | 1 | [1:2] |
| bridges | A1-3 | 46 | 39 | 7 | 1.15 | [3:1,2:4,1:34] | 1 | [1:7] |
| | A4 | 44 | 37 | 7 | 1.19 | [3:1,2:5,1:31] | 1 | [1:7] |
| | A5 | 43 | 34 | 9 | 1.21 | [3:1,2:5,1:28] | 1 | [1:9] |
| echo | A1-3 | 72 | 65 | 7 | 1.46 | [3:6,2:18,1:41] | 1.14 | [2:1,1:6] |
| | A4-5 | 72 | 65 | 7 | 1.58 | [4:1,3:9,2:17,1:38] | 1.14 | [2:1,1:6] |
| hepatitis | A1-2 | 1150 | 842 | 308 | 1.12 | [3:9,2:82,1:751] | 1.88 | [10:1,9:1,8:1,7:3,6:4,5:12,4:12,3:31,2:63,1:180] |
| | A3 | 1130 | 826 | 304 | 1.12 | [3:10,2:82,1:734] | 1.97 | [10:1,9:1,8:3,7:4,6:5,5:12,4:13,3:27,2:66,1:172] |
| | A4 | 1123 | 819 | 304 | 1.13 | [4:1,3:10,2:80,1:728] | 1.97 | [10:1,9:1,8:3,7:4,6:5,5:12,4:13,3:27,2:66,1:172] |
| | A5 | 1113 | 784 | 329 | 1.1 | [4:1,3:6,2:66,1:711] | 1.93 | [10:1,9:1,8:3,7:4,6:6,5:13,4:14,3:27,2:67,1:193] |
| letter | A1-5 | 62 | 62 | 0 | 1 | [1:62] | | |
| lineitem | A1 | 590 | 560 | 30 | 1.39 | [15:1,10:1,6:3,5:3,4:5,3:23,2:105,1:419] | 1.4 | [4:1,2:9,1:20] |
| | A2 | 596 | 567 | 29 | 1.39 | [15:1,10:1,6:4,5:3,4:5,3:23,2:105,1:425] | 1.41 | [4:1,2:9,1:19] |
| | A3 | 587 | 558 | 29 | 1.38 | [15:1,10:1,6:3,5:3,4:5,3:22,2:104,1:419] | 2.62 | [10:2,9:1,5:1,4:1,3:2,2:10,1:12] |
| | A4 | 562 | 533 | 29 | 2.32 | [15:1,11:1,10:2,9:9,8:9,7:19,6:26,5:26,4:26,3:26,2:46,1:342] | 2.62 | [10:2,9:1,5:1,4:1,3:2,2:10,1:12] |
| | A5 | 531 | 466 | 65 | 2.29 | [11:1,10:1,9:8,8:9,7:19,6:21,5:25,4:24,3:18,2:30,1:310] | 2.28 | [10:3,9:1,5:1,4:4,3:6,2:20,1:30] |
| ncvoter | A1 | 166 | 145 | 21 | 1.19 | [2:27,1:118] | 1.19 | [3:1,2:2,1:18] |
| | A2 | 166 | 145 | 21 | 1.2 | [3:1,2:27,1:117] | 1.19 | [3:1,2:2,1:18] |
| | A3 | 168 | 147 | 21 | 1.2 | [3:1,2:28,1:118] | 1.29 | [4:1,2:3,1:17] |
| | A4 | 162 | 141 | 21 | 1.28 | [4:2,3:5,2:23,1:111] | 1.29 | [4:1,2:3,1:17] |
| | A5 | 154 | 123 | 31 | 1.24 | [4:1,3:3,2:20,1:99] | 1.35 | [4:1,2:8,1:22] |
| pdbx | A1-3 | 19 | 14 | 5 | 1.21 | [2:3,1:11] | 1 | [1:5] |
| | A4-5 | 18 | 13 | 5 | 1.31 | [2:4,1:9] | 1 | [1:5] |
| uniprot | A1 | 1992 | 1576 | 416 | 1.13 | [4:1,3:5,2:187,1:1383] | 1.48 | [14:1,8:1,7:2,5:1,4:13,3:17,2:91,1:290] |
| | A2 | 1994 | 1578 | 416 | 1.13 | [4:1,3:5,2:187,1:1385] | 1.48 | [14:1,8:1,7:2,5:1,4:13,3:17,2:91,1:290] |
| | A3 | 1981 | 1564 | 417 | 1.13 | [4:1,3:5,2:186,1:1372] | 1.56 | [14:1,11:1,8:1,7:2,5:3,4:17,3:19,2:91,1:282] |
| | A4 | 1946 | 1529 | 417 | 1.16 | [5:1,4:2,3:10,2:207,1:1309] | 1.56 | [14:1,11:1,8:1,7:2,5:3,4:17,3:19,2:91,1:282] |
| | A5 | 1923 | 1443 | 480 | 1.13 | [5:1,4:1,3:8,2:169,1:1264] | 1.53 | [14:1,11:1,8:1,7:2,5:3,4:17,3:23,2:103,1:329] |
| weather | A1 | 1186 | 796 | 390 | 1.2 | [6:1,5:1,4:3,3:11,2:120,1:660] | 2.47 | [7:5,6:10,5:27,4:46,3:68,2:112,1:122] |
| | A2 | 1188 | 796 | 392 | 1.2 | [6:1,5:1,4:3,3:11,2:119,1:661] | 2.46 | [7:5,6:10,5:27,4:46,3:68,2:112,1:124] |
| | A3 | 1162 | 770 | 392 | 1.21 | [6:1,5:1,4:3,3:11,2:119,1:635] | 2.56 | [9:1,7:7,6:11,5:27,4:50,3:67,2:115,1:114] |
| | A4 | 1154 | 762 | 392 | 1.25 | [6:2,5:3,4:3,3:16,2:126,1:612] | 2.56 | [9:1,7:7,6:11,5:27,4:50,3:67,2:115,1:114] |
| | A5 | 1127 | 702 | 425 | 1.19 | [4:1,3:12,2:104,1:585] | 2.53 | [9:1,7:8,6:12,5:27,4:53,3:74,2:120,1:130] |

**Table 2: Properties of Output Decomposition based on FD sets mined from Data Sets**

achieve over state-of-the-art, such as minimizing non-key FDs in critical schemata. Considering computational barriers to overall efficiency, our algorithms achieve their goals efficiently in practice.

## 4.9 How much overhead do we save?

We will study now how our optimizations on the logical level transcend to integrity maintenance at the operational level. For that purpose, we insert 10k, 20k, and 30k of records into abalone, hepatitis, lineitem, ncvoter, and weather. These insertions are done for the projections of these records onto the output schemata of our decompositions, resulting from *iConf-f*, *Conf*, *BC-Cover*, and *Synthesis*. Operations are repeated 10 times and the average runtime reported. We report the total times where all constraints are enforced by FDs and where FDs are separated into non-key FDs and minimal keys.

Firstly, integrity maintenance with only FDs is inefficient, if not infeasible. There are orders of time units difference (hours over minutes, or minutes over seconds) between the uniform use of FDs and the combined use of non-key FDs and minimal keys. In fact, non-key FDs require triggers while minimal keys are supported by UNIQUE indices. This further motivates our parameterized framework that inherently links representations of constraints at the schema level with integrity maintenance at the operational level.

Secondly, our solution really does address the bottleneck of update inefficiency. By minimizing non-key FDs we do not just reduce update overheads further, but our reduction comes at a larger scale than optimizations from previous work. This is quantified in Table 3 that compares the algorithms in their ability to reduce update overheads when FDs are separated into non-key FDs and minimal keys. We report the average reductions in total and per schema (in

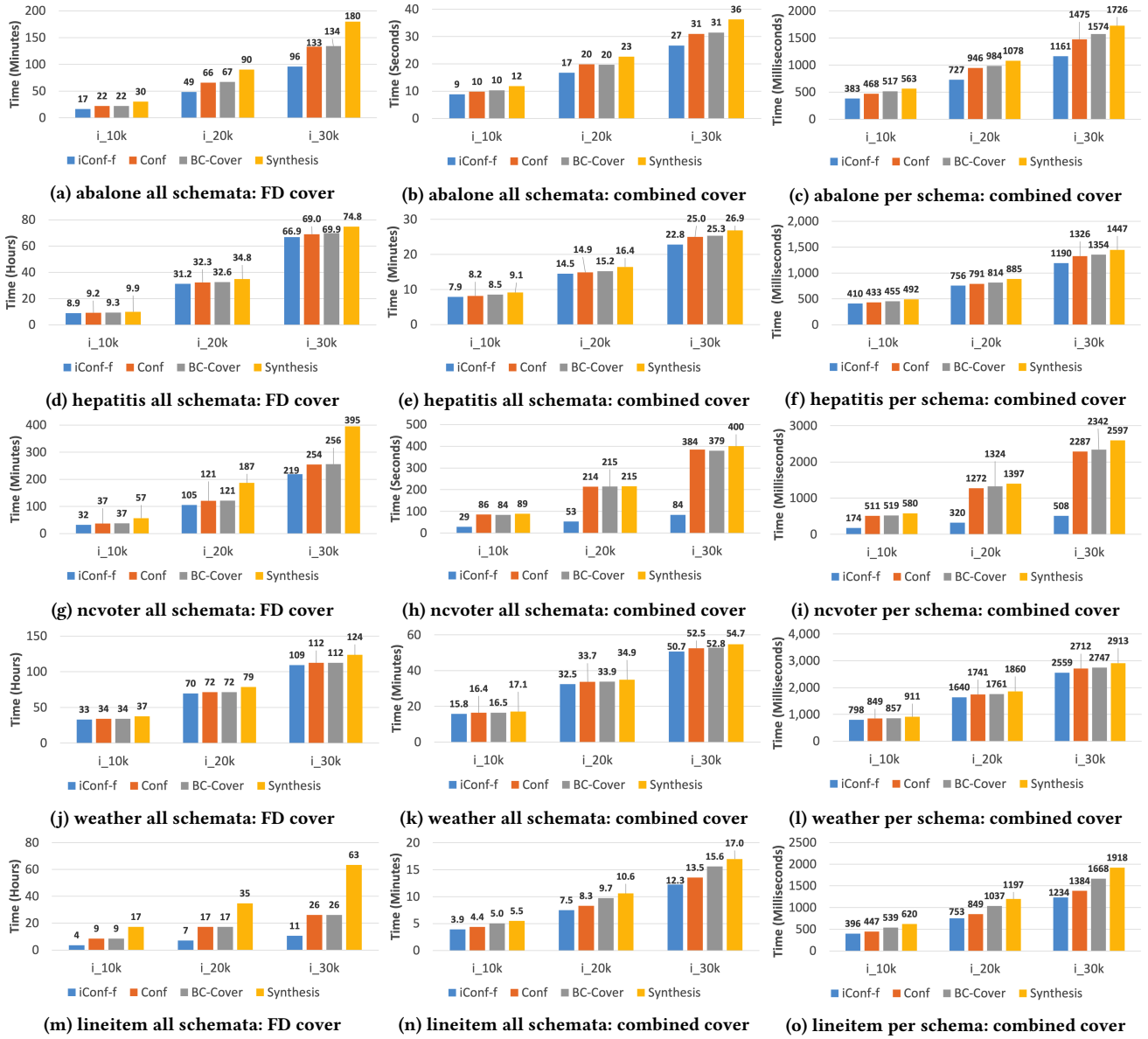



**Figure 5: Overheads for maintaining integrity with FD and combined covers when inserting 10k, 20k, 30k of records on schemata obtained by different normalization methods**

| Comparison | total | per schema |
|---|---|---|
| *iConf-f* over *Conf* | 20.0% | 23.5% |
| *Conf* over *BC-Cover* | 3.0% | 6.2% |
| *BC-Cover* over *Synthesis* | 5.7% | 8.7% |

**Table 3: Average reduction of overheads across algorithms**

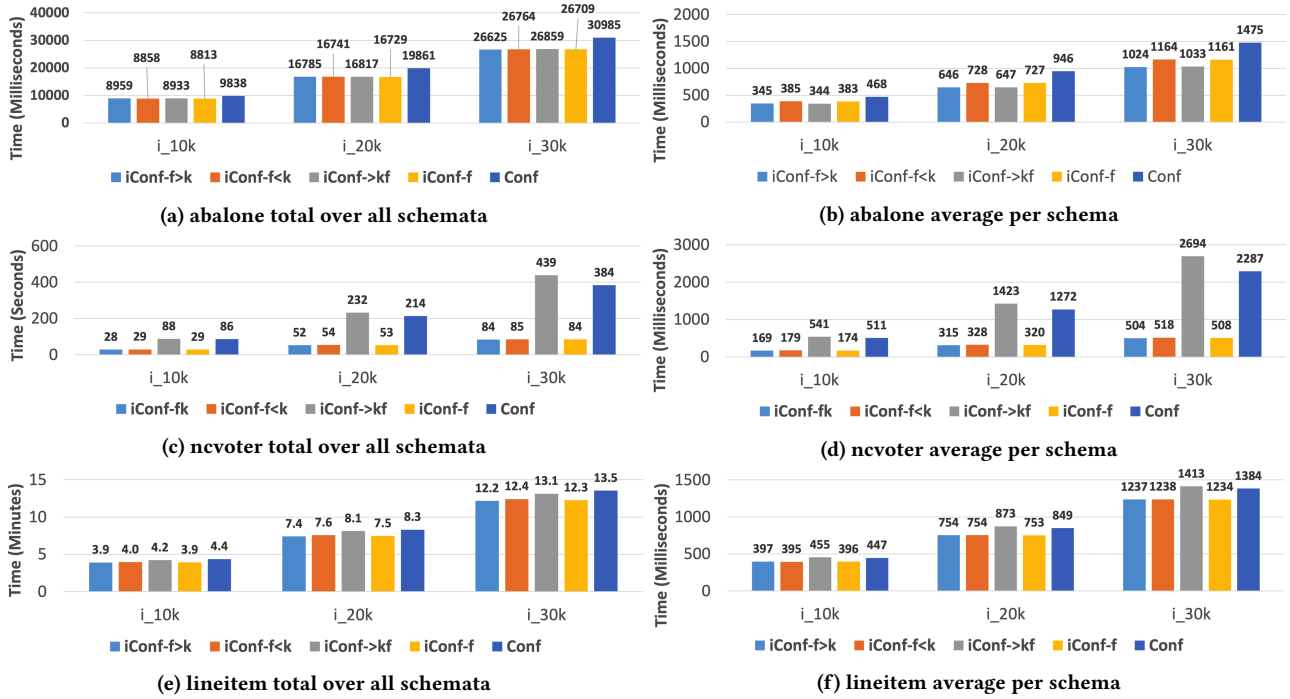
percent), across all update operations and data sets for each of the two algorithms we compare.

Figure 5 details update overheads on all 5 data sets, including total times using (1) minimal-reduced covers with FDs only, and (2) constraint sets combining all minimal keys with a minimal-reduced cover for all non-key FDs. Indeed, *iConf-f* outperforms the previously best algorithm *Conf*, across all scenarios with different input sizes for schemata, constraints and records. Hence, the optimizations do translate from logical to operational level. The magnitudes of reduction differ between scenarios but are significant.

*Optimizations.* We may use secondary parameters to break some ties that persist to hold after using primary parameters. Table 4 lists properties of decompositions resulting from these strategies on some data sets where they differ. We note small differences, and sometimes few schemata may be eliminated or added. Figure 6

| Data | Alg | Size | BC | 3NF | BCNF distribution | 3NF distribution |
|---|---|---|---|---|---|---|
| abalone | iConf-fk | 26 | 22 | 4 | [3:1,2:12,1:9] | [4:1,2:1,1:2] |
| | iConf-f<k | 23 | 18 | 5 | [2:11,1:7] | [4:1,2:1,1:3] |
| | iConf->kf | 26 | 22 | 4 | [3:1,2:12,1:9] | [4:1,2:1,1:2] |
| | iConf-f | 23 | 18 | 5 | [2:11,1:7] | [4:1,2:1,1:3] |
| ncvoter | iConf-fk | 166 | 145 | 21 | [2:27,1:118] | [3:1,2:2,1:18] |
| | iConf-f<k | 164 | 143 | 21 | [3:1,2:28,1:114] | [3:1,2:2,1:18] |
| | iConf->kf | 163 | 142 | 21 | [2:28 1:114] | [5:1,3:1,2:3,1:16] |
| | iConf-f | 166 | 145 | 21 | [3:1,2:27,1:117] | [3:1,2:2,1:18] |
| lineitem | iConf-fk | 590 | 560 | 30 | [15:1,10:1,6:3,5:3,4:5,3:23,2:105,1:419] | [4:1, 2:9, 1:20] |
| | iConf-f<k | 602 | 573 | 29 | [15:1,10:1,6:4 5:3,4:6,3:23,2:106,1:429] | [4:1, 2:9, 1:19] |
| | iConf->kf | 558 | 528 | 30 | [15:1,10:1,6:3,5:3,4:5,3:21,2:95,1:399] | [10:1,8:2,6:1,5:1,4:1,3:2,2:9,1:13] |
| | iConf-f | 596 | 567 | 29 | [15:1,10:1,6:4,5:3,4:5,3:23,2:105,1:425] | [4:1,2:9,1:19] |

**Table 4: Properties of Designs for Optimized Strategies**




**(a) abalone total over all schemata**

**(b) abalone average per schema**

**(c) ncvoter total over all schemata**

**(d) ncvoter average per schema**

**(e) lineitem total over all schemata**

**(f) lineitem average per schema**

**Figure 6: Optimizations of insertion overheads: total time for all schemata and average time per schema with combined covers**

illustrates the update performance on these decompositions. In line with the small differences at logical level, there are small differences at operational level. Overall, breaking further ties by maximizing the number of minimal keys, that is strategy (A1), appears to result in further small reductions of update overheads.

**Conclusion.** Experiments at operational level demonstrate that our framework does address the bottleneck of previous normalization efforts. By minimizing non-key FDs we achieve reductions at a scale larger than optimizations from previous work. Without separating non-key FDs from minimal keys, integrity maintenance degrades by orders of magnitude. Selecting redundant critical schemata with fewer FDs (and more keys if ties persist) results in the largest reduction of update overheads we were able to demonstrate.

## 5 CONCLUSION AND FUTURE WORK

We will summarize how our contributions address the research questions from the introduction. Firstly, we have shown how 3NF schemata can be separated by partitioning a set of FDs into its set of $k$ minimal keys and a minimal-reduced cover for the remaining non-key FDs with $f$ elements. Access to the parameters enables us to compare 3NF schemata based on what we consider better in terms of $k$ and $f$, thereby addressing (Q1). While 3NF says that all integrity constraints can be enforced by minimal keys and prime FDs (non-key FDs where the RHS is a prime attribute), we defined $(k, f)$-3NF expressing that all integrity constraints can be enforced by $k$ minimal keys and $f$ prime FDs. BCNF and $k$-CONF are covered

by the special case where $f = 0$. This answers (Q2). 3NF synthesis can be optimized with respect to any target strategy that we declare in terms of $k$ and $f$. We can choose from a diverse range of strategies by minimizing or maximizing parameters, declaring primary and secondary parameters, and merging different strategies for BCNF and critical schemata. Hence, we address (Q3) by provably optimizing 3NF synthesis for the target strategy we declare. Despite the likely intractability of the underlying computational problem in general, our algorithms perform efficiently in practice, especially considering that schema design happens rarely compared to frequent updates at operational level. Indeed, our parameterized framework is intrinsically linked to operational performance. In addressing (Q4) and the bottleneck of integrity maintenance, we can simply declare any target strategy that minimizes $f$ as primary parameter. Our experiments show that this strategy brings forward schema designs that improve update performance significantly more than designs resulting from previous optimizations that only involve minimal keys.

Future work will address optimum covers that use sizes of keys and FDs [25], rather than their numbers. Here, the trade-off between expensive computations of optimum covers and additional performance gains will be interesting to analyze. It will also be interesting to investigate schema optimization after the database has become operational and information about workload patterns of updates and queries are available. Such knowledge is not input for classical normalization, including BCNF and 3NF. Higher normal forms [9], such as 4NF [11], 5NF [32] and Inclusion Dependency Normal Form [21], will also be investigated. Interestingly, notions of minimal and optimal covers do not exist for the dependencies they are based on, such as multivalued, join and inclusion dependencies. It is also important to extend the work to other data models, including incomplete [19, 34], temporal [14], Web [3, 10, 35], uncertain [22] and graph data [2, 31].

## REFERENCES


[1] S. Sudarshan A. Silberschatz, H.F. Korth. 2019. *Database System Concepts* (7 ed.). McGraw-Hill.

[2] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Alastair Green, Jan Hidders, Bei Li, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Stefan Plantikow, Ognjen Savkovic, Michael Schmidt, Juan Sequeda, Slawek Staworko, Dominik Tomaszuk, Hannes Voigt, Domagoj Vrgoc, Mingxi Wu, and Dusan Zivkovic. 2023. PG-Schema: Schemas for Property Graphs. *Proc. ACM Manag. Data* 1, 2 (2023), 198:1–198:25.

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

[4] William Ward Armstrong. 1974. Dependency Structures of Data Base Relationships. In *Information Processing, Proceedings of the 6th IFIP Congress 1974, Stockholm, Sweden, August 5-10, 1974*. 580–583.

[5] Catriel Beeri and Philip A. Bernstein. 1979. Computational Problems Related to the Design of Normal Form Relational Schemas. *ACM Trans. Database Syst.* 4, 1 (1979), 30–59.

[6] Catriel Beeri, Philip A. Bernstein, and Nathan Goodman. 1978. A Sophisticate's Introduction to Database Normalization Theory. In *Fourth International Conference on Very Large Data Bases, September 13-15, 1978, West Berlin, Germany*. 113–124.

[7] Philip A. Bernstein. 1976. Synthesizing Third Normal Form Relations from Functional Dependencies. *ACM Trans. Database Syst.* 1, 4 (1976), 277–298.

[8] Joachim Biskup, Umeshwar Dayal, and Philip A. Bernstein. 1979. Synthesizing Independent Database Schemas. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, USA, May 30 - June 1*. 143–151.

[9] C. J. Date and Ronald Fagin. 1992. Simple Conditions for Guaranteeing Higher Normal Forms in Relational Databases. *ACM Trans. Database Syst.* 17, 3 (1992), 465–476.

[10] Michael DiScala and Daniel J. Abadi. 2016. Automatic Generation of Normalized Relational Schemas from Nested Key-Value Data. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*. 295–310.

[11] Ronald Fagin. 1977. Multivalued Dependencies and a New Normal Form for Relational Databases. *ACM Trans. Database Syst.* 2, 3 (1977), 262–278.

[12] Marie Fischer, Paul Roessler, Paul Sieben, Janina Adamcic, Christoph Kirchherr, Tobias Straeubig, Youri Kaminsky, and Felix Naumann. 2023. BCNF* - From Normalized- to Star-Schemas and Back Again. In *Companion of the 2023 International Conference on Management of Data, SIGMOD/PODS 2023, Seattle, WA, USA, June 18-23, 2023*. 103–106.

[13] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, Sanjeev Saluja, Hannu Toivonen, and Ram Sewak Sharm. 2003. Discovering all most specific sentences. *ACM Trans. Database Syst.* 28, 2 (2003), 140–174.

[14] Christian S. Jensen, Richard T. Snodgrass, and Michael D. Soo. 1996. Extending Existing Dependency Theory to Temporal Databases. *IEEE Trans. Knowl. Data Eng.* 8, 4 (1996), 563–582.

[15] Henning Köhler. 2006. Finding Faithful Boyce-Codd Normal Form Decompositions. In *Algorithmic Aspects in Information and Management, Second International Conference, AAIM 2006, Hong Kong, China, June 20-22, 2006, Proceedings*. 102–113.

[16] Christoph Köhnen, Stefan Klessinger, Jens Zumbrägel, and Stefanie Scherzinger. 2023. A Plaque Test for Redundancies in Relational Data. In *Joint Proceedings of Workshops at the 49th International Conference on Very Large Data Bases (VLDB 2023), Vancouver, Canada, August 28 - September 1, 2023*.

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

[18] Carol Helfgott LeDoux and Douglas Stott Parker Jr. 1982. Reflections on Boyce-Codd Normal Form. In *Eigth International Conference on Very Large Data Bases, September 8-10, 1982, Mexico City, Mexico, Proceedings*. 131–141.

[19] Mark Levene and George Loizou. 1999. Database Design for Incomplete Relations. *ACM Trans. Database Syst.* 24, 1 (1999), 80–125.

[20] Mark Levene and George Loizou. 1999. *A guided tour of relational databases and beyond*. Springer.

[21] Mark Levene and Millist W. Vincent. 2000. Justification for Inclusion Dependency Normal Form. *IEEE Trans. Knowl. Data Eng.* 12, 2 (2000), 281–291.

[22] Sebastian Link and Henri Prade. 2019. Relational database schema design for uncertain data. *Inf. Syst.* 84 (2019), 88–110.

[23] Sebastian Link and Ziheng Wei. 2021. Logical Schema Design that Quantifies Update Inefficiency and Join Efficiency. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*. 1169–1181.

[24] Claudio L. Lucchesi and Sylvia L. Osborn. 1978. Candidate Keys for Relations. *J. Comput. Syst. Sci.* 17, 2 (1978), 270–279.

[25] David Maier. 1980. Minimum Covers in Relational Database Model. *J. ACM* 27, 4 (1980), 664–674.

[26] David Maier. 1983. *The Theory of Relational Databases*. Computer Science Press.

[27] Heikki Mannila and Kari-Jouko Räihä. 1986. Design by Example: An Application of Armstrong Relations. *J. Comput. Syst. Sci.* 33, 2 (1986), 126–141.

[28] Sylvia L. Osborn. 1979. Testing for Existence of a Covering Boyce-Codd Normal Form. *Inf. Process. Lett.* 8, 1 (1979), 11–14.

[29] Thorsten Papenbrock and Felix Naumann. 2016. A Hybrid Approach to Functional Dependency Discovery. In *SIGMOD*. 821–833.

[30] Thorsten Papenbrock and Felix Naumann. 2017. Data-driven Schema Normalization. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017*. 342–353.

[31] Philipp Skavantzos and Sebastian Link. 2023. Normalizing Property Graphs. *Proc. VLDB Endow.* 16, 11 (2023), 3031–3043.

[32] Millist W. Vincent. 1997. A Corrected 5NF Definition for Relational Database Design. *Theor. Comput. Sci.* 185, 2 (1997), 379–391.

[33] Ziheng Wei and Sebastian Link. 2019. Discovery and Ranking of Functional Dependencies. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. 1526–1537.

[34] Ziheng Wei and Sebastian Link. 2021. Embedded Functional Dependencies and Data-completeness Tailored Database Design. *ACM Trans. Database Syst.* 46, 2 (2021), 7:1–7:46.

[35] Cong Yu and H. V. Jagadish. 2008. XML schema refinement through redundancy detection and normalization. *VLDB J.* 17, 2 (2008), 203–223.

[36] Carlo Zaniolo. 1982. A New Normal Form for the Design of Relational Database Schemata. *ACM Trans. Database Syst.* 7, 3 (1982), 489–499.

[37] Zhuoxing Zhang, Wu Chen, and Sebastian Link. 2023. Composite Object Normal Forms: Parameterizing Boyce-Codd Normal Form by the Number of Minimal Keys. *Proc. ACM Manag. Data* 1, 1 (2023), 13:1–13:25.

# A PROOFS

Lemma A.1 (Lemma 3.7 restated).
*Let $(R, \Sigma)$ denote a set $\Sigma$ of FDs over relation schema $R$. Let $\Omega$ denote a 3NF-substructure of $(R, \Sigma)$. Then $\Omega$ is 3NF update complete if and only if all of the following hold:*

*(1)* $\Sigma \subseteq \Omega^+$

*(2)* $(R, \Sigma)$ *is in Third Normal Form*

Proof. "⇐": We show first that the two conditions (1) and (2) are sufficient for 3NF update completeness to hold.

Let $r$ denote a relation over $R$ that satisfies $\Sigma$, and let $t \in dom(R)$ and $t' \in r$ be such that for all $X \to Y \in \Omega$, i) if $X \to R \in \Omega^+$, then $X \not\subseteq ag(t, t')$ and ii) if $Y - X \subseteq P_\Omega$ and $X \subseteq ag(t, t')$, then $Y \subseteq ag(t, t')$. We need to show that $r \cup \{t\}$ satisfies $\Sigma$.

Let $X \to Y \in \Sigma$. Assume that $X \subseteq ag(t, t')$, otherwise there is nothing else to show. Since (2) holds it follows that $X \to R \in \Sigma^+$ or $Y - X \subseteq P_\Sigma$. Since (1) holds we have $\Omega^+ \subseteq (\Sigma^+)^+ = \Sigma^+ \subseteq (\Omega^+)^+ = \Omega^+$ and therefore $\Omega^+ = \Sigma^+$. It follows that a) $X \to R \in \Omega^+$ or b) $Y - X \subseteq P_\Omega$. Since $X \subseteq ag(t, t')$, our assumption dictates that $Y \subseteq ag(t, t')$, which is what we needed to show. Consequently, $r \cup \{t\}$ satisfies $\Sigma$ and we have shown that 3NF udpate completeness holds.

"⇒": We show now that 3NF update completeness entails (1) and (2). We show first that not (2) implies that 3NF update completeness does not hold.

Assume that $(R, \Sigma)$ is not in Third Normal Form. Then there is some $X \to Y \in \Sigma^+$ such that $X \to R \notin \Sigma^+$ and $Y - X \not\subseteq P_\Sigma$. In particular, $X \to Y \notin \Omega$. Let $r := \{t'\}$ be a single tuple relation and $t \in dom(R)$ such that for all $A \in R$, $t'(A) = t(A)$ if and only if $A \in X^+_\Omega$. Let $Z \to Y \in \Omega$. Assume $Z \to R \in \Omega^+$. If $Z \subseteq X^+_\Omega$, then $X \to Z \in \Omega^+$, and thus $X \to R \in \Omega^+$. Hence, $X \to R \in \Sigma^+$, a contradiction. Consequently, $Z \not\subseteq X^+_\Omega$, which means $Z \not\subseteq ag(t, t')$. Assume now that $Y - Z \subseteq P_\Omega$ and $Z \subseteq ag(t, t')$. We show this case cannot occur. Indeed, this case would mean that $Z \subseteq X^+_\Omega$ by construction. This meant $X \to Z \in \Omega^+$ and therefore $X \to Y \in \Omega^+$, a contradiction.

We show secondly that not (1) implies that 3NF update completeness does not hold. Assume that there is some $X \to Y \in \Sigma - \Omega^+$. Then it follows that $X \to R \notin \Sigma^+$ and $Y - X \not\subseteq P_\Sigma$. Just like before we can show by the same relation $r$ and tuple $t$ that 3NF update completeness does not hold. □

Corollary A.2 (Corollary 3.8 restated). *If $(R, \Sigma)$ is in 3NF, then the 3NF-core $\mathcal{K}_\Sigma \cup \mathcal{F}_\Sigma$ is an intransitive composite object for $(R, \Sigma)$.*

Proof. The 3NF-core is always a 3NF-substructure. If $(R, \Sigma)$ is in 3NF, then $\Sigma \subseteq (\mathcal{K}_\Sigma \cup \mathcal{F}_\Sigma)^+$. Consequently, (1) and (2) of Lemma 3.7 are satisfied by $\Omega := \mathcal{K}_\Sigma \cup \mathcal{F}_\Sigma$. Hence, $\Omega$ is an intransitive composite object for $(R, \Sigma)$ by Lemma 3.7. □

Theorem A.3 (Theorem 3.9 restated). *For all relation schemata $R$, and all sets $\Sigma$ of FDs over $R$ the following holds: $(R, \Sigma)$ is in 3NF if and only if $(R, \Sigma)$ is in iCONF.*

Proof. "⇒": Assume that $(R, \Sigma)$ is in 3NF. We show that $(R, \Sigma)$ is in iCONF.

Indeed, Proposition 3.8 shows that $\Omega := \mathcal{K}_\Sigma \cup \mathcal{F}_\Sigma$ is an intransitive composite object. By definition, $(R, \Sigma)$ is in iCONF.

"⇐": We show the contraposition. Hence, we assume that $(R, \Sigma)$ is not in 3NF, and need to show that $(R, \Sigma)$ is not in iCONF.

The 3NF-core $\Omega := \mathcal{K}_\Sigma \cup \mathcal{F}_\Sigma$ is a 3NF-substructure, but since $(R, \Sigma)$ is not in 3NF, Lemma 3.7 shows that $\Omega$ is not 3NF update complete. Consequently, $\Omega$ is not an intransitive composite object, so $(R, \Sigma)$ is not in iCONF. □