The VLDB Journal (2025) 34:29
https://doi.org/10.1007/s00778-025-00910-2

REGULAR PAPER

Mixed covers: optimizing updates and queries using minimal keys and functional dependencies

Zhuoxing Zhang[1] · Sebastian Link[1]

Received: 17 July 2024 / Revised: 17 January 2025 / Accepted: 25 February 2025 / Published online: 17 March 2025

Abstract
Covers for a set of functional dependencies (FDs) are fundamental for many areas of data management, such as integrity maintenance, query optimization, database design, and data cleaning. When declaring integrity constraints, keys enjoy native support in database systems while FDs need to be enforced by triggers or at application level. Consequently, maximizing the use of keys will provide the best support. We propose the new notion of mixed cover for a set of FDs, comprising the set of minimal keys together with a cover for the set of non-key FDs implied by the FD set. We establish sequential and parallel algorithms for computing mixed covers from a given set of FDs, and illustrate that they complement each other in terms of their performance. Even though FD covers are typically smaller in number or size than their corresponding mixed cover, the latter generate orders of magnitude lower overheads during integrity maintenance. We also quantify how mixed covers improve the performance of query, refresh and insert operations on the TPC-H benchmark under different constraint workloads and scaling factors. Finally, we illustrate the growth of performance improvement for optimal over minimal-reduced covers, justifying the significant time required for computing the former.

Keywords Cover · Functional dependency · Data integrity · Key · Query · Update

1 Introduction

Functional dependencies (FDs) can express many requirements of applications. RDBMSs provide means for maintaining the integrity of data by enforcing such FDs. When an FD $X \rightarrow Y$ is declared on a relation schema R, all records with values matching on all the attributes in X must have values matching on all the attributes in Y. Hence, updates on a given relation prompt the RDBMS to validate a given set F of FDs. Intuitively, both the number and size of FDs in F have a direct impact on the overhead required for integrity maintenance. Here, $|F|$ will denote the number of FDs in F, and $||F||$ the total number of attribute occurrences in F. David Maier investigated two notions of covers for a set of FDs [29]. A *cover* for F is a set G of FDs that implies the same FDs as F. G is *minimal* if there is no other cover for F with fewer FDs than G, and G is *optimal* if there is no other cover for F with fewer total attribute occurrences than G. While minimal covers can be found in polynomial time, the problem of deciding whether there is an optimal cover of a given size is *NP*-complete [29].

Covers of FDs naturally facilitate fundamental tasks in data management. Intuitively, the covers constitute smaller representations of a given set of FDs. Hence, the same benefits are realized with less effort. Firstly, smaller FD covers cause less overheads for integrity maintenance. In particular, FDs or attributes that are redundant in an FD set cause overheads during integrity maintenance that are redundant, too. This is true for transactional workloads that use normalized, narrow tables, but also for analytical tasks that use non-normalized, wider tables. In fact, modern data architectures such as active data warehouses and cloud databases are expected to process updates frequently to enable real-time decision making based on current data. Secondly, smaller FD covers offer opportunities for further query optimization [10, 11, 22, 27]. For example, optimizers may detect redundant DISTINCT clauses or GROUP BY attributes by validating whether a key or FD is implied by the underlying FD set [22], and checking implication is faster when smaller FD covers

✉ Sebastian Link
s.link@auckland.ac.nz

Zhuoxing Zhang
zzha969@aucklanduni.ac.nz

[1] School of Computer Science, The University of Auckland, Auckland, New Zealand

are used. Thirdly, smaller covers result in outputs of normalization algorithms with fewer relation schemata or fewer attributes during logical schema design [7, 26, 32, 49], which facilitates more efficient data management in the lifetime of databases [30]. As a final example, smaller covers for the set of FDs that are mined from database instances are easier to comprehend for humans who need to identify FDs that are meaningful rather than those that hold only accidentally [1, 46]. Here, an FD is meaninful when it expresses a general rule of the underlying domain that every relation ought to satisfy, while accidental refers to the fact that an FD may hold (accidentally) on a given relation but does not constitute a general rule of the application domain. Example of meaningful FDs are listed in Example 1, but an FD like DATE → OFFENSE would be accidental as different offenses may occur on any given day. Intriguingly, FDs or attributes that are mined redundantly slow down the progress of FD discovery unnecessarily [16, 21, 33–35, 39, 44, 46]. Even if FDs only hold accidentally on the relation they were mined from, they may still benefit query optimization [22]. These areas of impact greatly motivate the study of covers.

A natural question arising is how we would use any set of FDs for maintaining the integrity of data? Firstly, the most important special case of FDs are keys. Given a relation schema R, an attribute subset X is called a *key* whenever the FD $X \rightarrow R$ holds. Indeed, no relation satisfying $X \rightarrow R$ can ever have different records with matching values on all the attributes in X. While keys enjoy native support in RDBMSs, FDs need to be enforced by triggers or at application level. Hence, for implementation purposes only, it makes sense to rely on keys as much as possible. Secondly, every key declared on a relation schema (primary key or UNIQUE constraint) will result in a UNIQUE index, meaning integrity enforcement of keys becomes efficient. Hence, for performance purposes, it makes sense to rely on keys as much as possible. Thirdly, database normalization aims at the transformation of any dependencies into keys [12]. For FDs only, state-of-the-art computes dependency-preserving decompositions in Third Normal Form (3NF), which are in Boyce-Codd Normal Form (BCNF) whenever possible [7, 32]. Given only dependency-preserving decompositions into 3NF can be guaranteed, not all FDs can be expressed by keys. It is therefore more natural and practical to investigate mixed notions of covers for a set F of FDs, comprising the set of minimal keys implied by F and a cover for the set of non-key FDs in F (FDs in F not implied by the set of minimal keys). Mixed covers may even be of smaller size than their FD cover, as illustrated next.

Example 1 Schema TRAFFIC consist of the attributes CAR-SERIAL#, LICENSE#, OWNER, DATE, TIME, TICKET#, and OFFENSE. The following is a minimal cover of size 15 for the underlying FDs:

- CAR-SERIAL# → LICENSE#
- LICENSE# → CAR-SERIAL#, OWNER
- TICKET# → CAR-SERIAL#, OWNER, DATE, TIME
- CAR-SERIAL#, DATE, TIME → TICKET#, OFFENSE.

The following is an optimal cover of size 14:

- CAR-SERIAL# → LICENSE#
- LICENSE# → CAR-SERIAL#, OWNER
- TICKET# → CAR-SERIAL#, DATE, TIME
- CAR-SERIAL#, DATE, TIME → TICKET#, OFFENSE.

However, the following optimal mixed cover has size 12:

- three keys {TICKET#}, {CAR-SERIAL#, DATE, TIME}, {LICENSE#, DATE, TIME}
- CAR-SERIAL# → LICENSE#
- LICENSE# → CAR-SERIAL#, OWNER.

□

Even though they are typically larger than their FD covers, mixed covers are better for integrity maintenance since keys use UNIQUE indexes, as illustrated next by a famous 3NF example [4].

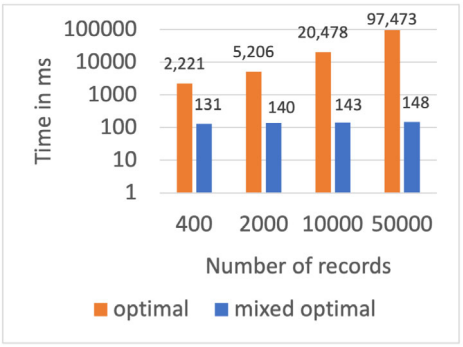
Example 2 Consider the 3NF schema MAIL with attributes *A(DDRESS)*, *C(ITY)*, and *Z(IP)*, and FDs: $AC \rightarrow Z$ and $Z \rightarrow C$, which is its own optimal FD cover of size 5. The FD set implies two minimal keys $\{A, C\}$ and $\{A, Z\}$, plus the FD $Z \rightarrow C$, so the size of an optimal mixed cover is 6. Figure 1a shows a synthetic relation over MAIL, which satisfies the optimal covers, and violates all FDs not implied by the covers. Hence, the relation is a perfect sample for the FD set given. To compare how FD and mixed covers handle integrity maintenance, we have created relations of increasing sizes by taking disjoint unions of copies of the sample. Figure 1b, c compare the times (in ms) for performing inserts for 10% (respectively, 40%) of records from the given relations, based on optimal FD and mixed covers (note the logarithmic scales). Not only does the mixed cover perform 99% faster than the FD cover, but it also shows robust scalability. This superior performance is natural due to UNIQUE indices. On perfect samples, such as the Armstrong table [5, 18, 24] from Table 1, the improvement is also evident for Example 1, as illustrated in Fig. 2a, b. □

Our contributions are summarized as follows:

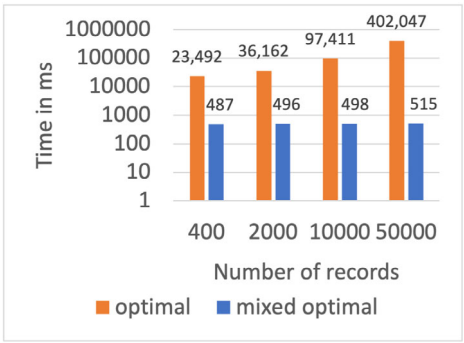
(1) For classical types of FD covers, such as non-redundant, reduced, canonical, minimal, minimal reduced, and optimal cover [30], we analyze their computation on real and synthetic data, their reduction in numbers and size of

ADDRESS	CITY	ZIP
38 Princes	AKL	1010
12 Borneo	AKL	0632
38 Princes	Levin	5510
3 Symmonds	AKL	1010

(a) Perfect Sample



(b) Insert 10% records

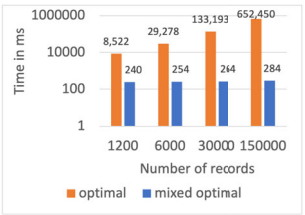


(c) Insert 40% records

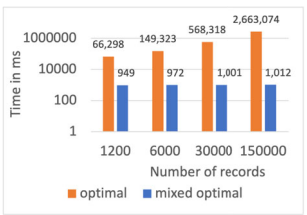
Fig. 1 Update times using optimal FD covers and mixed optimal covers on MAIL

Table 1 Armstrong table for traffic example

CAR-SER#	LIC#	Owner	Date	Time	Ticket#	Offense
L8ER	0A0	Jack	26/2	9 pm	0-ABC	Speed
FAST	1B1	Jack	26/2	9 pm	1-BCD	Speed
L8ER	0A0	Jack	13/7	9 pm	2-CDE	Speed
L8ER	0A0	Jack	26/2	10 am	3-DEF	Speed
2MUCH	2C2	Jack	26/2	9 pm	4-EFG	Drunk
L8ER	0A0	Jack	20/6	9 pm	5-FGH	No-Belt
L8ER	0A0	Jack	26/2	4 pm	6-GHI	Mobile
FIR3	3D3	Taz	26/2	9 pm	7-HIJ	Speed
L8ER	0A0	Jack	21/11	9 pm	8-IJK	Speed
L8ER	0A0	Jack	26/2	7 pm	9-JKL	Speed
4U4ME	4E4	Jack	26/2	9 pm	10-KLM	Speed
XP1R3D	5F5	Bibi	19/12	3 am	11-LMN	Warrant



(a) Insert 10% new records



(b) Insert 40% new records

Fig. 2 Update times using optimal FD covers vs mixed optimal covers on TRAFFIC

representing FDs, and how much they reduce overheads during integrity management.

(2) For each notion of FD covers, we introduce a mixed variant, comprising the set of minimal keys implied and a corresponding cover for the set of FDs not implied by the minimal keys.

(3) Previous work has not properly justified that different notions of FD covers are actually different. In particular, for all schemata in BCNF, all notions of FD covers and mixed variants collapse into the set of minimal keys. However, we will show that the relationships between FD covers known from previous work [30] already apply to schemata on 3NF. Since this is the only case in practice where frequent integrity maintenance should happen, our result does justify the different notions of FD covers. We also show that the same relationships transcend to mixed variants.

(4) We propose sequential and parallel algorithms for computing mixed covers of a target type for an FD set. Experiments illustrate that they complement one another, as the sequential algorithm performs well when the parallel algorithm does not and vice versa.

(5) We extend our experiments from FD covers to mixed variants. Firstly, we analyze the performance of our sequential and parallel algorithms in computing mixed covers, illustrating when one outperforms the other and therefore benefiting from both algorithms. We demonstrate on a variety of schemata and data how mixed covers provide high performance support for integrity maintenance, for both analytical tasks over non-normalized schemata and transactional tasks over normalized schemata. Finally, we show that using mixed over FD covers improves query

performance significantly and is essential for refresh and insert operations of the TPC-H benchmark across different constraint workloads and scaling factors (factors by which the standard size of an input is multiplied).

(6) Minimal-reduced covers can be arbitrarily poor approximations of optimal covers [31]. We strengthen this result to hold for 3NF schemata already, and further show that it also holds for mixed variants of both covers. This provides justification for the potentially large costs of computing (mixed) optimal covers.

In summary, mixed covers transform the classical notion of FD covers from database theory into best practice for minimizing overheads during update maintenance in both non-normalized and normalized settings. We highlight the impact on application areas of FD covers from before. Firstly, utilizing the set K of minimal keys saves orders of magnitude during integrity maintenance, as we will show. This clarifies that even larger-sized covers (namely mixed covers) can perform much better than FD covers of smaller size. Secondly, knowing K enables us to specify keys and utilize their UNIQUE indices for data management, including query optimization. This applies even when minimal keys or FDs are used that only hold accidentally [11, 22]. Thirdly, the benefit of utilizing minimal keys during logical database design has recently been illustrated by parameterizing BCNF by the number of minimal keys [49]. For data profiling [1], our ideas suggest separating FD mining into the discovery of minimal keys and non-key FDs, respectively. This calls for a dedicated line of future work, while our focus here is the impact on integrity maintenance and queries.

Organization. We repeat fundamental concepts in Sect. 2, and summarize classical types of covers in Sect. 3. We introduce the notion of mixed covers in Sect. 4, where we also establish that relationships, known to hold between different types of FD covers, also hold between the corresponding types of their mixed variants, and that these relationships already hold on 3NF schemata. Our sequential and parallel algorithms for computing mixed covers of a target type t are proposed and analyzed in Sect. 5. Section 6 is dedicated to experiments, before concluding and commenting on future work in Sect. 7. All artifacts associated with our experiments can be found in an open repository.[1]

Extension. The current article has extended the conference paper [50] in multiple directions. Firstly, we have included a thorough summary of FD covers and their computation to make the paper self-contained. Secondly, we have extended examples and included all proofs in the article. Thirdly, we have broken down our experimental analysis to individual datasets, in contrast to only summarizing our comparison between FD covers and their mixed variants across all datasets. Fourthly, we have entirely new sections on the relationship between minimal-reduced and optimal covers, and their mixed variants. We show that minimal-reduced covers are arbitrarily poor approximations of optimal covers, for both FD covers and their mixed variants on database schema in 3NF. We have also included a new research question for the experimental evaluation. Our analysis shows that (mixed) optimal covers can significantly improve update efficiency over (mixed) minimal-reduced covers. This justifies the effort of computing (mixed) optimal covers. Finally, we also illustrate experimentally on the TPC-H benchmark how the gain in running updates with mixed instead of FD covers grows with the underlying volume of the data.

[1] https://github.com/ZhuoxingZhang/mixed-covers-journal.

2 Functional dependencies

We will provide the necessary background on functional dependencies, including their definition within the relational model of data, axiomatic and algorithmic solutions to their implication problem.

2.1 Relation schemata and relations

We may use attribute symbols, such as A, B, C, etc., that stand for column names of a table. For each attribute A there is some set $dom(A)$, the set of values that may occur in column A of any table. A *relation schema* is a finite set R of attributes. Intuitively, R comprises all properties that every entity over R needs to be described by. A *tuple* or *record* over R is a function $t : R \to \bigcup_{A \in R} dom(A)$ that maps every attribute of R to a value $t(A) \in dom(A)$ from its domain. A *relation* over R is a finite set of tuples over R. Example 2 features relation schema MAIL and Fig. 1a shows a relation over MAIL with synthetic values from non-negative integer domains.

SQL uniformly uses the null marker symbol `null` to handle all kinds of interpretations for incomplete values. Out of convenience we include `null` as a distinguished symbol of every domain, so we assume `null` $\in dom(A)$. Every occurrence of `null` means no information is currently available for the actual value [3, 19, 48].

2.2 Syntax and semantics

A *functional dependency* (FD) over relation schema R is an expression $X \to Y$ with attribute subsets $X, Y \subseteq R$. A relation r over R is said to *satisfy* the FD $X \to Y$ over R whenever every pair of tuples $t, t' \in r$ with values matching on all the attributes in X has also values matching on all the attributes in Y, that is, if $t(X) = t'(X)$ implies $t(Y) = t'(Y)$. FDs $X \to Y$ that are satisfied by all relations are called *trivial*, which is the case if and only if $Y \subseteq X$. We interpret every occurrence of `null` by either `null <> null` or

Algorithm 1 CLOSURE(X, F)

Require: A set of attributes X, and a set F of FDs
Ensure: The closure of X under F
1: *Closure* := X
2: *FDList* := List of FDs in F
3: **repeat**
4: *OldClosure* := *Closure*
5: Remove *Closure* from the left-hand side of FDs in *FDList*
6: **for** each FD $X \rightarrow Y$ in F **do**
7: *Closure* := *Closure* $\cup$ Y
8: *FDList* := *FDList* $-$ $\{\emptyset \rightarrow Y\}$
9: **until** *Closure*=*OldClosure* or *FDList*=[ ]
10: **return** *Closure*

Algorithm 2 MEMBER(F, $X \rightarrow Y$)

Require: A set $F \cup \{X \rightarrow Y\}$ of FDs
Ensure: *true*, if $\Sigma \models X \rightarrow Y$ *false*, otherwise
1: **if** $Y \subseteq$ CLOSURE(X, F) **then**
2: **return** *true*
3: **else**
4: **return** *false*

`null` = `null`. While different interpretation can produce different FD sets, the difference in their performance is minor. Hence, we only report results for the former interpretation, but do illustrate the similarity across both interpretations on one example. Comprehensive treatments of nulls like [6, 45] are not the subject of this study but interesting future work.

As example, the relation in Table 1 satisfies the FD {CAR-SERIAL#, DATE, TIME} $\rightarrow$ {TICKET#, OFFENSE}. It does not satisfy the FD {CAR-SERIAL#, DATE} $\rightarrow$ {TICKET#} nor the FD {CAR-SERIAL#, DATE} $\rightarrow$ {OFFENSE}. As another example, while the relation in Fig. 1a satisfies the FD $AC \rightarrow Z$, it does not satisfy the FD $CZ \rightarrow A$.

A *key* over relation schema R is an expression $X \subseteq R$. A relation r over R is said to *satisfy* the key X over R whenever there are no two different tuples in r with values matching on all the attributes in X, that is, if $t(X) = t'(X)$ holds for any $t, t' \in r$, then $t = t'$. It is easy to see that a relation over R satisfies the key X over R if and only if the relation satisfies the FD $X \rightarrow R$ over R. The relation in Fig. 1a satisfies the keys $\{A, C\}$ and $\{A, Z\}$, but not the key $\{C, Z\}$. For another example, the relation in Table 1 satisfies the key {CAR-SERIAL#, DATE, TIME}, but not the key {CAR-SERIAL#, DATE}.

FDs form an important class of integrity constraints, restricting relations to those considered meaningful for the underlying application. Whenever the set F contains the integrity constraints for an application domain, every relation ought to satisfy all constraints in F, particularly after the relation is updated. That is, we need to maintain the integrity of data under updates.

2.3 Implication problem

Constraints interact with one another. For a set $F \cup \{X \rightarrow Y\}$ of FDs we say that F *implies* $X \rightarrow Y$, denoted by $F \models X \rightarrow Y$, whenever every relation that satisfies every FD in F also satisfies $X \rightarrow Y$. This notion is fundamental for integrity management: If a relation satisfies all FDs in F, then we need not check whether the relation satisfies any other FD implied by F. However, if F does not imply $X \rightarrow Y$, then there is some relation that satisfies all FDs in F but does not satisfy $X \rightarrow Y$. Hence, if $X \rightarrow Y$ is a meaningful FD, we still need to check whether it is satisfied, even if we already know all FDs in F are satisfied.

Due to this importance, the *implication problem* has received considerable attention from the database community [13, 17, 43]. For a given class of constraints, the problem is to decide whether for a given relation schema R, and a given set $\Sigma \cup \{\varphi\}$ of constraints over R from that class, $\Sigma \models \varphi$. For the class of functional dependencies, the implication problem is *PTIME*-complete (that is complete for the class of problems decidable in deterministic polynomial time), axiomatisable by Armstrong's axioms and decidable in linear time [25].

For a given FD set F, we denote by $F^+ = \{X \rightarrow Y \mid F \models X \rightarrow Y\}$ the set of FDs implied by F. Similarly, for an attribute subset $X \subseteq R$ and an FD set F over R, we denote by $X_F^+ = \{A \in R \mid F \models X \rightarrow A\}$ the *attribute set closure* of X given F, that is, the set of attributes functionally determined by X given F. The implication problem for FDs can be solved efficiently by using the result that $X \rightarrow Y \in F^+$ if and only if $Y \subseteq X_F^+$. In words, F implies $X \rightarrow Y$ iff every attribute in Y is contained in the attribute set closure X_F^+ of X given F.

Algorithm 1 shows how the attribute set closure of X given F can be computed, that is, CLOSURE(X, F) $= X_F^+$, and Algorithm 2 returns MEMBER(F, $X \rightarrow Y$) $=$ *true* iff F implies $X \rightarrow Y$. Since Algorithm 1 can be implemented to run in linear time, Algorithm 2 decides the implication problem for FDs in the same time, too.

For example, given the set F of FDs from Example 1, we can deduce that {CAR-SERIAL#, DATE} is not a key implied by F. Indeed, CLOSURE(F,{CAR-SERIAL#, DATE}) consists of

$$\text{CAR} - \text{SERIAL\#, DATE, LICENSE\#, OWNER,}$$

which does not include TICKET#, TIME, OFFENSE. Hence, TRAFFIC is not a subset of CLOSURE(F,{CAR-SERIAL#, DATE}), and thus

$$\text{MEMBER(F, \{CAR} - \text{SERIAL\#, DATE\})} = \textit{false}.$$

Given F, a key X implied by F is *minimal* iff every proper subset $Y \subseteq X$ is not a key implied by F. Indeed, the key

Algorithm 3 NON- REDUNDANT(F)

Require: Set F of FDs
Ensure: A non-redundant cover G for F
1: $G := F$
2: **for** each FD $X \to Y$ in F **do**
3: **if** MEMBER($G - \{X \to Y\}, X \to Y$) **then**
4: $G := G - \{X \to Y\}$
5: **return** G

Algorithm 4 LEFT- REDUCED(F)

Require: Set F of FDs
Ensure: A left-reduced cover G for F
1: $G := F$
2: **for** each FD $X \to Y$ in F **do**
3: **for** each attribute A in X **do**
4: **if** MEMBER($G, (X - A) \to Y$) **then**
5: $G := (G - \{X \to Y\}) \cup \{(X - \{A\}) \to Y\}$
6: **return** G

Algorithm 5 RIGHT- REDUCED(F)

Require: Set F of FDs
Ensure: A right-reduced cover G for F
1: $G := F$
2: **for** each FD $X \to Y$ in F **do**
3: **for** each attribute A in Y **do**
4: **if** MEMBER($G - \{X \to Y\} \cup \{X \to (Y - A)\}, X \to A$) **then**
5: $G := (G - \{X \to Y\}) \cup \{X \to (Y - A)\}$
6: **return** G

{CAR-SERIAL#, DATE, TIME} is minimal given the FD set F from Example 1.

3 Notions of FD covers

The goal of this section is to summarize useful notions of covers and how to compute them [30].

Extensive research has been made to represent sets of FDs succinctly. For example, any FD implied by the set F_0 consisting of $Emp \to Dep$, $Dep \to Mgr$, $Emp \to Mgr$, $\{Emp, Dep\} \to Mgr$, $Emp \to \{Dep, Mgr\}$ is also implied by the set $G_0 = \{Emp \to Dep, Dep \to Mgr\}$.

Representation of FD sets are known as covers. Formally, two FD sets F and G over relation schema R are *equivalent*, written $F \equiv G$, if $F^+ = G^+$, that is, when they both imply the same set of FDs. If $F \equiv G$, then we say that F is a *cover* of G.

For FD sets F and G, $F \equiv G$ if and only if (1) for every $X \to Y \in G$, MEMBER($F, X \to Y$) $= true$ and (2) for every $X \to Y \in F$, MEMBER($G, X \to Y$) $= true$. For example, the FD sets F_0 and G_0 above are covers of one another. Indeed, $F_0 \equiv G_0$ since $G_0 \subseteq F_0$ and every FD in F_0 is implied by G_0.

3.1 Non-redundant covers

Non-redundant covers do not contain FDs implied by others. An FD set F is *non-redundant* if there is no proper subset F' of F where $F' \equiv F$. If such an F' exists, F is *redundant*. G is a non-redundant cover for FD set F, if $F \equiv G$ and G is non-redundant.

An FD $X \to Y \in G$ is called *redundant in* G if $G - \{X \to Y\} \models X \to Y$. Algorithm 3 eliminates FDs redundant in its input set.

Example 3 For $F = \{HoD \to Mgr, Mgr \to HoD, Mgr \to Dep, HoD \to Dep\}$, the result of NON- REDUNDANT(F) is $\{HoD \to Mgr, Mgr \to HoD, HoD \to Dep\}$. If F is represented in the order $[HoD \to Mgr, HoD \to Dep, Mgr \to HoD, Mgr \to Dep]$, the result of NON- REDUNDANT(F) is $\{HoD \to Mgr, Mgr \to HoD, Mgr \to Dep\}$.

Example 3 shows that non-redundant covers are not unique. Algorithm 3 finds non-redundant covers in time $\mathcal{O}(||F||^2)$, but only those contained in F. Hence, it cannot find the non-redundant cover $\{HoD \to Mgr, Mgr \to HoD, \{HoD, Mgr\} \to Dep\}$ of G from Example 3.

3.2 Reduced covers

Since non-redundant covers carry no redundant FDs, removal of further FDs does not result in covers. However, one may still reduce the size of a non-redundant cover by removing attributes from FDs.

Intuitively, we can remove *extraneous* attributes from the left or right side of an FD in F whenever this does not affect the closure of F. More formally, let F denote an FD set over relation schema R, and $X \to Y$ an FD in F. An attribute A in R is *extraneous in* $X \to Y$ with respect to F if (1) $X = AZ$, $X \neq Z$, and $(F - \{X \to Y\}) \cup \{Z \to Y\} \equiv F$, or (2) $Y = AW$, $Y \neq W$, and $(F - \{X \to Y\}) \cup \{X \to W\} \equiv F$.

For G_1 with FDs $Emp \to \{Dep, Mgr\}$, $Dep \to Mgr$, $\{Emp, Dep\} \to HoD$, attribute Mgr is extraneous in the right side of $Emp \to \{Dep, Mgr\}$, and attribute Dep is extraneous in the left side of $\{Emp, Dep\} \to HoD$.

The FD $X \to Y \in F$ is *left-reduced* if X contains no attribute A extraneous in $X \to Y$, $X \to Y$ is *right-reduced* if Y contains no attribute A extraneous in $X \to Y$, and $X \to Y$ is *reduced* if it is left-reduced, right-reduced and $Y \neq \emptyset$. Similarly, an FD set F is left-reduced (right-reduced, reduced) if every FD in F is left-reduced (respectively, right-reduced, reduced).

G_1 is neither left- nor right-reduced, while G_2 with FDs $Emp \to \{Dep, Mgr\}$, $Dep \to Mgr$, $Emp \to HoD$ is left- but not right-reduced, and the set G_3 with FDs $Emp \to Dep$, $Dep \to Mgr$, $Emp \to HoD$ is reduced.

Algorithm 6 computes a reduced cover for input FD set F. Step (1) computes a left-reduced cover, step (2) returns

Algorithm 6 REDUCED(F)

Require: Set F of FDs
Ensure: A reduced cover G for F
1: $G :=$ RIGHT- REDUCED(LEFT- REDUCED(F))
2: **for** each FD $X \to \emptyset$ in G **do**
3: $G := G - \{X \to \emptyset\}$
4: **return** G

Algorithm 7 CANONICAL(F)

Require: Set F of FDs
Ensure: A canonical cover G for F
1: $G :=$ REDUCED(F)
2: **for** each FD $X \to A_1 \cdots A_m$ in F with $m \geq 2$ **do**
3: $G := (G - \{X \to A_1 \cdots A_m\}) \cup \bigcup_{i=1}^{m} \{X \to A_i\}$
4: **return** G

Algorithm 8 MINIMAL(F)

Require: Set F of FDs
Ensure: A minimal cover G for F
1: $G :=$ NON- REDUNDANT(F)
2: **for** each $E_G(X)$ in $\overline{E}_G$ **do**
3: **for** each $Y \to U$ in $E_G(X)$ **do**
4: **if** there is some $Z \to V$ in $E_G(X)$ such that MEMBER($G - E_G(X), Y \to Z$) **then**
5: $G := (G - \{Y \to U, Z \to V\}) \cup \{Z \to UV\}$
6: **return** G

Algorithm 9 MINIMAL- REDUCED(F)

Require: Set F of FDs
Ensure: A reduced minimal cover G for F
1: **return** $G :=$ REDUCED(MINIMAL(F))

a right-reduced cover for the output of step (1), and step (3) eliminates any FDs $X \to \emptyset$ from the output of step (2). The algorithms for computing left- and right-reduced covers are given as Algorithms 4 and 5, respectively. Steps (1) and (2) must not be switched, and step (3) is necessary. Every reduced cover is non-redundant as every attribute on the right side of any redundant FD is extraneous. Algorithm 6 returns a reduced cover of F in time $\mathcal{O}(||F||^2)$.

Applying Algorithm 4 to G_1 above will result in G_2, and applying Algorithm 5 to G_2 will result in G_3. Since G_3 contains no FD with empty right side, G_3 is also returned by Algorithm 4 applied to G_1.

3.3 Canonical covers

An FD set F is *canonical* if every FD in F is of the form $X \to A$ and F is left-reduced and non-redundant. Canonical covers are reduced since every FD must be right-reduced due to having only single attributes as their right side.

Algorithm 7 computes a canonical cover of input F in time $\mathcal{O}(||F||^2)$. It computes a reduced cover and splits every FD with multiple attributes on the right side into separate FDs with the same left side and a single attribute on the right side. This yields a canonical cover. We obtain a reduced from a canonical cover by combining FDs with equal left sides into a single FD.

G_3 is a canonical cover of the reduced cover $G_4 = \{Emp \to \{Dep, HoD\}, Dep \to Mgr\}$. G_4 results from G_3 by combining $Emp \to Dep$ and $Emp \to HoD$ into the single FD $Emp \to \{Dep, HoD\}$.

3.4 Minimal covers

Reduced covers of an FD set F do not necessarily have as few FDs as some other cover for F. Hence, Maier defined an FD set F as *minimal* if F has no other cover with fewer FDs than F [29].

The set F with the FDs $Dep \to \{HoD, Mgr\}$, $HoD \to Dep$, $\{Dep, Emp\} \to Salary$, $\{HoD, Emp\} \to Position$ is non-redundant but not minimal. Indeed, the set G with FDs $Dep \to \{HoD, Mgr\}$, $HoD \to Dep$, $\{Dep, Emp\} \to \{Salary, Position\}$ is equivalent to F but has fewer FDs. Actually, G is a minimal cover for F.

Two attribute sets X and Y are *equivalent* under an FD set F, written $X \leftrightarrow Y$, if $X_F^+ = Y_F^+$. Let $E_F(X)$ be the set of FDs in F with left sides equivalent to X. Note that $E_F(X)$ is empty when no left side of any FD in F is equivalent to X. Then the set $\overline{E}_F = \{E_F(X) \mid X \subseteq R \text{ and } E_F(X) \neq \emptyset\}$ is always a partition of F.

Algorithm 8 starts with a non-redundant cover G of input F, and then checks if there is any $E_G(X)$ for which $Y \to U, Z \to V \in E_G(X)$ exist such that $G - E_G(X) \models Y \to Z$. In that case, the two FDs $Y \to U$ and $Z \to V$ can be replaced in G by the single FD $Z \to UV$, resulting in fewer FDs. For our example F above we have $Z = \{Dep, Emp\}$, $V = Salary$, $Y = \{HoD, Emp\}$ and $V = Position$.

Algorithm 8 can be implemented to compute a minimal cover for input FD set F in time $\mathcal{O}(||F|| \cdot |F|)$.

3.5 Minimal-reduced covers

Minimal covers may still exhibit extraneous attributes. Algorithm 9 applies Algorithm 6 to the result of Algorithm 8 to return a reduced, minimal cover. Algorithm 9 can be implemented to compute a minimal-reduced cover for input FD set F in time $\mathcal{O}(||F||^2)$.

3.6 Optimal covers

An FD set F is *optimal* if there is no cover of F with fewer attribute symbols than F. Maier showed that it is *NP*-complete to decide if a given FD set is optimal [29]. It is thus unlikely a deterministic algorithm exists that runs in time polynomial in the input.

Algorithm 10 OPTIMAL(F)

Require: Set F of FDs
Ensure: An optimal cover G for F
1: $G :=$ MINIMAL(MINI(F))
2: **return** G

Optimal covers can be computed as minimal covers of so-called mini covers [37]. An FD set F is *mini* if the right side of every FD in F is a single attribute, F has the fewest FDs and, within that constraint, the fewest attributes [37]. A mini cover can be computed by (1) transforming the input into a Boolean formula, (2) using a standard method to minimize the formula, and (3) transforming the minimized formula back into an FD set which will be mini.

The first Delobel-Casey transform [37] of an FD $\{A_1, \ldots A_m\} \rightarrow \{B_1, \ldots, B_r\}$ is the Boolean expression $(A_1 \wedge \cdots \wedge A_m \wedge \neg B_1) \vee \cdots \vee (A_1 \wedge \cdots \wedge A_m \wedge \neg B_r)$. For an FD set F, the first Delobel-Casey transform of F is a Boolean expression which is the conjunction of the first Delobel-Casey transform for each member. For example, let $F = \{Emp \rightarrow \{Dep, Mgr\}, Dep \rightarrow HoD\}$, then the transform of F is $b_F = (Emp \wedge \neg Dep) \vee (Emp \wedge \neg Mgr) \vee (Dep \wedge \neg HoD)$. Conversely, a Boolean formula can also be converted into a corresponding FD set. For example, the Boolean formula $b_G = (Emp \wedge Dep \wedge \neg Position) \vee (Emp \wedge Dep \wedge \neg Salary) \wedge (Position \wedge \neg Salary)$ can be converted into the corresponding FD set $G = \{\{Emp, Dep\} \rightarrow \{Position, Salary\}, Position \rightarrow Salary\}$.

Transforming FD sets into their Boolean formulae preserves equivalence, which means steps (1)-(3) above will indeed result in a mini cover. Optimal covers are minimal covers of mini covers, resulting in Algorithm 10 that runs in time worst-case exponential in the size $||F||$ of the input FD set F [37].

Correctness and time complexity result from any standard technique for minimizing Boolean formulae [20], and the insight above.

The set G with FDs $\{HoD, Emp\} \rightarrow Service\_role$, $\{Dep, Mgr\} \rightarrow HoD$ and $HoD \rightarrow \{Dep, Mgr\}$ is an optimal cover for FD set F with $\{Dep, Mgr, Emp\} \rightarrow Service\_role$, $\{Dep, Mgr\} \rightarrow HoD$, and the FD $HoD \rightarrow \{Dep, Mgr\}$. F is minimal-reduced, but not optimal.

4 Mixed covers of keys and FDs

We will introduce the main concept of our work, mixed covers, before discussing mixed variants for various types of FD covers, establishing important relationships between them, and showing that these already hold on schemata in Third Normal Form (3NF).

The last section has summarized different types of FD covers from the literature, such as non-redundant, reduced, canonical, minimal, minimal-reduced, and optimal. For each of these types t, we will now introduce the notion of a mixed cover.

Definition 1 For a set F of FDs, the set (K, G) is a *mixed cover* of type t if and only if K denotes the set of minimal keys implied by F, and G is an FD cover of type t for the set F' of FDs in F not implied by K, that is, $G \equiv F' = \{f \in F \mid K \not\models f\}$. □

Integrity maintenance in practice naturally leads to our definition of mixed covers. Indeed, when updates occur frequently, integrity should be enforced on normalized database schemata. State-of-the-art methods can normalize any given schema into a dependency-preserving 3NF decomposition, which will be in BCNF whenever possible [32]. If a schema is in dependency-preserving BCNF, all FDs can be enforced by minimal keys. Otherwise, integrity can be maintained by the set of minimal keys and some non-key FDs whose attributes on the right side are prime (that is, belong to some minimal key), which is the definition of 3NF [8, 9, 25]. For that reason, it is somewhat surprising that mixed notions of covers have not been studied before. Previous work has not even investigated FD covers over normalized schemata.

4.1 Boyce-Codd normal form

Let us consider relation schemata R that are in BCNF for a given FD set F. That is, for every non-trivial FD $X \rightarrow Y \in F$ it holds that $X \rightarrow R \in F^+$. Consequently, the set of minimal keys implied by F is sufficient and necessary to maintain the integrity on a schema R that is in BCNF for F.

Interestingly, all notions of an FD cover collapse into the set of minimal keys. Indeed, this set is a non-redundant cover of F, which is also optimal since we cannot find any other cover with fewer attribute occurrences. Hence, for every BCNF schema there is only one choice for a cover, namely the set of minimal keys.

For example, the schema $R = \{Emp, Dep, Mgr\}$ is in BCNF for the FD set F consisting of $\{Emp, Dep\} \rightarrow Mgr$ and $\{Emp, Mgr\} \rightarrow Dep$. Its set of minimal keys consists of $\{Emp, Dep\}$ and $\{Emp, Mgr\}$.

Given the situation for BCNF, it is natural to ask what notions of covers need to be considered when schemata are in 3NF. Here, for every non-trivial FD $X \rightarrow Y \in F$ it holds that $X \rightarrow R \in F^+$ or every attribute in Y must be prime. A dependency-preserving decomposition into 3NF is always possible, where dependency-preservation means that the original set F of FDs is implied by the union of FD sets that hold on elements of the decomposition. Hence, it is sensible to ask whether the different notions of FD covers are also different on schemata in 3NF.

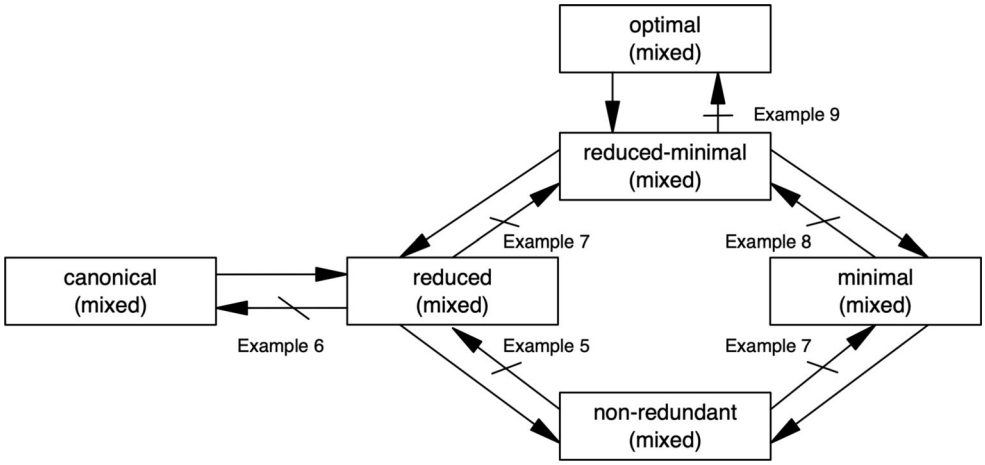


Fig. 3 Strict Relationships between Types of FD (Mixed) covers that already hold on schemata in 3NF

4.2 Relationships between types of mixed covers

We will now discuss the relationships between the different types of FD covers and their mixed variants. We will demonstrate that (1) all known relationships between FD covers already apply to schemata in 3NF, and are therefore all relevant, and (2) the same is true for our notions of mixed covers. In particular, (1) offers the first actual justification for using the different notions of FD covers for maintaining integrity, and (2) extends this to the actual use of mixed covers in practice, by maximizing the use of minimal keys and their UNIQUE indexes. The strict relationships are summarized in Fig. 3. A direct edge with no line on it means that every cover with the type of its origin is also a cover with the type of its destination, while the reversed edge has a line through it, indicating that the opposite direction does not hold. For instance, every optimal cover is reduced-minimal, but there are reduced-minimal covers that are not minimal.

The first example shows that redundant (mixed) covers for schemata in 3NF exist.

Example 4 Consider schema $R = ABCD$ with FD set $F = \{ABC \rightarrow D, CD \rightarrow B, D \rightarrow B\}$. The set K of minimal keys is ABC and ACD, so every attribute is prime, and (R, F) is in 3NF. The FD set $G = \{ABC \rightarrow D, D \rightarrow B\}$ is a non-redundant cover of F. Similarly, for the set $G' = \{D \rightarrow B\}$, (K, G') is a mixed non-redundant cover for the mixed cover (K, F') where $F' = \{CD \rightarrow B, D \rightarrow B\}$.

The next example shows that there are (mixed) covers that are non-redundant but not reduced on schemata that are in 3NF.

Example 5 Consider schema $R = ABCD$ with FD set $F = \{ABC \rightarrow D, CD \rightarrow B, D \rightarrow C\}$. The set K of minimal keys is ABC and AD, that is every attribute is prime, so (R, F) is in 3NF. F is non-redundant, but not reduced. FD set $G = \{ABC \rightarrow D, D \rightarrow B, D \rightarrow C\}$ is a reduced cover of F. For $G' = \{D \rightarrow B, D \rightarrow C\}$, (K, G') is a mixed reduced cover for the mixed non-redundant cover (K, F') where $F' = \{CD \rightarrow B, D \rightarrow C\}$ is not reduced.

The next example shows that there are (mixed) covers that are reduced but not canonical on schemata that are in 3NF.

Example 6 Consider schema $R = ABCD$ with FD set $F = \{ABC \rightarrow D, D \rightarrow BC\}$. As $K = \{ABC, AD\}$ every attribute is prime and (R, F) is in 3NF. F is reduced, but not canonical. FD set $G = \{ABC \rightarrow D, D \rightarrow B, D \rightarrow C\}$ is a canonical cover of F. For $G' = \{D \rightarrow B, D \rightarrow C\}$, (K, G') is a mixed canonical cover for mixed reduced cover (K, F') but $F' = \{D \rightarrow BC\}$ is not canonical.

The next example shows that there are (mixed) covers that are reduced but not minimal for schemata in 3NF. As every reduced cover is also non-redundant, the example also shows that there are (mixed) covers that are non-redundant but not minimal for schemata that are in 3NF.

Example 7 Consider schema $R = ABCD$ with FD set $F = \{ABC \rightarrow D, D \rightarrow B, D \rightarrow C\}$. K contains ABC and AD, so every attribute is prime and (R, F) is in 3NF. F is reduced and non-redundant, but not minimal. FD set $G = \{ABC \rightarrow D, D \rightarrow BC\}$ is a minimal cover of F. For $G' = \{D \rightarrow BC\}$, (K, G') is a mixed minimal cover for the mixed reduced and non-redundant cover (K, F') where $F' = \{D \rightarrow B, D \rightarrow C\}$ is not minimal.

The next example shows that there are (mixed) covers that are minimal but not reduced for schemata in 3NF.

Example 8 Let $R = ABCDE$ with $F = \{ABC \rightarrow DE, CDE \rightarrow AC, D \rightarrow C\}$. K contains ABC, ABD, and BDE, so every attribute is prime and (R, F) is in 3NF. F is minimal, but neither left- nor right-reduced. $G = \{ABC \rightarrow$

$DE, DE \to A, D \to C\}$ is a minimal-reduced cover of F. For $G' = \{DE \to A, D \to C\}$, (K, G') is a mixed minimal-reduced cover for the mixed minimal cover (K, F') where $F' = \{CDE \to BC, D \to C\}$ is neither left- nor right-reduced.

The final real-world example shows there are (mixed) covers that are minimal-reduced but not optimal for schemata in 3NF.

Example 9 Suppose users are assigned different IDs for each server, so R consists of *Fi(rstname)*, *L(astname)*, *I(D)*, *E(mail)*, *S(erver)*. The FD set

$$F = \{FiL \to E, E \to FiL, ES \to I, I \to FiL\}$$

is minimal-reduced, and (R, F) is in 3NF since K contains $FiLS$, ES, and IS. FD set

$$G = \{FiL \to E, E \to FiL, ES \to I, I \to E\}$$

is equivalent to F and has fewer attributes. Indeed, G is an optimal cover of F. For

$$G' = \{FiL \to E, E \to FiL, I \to E\},$$

(K, G') is a mixed optimal cover for the mixed minimal-reduced cover (K, F') where

$$F' = \{FiL \to E, E \to FiL, I \to E, I \to FiL\}$$

is minimal-reduced but not optimal.

4.3 Size bounds

While minimal-reduced covers can be computed in time quadratic in the input, the computation of optimal covers is exponential unless *P=NP*. Hence, minimal-reduced covers are computationally much more attractive than optimal ones. However, minimal-reduced covers are arbitrarily poor approximations of optimal covers. Hence, computing optimal covers provides us with the only assurance possible that we have not missed out on significant size savings.

The following theorem strengthens this previous result for FD covers [31] to schemata in 3NF. Indeed, the schema from [31] was not in 3NF, so our result did not follow automatically from there.

Theorem 1 *For all $c > 0$ there are a minimal reduced set F of FDs, a relation schema R in Third Normal Form for F, and an optimal cover G of F such that $\frac{||F||}{||G||} \geq c$.*

Proof Let $c > 0$. Consider FD set F_p with

- $A_1 \cdots A_p \to S$,
- $S \to A_1 \cdots A_p$,
- $SM \to I_1 \cdots I_p$, and
- $I_1 \to A_1 \cdots A_p, \ldots, I_p \to A_1 \cdots A_p$.

over schema $R_p = \{M, S, A_1, \ldots, A_p, I_1, \ldots, I_p\}$. For every fixed p, a S(ever) manages a unique set of p A(ccounts) $A_1, \ldots, A_p$, and a M(anager) oversees I(Ds) $I_1, \ldots, I_p$ that identify p people on that server, each of them having access to each of the p accounts. The set K_p of minimal keys consists of the $p + 2$ elements:

$$A_1 \cdots A_p M, \; SM, \; I_1 M, \; \ldots, \; I_p M.$$

Hence, every attribute of R_p is prime, and (R_p, F_p) is in 3NF. Clearly, F_p is minimal and reduced. Consider the FD set G'_p consisting of

- $A_1 \cdots A_p \to S$,
- $S \to A_1 \cdots A_p$,
- $SM \to I_1 \cdots I_p$, and
- $I_1 \to S, \ldots, I_p \to S$.

Clearly, $F_p \equiv G'_p$, and for the optimal cover G_p of G'_p we have $||G'_p|| \geq ||G_p||$ (in fact, $G_p = G'_p$). We have $||F_p|| = p^2 + 4p + 4$ and $||G'_p|| = 4p + 6$. Hence,

$$\frac{||F_p||}{||G_p||} \geq \frac{||F_p||}{||G'_p||} = \frac{p^2 + 4p + 4}{4p + 6} \geq \frac{p^2 + 4p + 4}{10p} \geq \frac{p}{10} \geq c$$

provided that $p \geq \max\{1, 10c\}$. □

In fact, we also show that the same result holds for our mixed variants. Hence, the trade-off in computing time and size savings between (mixed) optimal and (mixed) minimal-reduced covers applies to cases where integrity maintenance should happen in practice.

Theorem 2 *For all $c > 0$ there are a minimal reduced mixed set (K, F) of keys and FDs, a relation schema R in Third Normal Form for (K, F), and an optimal mixed cover (K, G) of (K, F) such that $\frac{||F||}{||G||} \geq c$.*

Proof Let $c \geq 0$. For the FD sets F_p and G'_p, and the set K_p of minimal keys over relation schema R_p, we can see that (K_p, G_p^m) is a mixed optimal cover of the mixed reduced and minimal set (K_p, F_p^m) over the schema R_p that is in 3NF for $F_p \equiv G_p$. Note that G_p^m (F_p^m) results from G'_p (respectively F_p) by removing the FD $SM \to I_1 \cdots I_p$. We have $||F_p^m|| = p^2 + 3p + 2$ and $||G_p^m|| = 3p + 4$. Hence,

$$\frac{||F_p^m||}{||G_p^m||} = \frac{p^2 + 3p + 2}{3p + 4} \geq \frac{p^2 + 3p + 2}{7p} \geq \frac{p}{7} \geq c$$

provided that $p \geq \max\{1, 7c\}$. □

5 Computation of mixed covers

We will now devise algorithms for computing mixed covers, one sequential and one parallel. We will also analyze their worst-case computational complexity, and that of closely related problems.

5.1 Sequential algorithm

We want to compute a mixed cover (K, G) of a given type t for a given set F of FDs. We do *not* assume input F is of target type t.

Algorithm 11 is sequential and works as follows. Firstly, we compute a type-t FD cover G' of input F. Then we compute the set K of minimal keys from G'. Finally, we obtain G by keeping those FDs of G' that are not keys. Since G' is a cover of F, K is indeed the set of minimal keys for F. The next result shows why the last step produces a type-t cover G for the set of non-key FDs for F.

Theorem 3 *Let G' be an FD cover of type t, K the set of minimal keys implied by G', and $G = \{\sigma \in G' \mid K \not\models \sigma\}$ the set of FDs from G' not implied by K. Then (K, G) is a mixed cover for G' of type t.*

Proof We show first that (K, G) is a cover of G'. Indeed, if $\sigma \in G'$ is not implied by K, then $\sigma \in G$. Hence, every $\sigma \in G'$ is implied by (K, G). Vice versa, every key in K is implied by G' and every $\sigma \in G \subseteq G'$, so all keys in K and FDs in G are implied by G'. Consequently, (K, G) and G' are covers of one another.

We now consider the different types t of FD covers.

- Let G' be a non-redundant cover. Suppose there is some $\sigma \in G$ that is redundant, that is, $G - \{\sigma\} \models \sigma$. Since $G \subseteq G'$, we would also have $G' - \{\sigma\} \models \sigma$ and G' would be redundant, a contradiction to our assumption that G' is non-redundant. Hence, (K, G) is a mixed cover that is non-redundant.
- Let G' be a left-reduced cover. Suppose there is some $X \to Y \in G$ and some $A \in X$ such that $G - \{X \to Y\} \models (X - \{A\}) \to Y$. Since $G \subseteq G'$, we would also have $G' - \{X \to Y\} \models (X - \{A\}) \to Y$ and G' would not be left-reduced, a contradiction to our assumption that G' is left-reduced. Hence, (K, G) is a mixed cover that is left-reduced.
- Let G' be a right-reduced cover. Suppose there is some $X \to Y \in G$ and some $A \in Y$ such that $G - \{X \to Y\} \models X \to (Y - \{A\})$. Since $G \subseteq G'$, we would also have $G' - \{X \to Y\} \models X \to (Y - \{A\})$ and G' would not be right-reduced, a contradiction to our assumption that G' is right-reduced. Hence, (K, G) is a mixed cover that is right-reduced.
- Let G' be a reduced cover. Suppose G is not reduced, so not left- or right-reduced. As in the two previous cases, this would imply that G' is not left- or right-reduced, respectively, a contradiction. Hence, (K, G) is a mixed cover that is reduced.
- Let G' be a canonical cover. Suppose G is not canonical. Consequently, it must be the case that a) there is some $\sigma \in G$ that is not of the form $X \to A$, or b) G is not left-reduced, or c) G is redundant. Since $G \subseteq G'$, sub-case a) would imply that G' is not canonical, and, just like in the previous cases, sub-case b) or sub-case c) would imply that G' is not left-reduced or redundant. In either sub-case, G' would not be canonical, a contradiction to our assumption. Hence, (K, G) is a mixed cover that is canonical.
- Let G' be a minimal cover. Suppose there is some FD cover G'' of G whose cardinality $|G''|$ is smaller than the cardinality $|G|$ of G, that is $|G'| < |G|$. Then $(G' - G) \cup G''$ is an FD cover of G' whose cardinality is smaller than $|G'|$. Consequently, G' would not be a minimal cover, in contradiction to our assumption. Hence, (K, G) is a mixed cover that is minimal.
- Let G' be a minimal-reduced cover. Suppose that G is not reduced or not minimal. As in the previous cases, it would follow that G' was not reduced or not minimal, correspondingly. Consequently, G' would not be a minimal-reduced cover, in contradiction to our assumption. Hence, (K, G) is a mixed cover that is minimal-reduced.
- Let G' be an optimal cover. Suppose there is some FD cover G'' of G whose size $||G''||$ is smaller than the size $||G||$ of G, that is $||G''|| < ||G||$. Then $(G' - G) \cup G''$ is an FD cover of G' whose size is smaller than $||G'||$. Consequently, G' would not be an optimal cover, in contradiction to our assumption. Hence, (K, G) is a mixed cover that is optimal. □

Correctness of Algorithm 11 follows by showing that FDs not implied by K are those whose LHS do not contain any key from K.

Lemma 1 *Let $Y \to Z$ denote a non-trivial FD over relation schema R, and let K denote a set of minimal keys for R. Then $K \models Y \to Z$ if and only if there is some $X \in K$ such that $X \subseteq Y$.*

Proof If there is some $X \in K$ such that $X \subseteq Y$, then $K \models X \to R$, and therefore $K \models Y \to R$, and therefore also $K \models Y \to Z$.

Vice versa, assume that for all $X \in K$, $X \not\subseteq Y$. We define the two-tuple relation $r = \{t, t'\}$ over R such that for all $A \in R, t(A) = t'(A)$ if and only if $A \in Y$. Since $Z \not\subseteq Y$ as $Y \to Z$ is non-trivial, we must have $R - Y \neq \emptyset$. Consequently, r does consist of two different tuples. Evidently, r does not satisfy

Algorithm 11 MIXED_SEQ(F)

Require: Set F of FDs, Type t of cover
Ensure: A mixed cover (K, G) for F of type t
1: $G' \leftarrow$ cover for F of type t using FD cover algorithm
2: $K \leftarrow$ the set of minimal keys implied by G' [28]
3: $G \leftarrow \{Y \rightarrow Z \in G' \mid \forall X \in K(X \not\subseteq Y)\}$
4: **return** (K, G)

Algorithm 12 MIXED_PARALLEL(F)

Require: Set F of FDs, Type t of cover
Ensure: A mixed cover (K, G) for F of type t
1: Compute G' and K in parallel:
$G' \leftarrow$ cover of F of type t using FD cover algorithm
$K \leftarrow$ the set of minimal keys implied by F [28]
2: $G \leftarrow \{Y \rightarrow Z \in G' \mid \forall X \in K(X \not\subseteq Y)\}$
3: **return** (K, G)

$Y \rightarrow Z$. However, r does satisfy K since $t(X) \neq t'(X)$ for every $X \in K$ because $X \not\subseteq Y$ and $X \cap (R - Y) \neq \emptyset$. Consequently, $K \not\models Y \rightarrow Z$. □

5.2 Parallel algorithm

Algorithm 11 computes the set K of minimal keys after computing G', which is a type-t FD cover of F. It makes sense to use G' since it is typically smaller than F. However, we can also compute K from the original FD set F. In this case, we do not need to wait until G' has been computed. That is, we could compute G' and K in parallel. This results in Algorithm 12.

Note that Algorithms 11 and 12 are guaranteed to return the same result since the set K' of minimal keys implied by G' and the set K of minimal keys implied by G coincide. Indeed, G' and G are covers of one another, so they determine the same set of minimal keys.

Illustrated by Fig. 4, Algorithm 12 runs faster than Algorithm 11 if and only if the computation of K from F is faster than the total of first computing G' from F and computing K from G' subsequently. As we will witness in the experiments, the two algorithms can have significantly different run times. One may execute both algorithms in parallel to ensure that maximum advantage is taken in terms of getting the result as fast as possible.

5.3 Computational complexity

Before we discuss the worst-case complexity of Algorithms 11 and 12, it is important to recall the complexity of computing the set K of minimal keys from a set F of FDs. Indeed, the problem of deciding whether or not there is a minimal key of cardinality not greater than a given positive integer is *NP*-complete [28]. Nevertheless, Osborne and Luccesi have devised an algorithm [28] for computing K

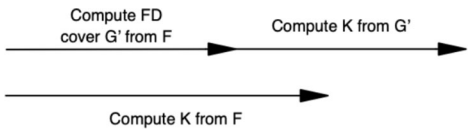


Fig. 4 Main parts of computing mixed covers

Algorithm 13 MIXED(G')

Require: Set G' of FDs of type t
Ensure: A mixed cover (K, G) for G' of type t
1: $K \leftarrow$ the set of minimal keys implied by G' [28]
2: $G \leftarrow \{Y \rightarrow Z \in G' \mid \forall X \in K(X \not\subseteq Y)\}$
3: **return** (K, G)

that is polynomial in R, F and K. While $|K|$ can be exponential in $|F|$, this only occurs rarely in practice, so K can often be computed efficiently. This will be confirmed by our experiments later.

Knowing K is of utmost importance to providing efficient access to data, both during update maintenance and query evaluation. Algorithms 11 and 12 compute a mixed cover of a target type t for a given FD set F, which are often efficient in practice.

Theorem 4 *Given FD set F, target type t over relation schema R, Algorithm 11 computes a mixed cover (K, G) for F of type t in time in $\mathcal{O}(|G'| \cdot |K| \cdot |R| \cdot (|K| + |R|) + C_t(F))$ where $C_t(F)$ denotes the function describing the complexity of computing an FD cover G' for F of type t. Similarly, Algorithm 12 computes a mixed cover (K, G) for F of type t in time in $\mathcal{O}(|F| \cdot |K| \cdot |R| \cdot (|K| + |R|) + C_t(F))$.*

Proof Theorem 3 shows that (K, G) with $G = \{\sigma \in F \mid K \not\models \sigma\}$ is a mixed cover for F of type t. Lemma 1 shows that G is equal to the set $\{Y \rightarrow Z \in F \mid \forall X \in K(X \not\subseteq Y)\}$. Hence, Algorithms 11 and 12 are both correct. Finding the set K of all minimal keys for F over R is in $\mathcal{O}(|F| \cdot |K| \cdot |R| \times (|K| + |R|))$ [28], and correspondingly for the FD cover G' of type t for F, and computing G from G' and K is in $\mathcal{O}(|G'| \cdot |K|)$, which gives the bound stated. □

For example, we have $C_{\text{minimal-reduced}}(F) = |F|^2$ and $C_{\text{optimal}}(F) = 2^{|F|}$ based on the worst-case complexity of FD cover computations.

5.4 Closely related computations

In case the given FD set G' is already of type t, computing a mixed cover of the same type reduces essentially to computing the set of minimal keys, due to Lemma 1. Algorithm 13 summarizes the two simple steps for this computation. Here, the worst-case complexity remains the same as that for computing K.

Corollary 1 *Given an FD cover G' of type t over relation schema R, Algorithm 13 computes a mixed cover (K, G) for F of type t in time that is in $\mathcal{O}(|G'| \cdot |K| \cdot |R| \cdot (|K| + |R|))$.* □

In case the target type is the same as the input type, and the set K of minimal keys is already given, the computation of a mixed cover is linear in G' and K.

Corollary 2 *Given FD cover G' of type t and the set K of minimal keys implied by G' over relation schema R, we can compute a mixed type-t cover (K, G) for G' in time $\mathcal{O}(|G'| \cdot |K|)$.* □

6 Experiments

We will now test the efficacy of our algorithms for computing (mixed) covers. We will state our research questions, summarize the datasets, and then answer each research question before we conclude.

6.1 Research questions

The main purpose of covers are savings on overheads required to maintain data integrity during updates. With this focus, it is interesting to ask the following research questions.

- Q1: What savings do notions of FD and mixed covers offer?
- Q2: What time does it take to compute these covers?
- Q3: What performance improvement do mixed covers achieve over FD covers on de-normalized and normalized databases?
- Q4: What impact does the use of mixed over FD covers have on industrial workloads regarding operations, constraints, and data volume?
- Q5: How much better can optimal covers be for update performance than minimal reduced covers?

6.2 Set up and datasets

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 the community edition of MySQL 8.0.29.

Data. We choose one synthetic (*fd-red*) and 16 real-world datasets that have been used as benchmarks for discovering database constraints from data, including FDs [33, 34, 46]. We use the results of discovery algorithms as inputs for computing FD covers and their mixed variants.

Given the variety of covers we are investigating, it is an interesting question in what format we present the FD set that is input to the algorithms. Clearly, we would like to avoid presentation in the form of any cover we want to investigate, yet the format should be standardized. As a simple solution, we present the input as set of FDs $X \to Y$ that hold on the underlying dataset and where X is a minimal attribute set for all the attributes in Y, and where left sides are unique (that is, no two different FDs with the same left side occur in the input FD set). That is, if X' results from X by removing some attribute, then none of the FDs $X' \to A$ with any $A \in Y$ holds on the underlying dataset. This format permits redundant FDs (and this is the case for all our datasets we consider), which means that each of the different notions of covers have an impact. All input FD sets and their output covers are available as part of our artifact.

Table 2 shows for each dataset, its numbers of rows (#*r*), columns (#*c*), number of valid FDs (#*fd*), their size (*fd-size*), the number of schemata in a lossless, depdendency-preserving decomposition into 3NF that is in BCNF if possible (#*norm*), and the percentage of schemata in BCNF (%bcnf). When nulls occur in the dataset, we append $\neq$ (=) to names of datasets to indicate FDs hold when we use the interpretation `null <> null` (`null = null`).

Integrity constraints, such as keys and FDs, encode rules that data ought to obey within the domain of application. The FDs mined from a given dataset include such rules but also FDs that only hold accidentally and for which integrity maintenance is unnecessary. Ideally, we would conduct experiments on real-world data over real-world schemata with real-world FDs. Unfortunately, these are hard to come by, but also not really necessary to gain the insight we want. The introduction has already looked at real-world schemata and FDs with synthetic data that satisfy precisely the FDs specified. Our research questions investigate overall trends for computing covers, such as the growth of runtime and output size in the input, and the update performance on non-normalized and normalized schemata of varying sizes with a variety of integrity constraints. Some of the datasets, such as routes, hospital, or bridges have fewer FDs and they all appear to be sensible. We believe that our datasets and FD sets do provide a good range of real-world schemata, real-world data, and real-world-like FDs.

When we measure runtime, we always report the average over hundred independent runs. This includes the experiments where we perform integrity checking using FDs (and keys) after inserting varying numbers of previously removed records into the given dataset. For experiments concerning schemata in 3NF, we used the state-of-the-art algorithm that returns a lossless, dependency-preserving 3NF decomposition that is in BCNF whenever possible [32], and used the sub-schemata which were in 3NF but not in BCNF together with the projection of the original database on the sub-schemata, for our experiments.

Table 2 Statistics of datasets used for experiments

Dataset	#r	#c	#fd	fd-size	#Norm	%bcnf (%)
Abalone	4,177	9	54	371	20	75
Adult	48,842	14	68	451	46	100
fd-red	250,000	30	3573	100,272	1341	100
Lineitem	6,001,215	16	901	8755	562	95
Breast ≠	699	11	43	207	39	95
Bridges ≠	108	13	67	379	44	84
Diabetic ≠	101,766	30	97,341	1,080,319	26703	91
Echo ≠	132	13	91	613	72	90
Hepatitis ≠	155	20	2995	21,215	1123	73
ncvoter ≠	1000	19	271	2166	162	87
pdbx ≠	17,305,799	13	37	226	18	72
Uniprot ≠	512,000	30	5794	49,574	1946	79
Weather ≠	262,920	18	2955	26,763	1154	66
Claims ≠	97,031	13	17	104	22	100
dblp ≠	10,000	34	708	3,688	294	82
Routes ≠	67,663	9	15	67	6	83
Hospital ≠	114,919	15	42	195	19	100

6.3 Savings

For our first research question we are interested in the savings that FD covers and our mixed variants accomplish, in both numbers and sizes. This was the original motivation for FD covers [29].

For each dataset, Table 3 lists the number and size of the FD covers, while Table 4 lists the number and size of the mixed variants. We terminate the computation of optimal covers if it is not completed after 4hrs.

Our main observations are: (1) the numbers and sizes quantify the relationships we expect to hold among FD covers, and between mixed variants. In particular, minimal covers achieve the lowest cardinality possible, reduced covers remove extraneous attributes, while optimal covers guarantee the smallest possible size; (2) for mixed covers, the number and size of minimal keys can be high, which means many FDs are implied by minimal keys; (3) the total in numbers and sizes are typically larger for mixed variants than they are for their FD covers.

Further to (3), Fig. 5 shows for each dataset, the percentage of savings in numbers and sizes. With rare exceptions, the savings in size exceed those in numbers, for both FD and mixed covers; and the savings by FD covers exceed those by mixed variants. Figure 5n is an outlier as the corresponding canoncial cover increases the number of FDs over the input (from 17 to 23), due to decomposing an FD with multiple attributes on its right-hand side into multiple FDs with a singleton right-hand side, and the fact many of these FDs are non-redundant.

Averaging over all datasets, Table 5 lists the average percentage of savings across different covers over the input set, for both FD and mixed covers, respectively. In particular, the savings in numbers for mixed variants are always smaller than those of the corresponding FD covers. This is the same in terms of sizes, but with a few exceptions (non-redundant and minimal covers).

In summary, FD covers typically achieve larger savings in numbers and sizes compared to their mixed variants.

6.4 Performance

Most applications are based on the set of FDs that have been identified as business rules for the underlying domain of application. This typically means that this set is quite stable, however, business rules may change over time. Whenever that happens, corresponding covers may require re-computation. It is therefore also an important criteria at which cost the savings of different covers can be accomplished. Here, we predominantly identify cost with the time required for computing covers.

Table 6 shows the time (in ms) required to compute the different notions of FD covers (*FD*), based on our implementations of algorithms from [30], and their mixed variants using Algorithms 11 (*mix-seq*) and 12 (*mix-par*), respectively. Our main observations are:

(1) The computation times quantify the relationships between original notions of FD covers and those of our mixed variants, respectively. In particular, savings by

Table 3 Numbers and size for FD covers by datasets (no entry means algorithm did not terminate after 4hrs)

Cover	Input		Non-Redundant		Reduced		Canonical		Minimal		Minimal-Reduced		Optimal	
Dataset	No	Size	No	Size	No	Size	No	Size	No	Size	No	Size	No	Size
Abalone	54	371	40	269	41	271	42	220	40	269	40	210	40	210
Adult	68	451	42	269	42	267	46	285	42	269	42	267	42	267
fd-red	3573	100,272	1550	46,042	1550	6203	1571	6260	1550	46,042	1550	6203		
Lineitem	901	8755	677	6284	679	4241	720	4416	677	6284	677	4229	677	4211
Breast	43	207	40	192	40	189	41	192	40	192	40	189	40	189
Bridges	67	379	50	278	52	239	56	247	50	278	50	230	50	228
Diabetic	97,341	1,080,319	62,249	656,454	62,381	653,339	64,014	665,279	62,249	656,454	62,249	652,166		
Echo	91	613	71	463	71	286	90	329	71	463	71	286	71	278
Hepatitis	2995	21,215	1439	9689	1450	9404	1530	9662	1439	9689	1439	9349		
ncvoter	271	2,166	159	1,069	169	744	205	830	159	1,069	159	704		
pdbx	37	226	21	113	22	92	26	99	21	113	21	90	21	88
Uniprot	5794	49,574	2140	14,855	2184	12,523	2401	13,273	2140	14,855	2140	12,320		
Weather	2955	26,763	1558	12,874	1575	11,882	1646	12,316	1558	12,874	1558	11,750	1558	11, 642
Claims	17	104	14	85	14	83	23	92	14	85	14	83	14	83
dblp	708	3688	310	1615	313	1286	342	1345	310	1615	310	1277		
Routes	15	67	6	22	6	20	7	23	6	22	6	20	6	20
Hospital	42	195	17	80	18	67	25	77	17	80	17	64	17	62

Table 4 Numbers and Size for Mixed Covers by Datasets (no entry means algorithm did not terminate after 4hrs)

Cover	Non-Redundant		Reduced		Canonical	
Dataset	No	Size	No	Size	No	Size
(a) Mixed Covers: Non-redundant, reduced, and canonical						
Abalone	(29,25)	(129,136)	(29,25)	(129,130)	(29,25)	(129,130)
Adult	(2,42)	(20,269)	(2,42)	(20,267)	(2,46)	(20,285)
fd-red	(3564,9)	(10692,32)	(3564,9)	(10692,21)	(3564,12)	(10692,24)
Lineitem	(390,432)	(2135,2718)	(390,433)	(2135,2659)	(390,458)	(2135,2778)
Breast	(2,40)	(10,192)	(2,40)	(10,189)	(2,41)	(10,192)
Bridges	(3,47)	(5,241)	(3,49)	(5,229)	(3,51)	(5,234)
Diabetic	(6530,62234)	(64378, 656124)	(6530,62365)	(64378, 653219)	(6530,63987)	(64378, 665138)
Echo	(39,45)	(111,191)	(39,45)	(111,180)	(39,56)	(111,204)
Hepatitis	(104,1433)	(499,9626)	(104,1443)	(499,9367)	(104,1523)	(499,9625)
ncvoter	(113,129)	(399,650)	(113,135)	(399,596)	(113,170)	(399,679)
pdbx	(11,17)	(39,70)	(11,17)	(39,68)	(11,17)	(39,68)
Uniprot	(850, 2093)	(3992,13627)	(850, 2135)	(3992,12291)	(850, 2346)	(3992,13035)
Weather	(523,1471)	(3814,11471)	(523,1481)	(3814,11109)	(523,1548)	(3814,11520)
Claims	(1,13)	(1,72)	(1,13)	(1,72)	(1,13)	(1,72)
dblp	(28,310)	(72,1615)	(28,313)	(72,1286)	(28,342)	(72,1345)
Routes	(2,5)	(7,16)	(2,5)	(7,16)	(2,6)	(7,19)
Hospital	(12,16)	(35,75)	(12,17)	(35,62)	(12,24)	(35,72)

Cover	Minimal		Minimal-Reduced		Optimal	
Dataset	No	Size	No	Size	No	Size
(b) Mixed Covers: Minimal, minimal-reduced, and optimal						
Abalone	(29,25)	(129,136)	(29,25)	(129,130)	(29,25)	(129,130)
Adult	(2,42)	(20,269)	(2,42)	(20,267)	(2,42)	(20,267)
fd-red	(3564,9)	(10692,32)	(3564,9)	(10692,21)		
Lineitem	(390,432)	(2135,2718)	(390,432)	(2135,2653)	(390,432)	(2135,2650)
Breast	(2,40)	(10,192)	(2,40)	(10,189)	(2,40)	(10,189)
Bridges	(3,47)	(5,241)	(3,47)	(5,220)	(3,47)	(5,219)
Diabetic	(6530,62234)	(64378, 656124)	(6530,62234)	(64378, 652059)		
Echo	(39,45)	(111,191)	(39,45)	(111,180)	(39,45)	(111,180)
Hepatitis	(104,1433)	(499,9626)	(104,1433)	(499,9316)		
ncvoter	(113,129)	(399,650)	(113,129)	(399,573)		
pdbx	(11,17)	(39,68)	(11,17)	(39,68)		
Uniprot	(850, 2093)	(3992,13627)	(850, 2093)	(3992,12098)		
Weather	(523,1471)	(3814,11471)	(523,1471)	(3814,11,036)	(523,1471)	(3814,10944)
Claims	(1,13)	(1,72)	(1,13)	(1,72)	(1,13)	(1,72)
dblp	(28,310)	(72,1615)	(28,310)	(72,1277)		
Routes	(2,5)	(7,16)	(2,5)	(7,16)	(2,5)	(7,16)
Hospital	(12,16)	(35,75)	(12,16)	(35,59)	(12,16)	(35,59)

minimizing numbers and sizes require additional times to obtain them. The cost of guaranteeing optimal sizes by optimal covers is worst-case exponential and that is visible in our results for larger inputs (Fig. 6).

(2) The computation of mixed from their corresponding FD covers also comes at a significant cost (Table 7). This, however, is expected since the problem of deciding whether a given attribute set is a minimal key for a given FD set is *NP*-complete. Emphasizing this point further, Table 8 shows the average percentage of additional time required to compute the mixed variant from the FD variant, across all datasets. Among the covers that can be computed in input-polynomial time (all but optimal covers), computing the mixed variant has a significant

(a) Abalone (b) Adult (c) FD-red (d) Lineitem

(e) Breast (*null≠null*) (f) Bridges (*null≠null*) (g) Diabetic (*null≠null*) (h) Echo (*null≠null*)

(i) Hepatitis (*null≠null*) (j) Ncvoter (*null≠null*) (k) PDBX (*null≠null*) (l) Uniprot (*null≠null*) (m) Weather (*null≠null*)

(n) Claims (*null≠null*) (o) DBLP (*null≠null*) (p) Routes (*null≠null*) (q) Hospital (*null≠null*)

Fig. 5 Percentage of savings for covers by datasets

Table 5 Percentage of savings for different covers

Dataset	All				Optimal terminated			
Measure	Number		Size		Number		Size	
Cover	FD	Mix	FD	Mix	FD	Mix	FD	Mix
Non-Redundant	39.8	25.3	42.7	45.7	33.7	21.9	36.4	39.2
Reduced	38.8	24.8	52.8	47.6	32.8	21.4	45.8	40.8
Canonical	29.6	20.3	49.9	45.7	20.9	16.7	42.1	39.6
Minimal	39.8	25.3	42.7	45.7	33.7	21.9	36.4	39.2
Minimal-Reduced	39.8	25.3	53.4	48.0	33.7	21.9	46.5	41.2
Optimal					33.7	21.9	46.9	41.3

overhead over the time for computing the corresponding FD cover. This, however, is intuitive since the problem of computing the set of all minimal keys from a given set of FDs is likely to have no polynomial time algorithm (unless *P=NP*). Since the problem of computing an optimal FD cover is also likely exponential, the overhead for computing a mixed variant of the optimal cover is less significant. We observe that the computation of mixed covers require significant overheads compared to FD covers.

(3) Sequential and parallel algorithms can exhibit significantly different runtimes. The former works well when

Table 6 Runtime (in ms) required to compute FD covers and mixed covers

Cover	Non-redundant			Reduced			Minimal			Minimal-reduced		
Dataset	FD	Mix-seq	Mix-par	FD	Mix-seq	Mix-par	FD	Mix-seq	Mix-par	FD	Mix-seq	Mix-par
Abalone	0.27	1.26	1.3	1.81	2.88	2.04	0.96	2.08	1.3	2.08	3.25	2.1
Adult	0.43	0.68	0.43	2.45	2.73	2.46	1.04	1.3	1.04	2	2.24	2
fd-red	4,426	103,343	315,090	152,891	222,541	315,926	8033	77,205	315,319	38,037	112,009	315,450
Lineitem	89.4	973.8	2,188.5	1,625	2,579.2	2,187.4	435.7	1,617.1	2,187.8	1,315.7	2,290.8	2,191
Breast	0.18	0.28	0.18	0.77	0.87	0.77	0.54	0.64	0.54	0.99	1.09	0.99
Bridges	0.36	0.54	0.37	3.24	3.48	3.25	1.77	2.01	1.78	3.71	3.97	3.72
Diabetic	2,998,498	8,788,805	10,849,996	26,190,534	32,287,042	26,204,724	11,231,958	17,350,185	11,245,740	20,454,337	26,563,510	20,468,352
echo	0.69	2.74	2.25	5.52	7.57	5.57	2.08	4.12	2.25	5.42	7.45	5.47
hepatitis	727.2	1101.2	1033.2	6427.4	7030.2	6434.4	2264.8	2857	2271.4	3231	3822.4	3236.6
ncvoter	13.5	56.22	79.44	125.38	179.69	126.2	30.48	85.56	79.64	38.12	75.91	79.28
pdbx	0.14	0.37	0.58	1.46	1.81	1.47	0.42	0.73	0.58	0.88	1.24	0.89
Uniprot	3,902	17,000.5	28,503	63,383	78,243	63,426	4,199	12,204	28,465	6,415	14,463.5	28,462
Weather	1355	5359	7237	6763	10,437	7229	1743	4971	7278	3058	6230	7273
Claims	0.03	0.06	0.05	0.2	0.25	0.2	0.14	0.19	0.14	0.33	0.38	0.33
dblp	50.78	65.01	50.92	380.02	401.71	380.52	117.3	136.58	117.37	184.23	203.43	184.46
Routes	0.03	0.05	0.03	0.1	0.12	0.1	0.04	0.06	0.04	0.05	0.07	0.05
Hospital	0.2	0.5	0.7	1.2	1.5	1.2	0.4	0.7	0.7	0.7	0.9	0.7

Table 7 Runtime (in ms) required to compute optimal FD covers and mixed optimal covers

cover	optimal		
dataset	FD	mix-seq	mix-par
Abalone	5.6	6.56	5.64
Adult	12,051	12,051.24	12,051
Lineitem	603,110	604,208	603,141
Breast	26.8	26.89	26.78
Bridges	4,098	4,099	4,098
Echo	2941	2943	2941
pdbx	4056	4057	4056
Weather	4,365,018	4,368,772	4,365,606
Claims	2945	2945	2945
Routes	8	8	8
Hospital	98,851	98,851	98,851

Table 8 Overhead for computing mixed variant in percent of time to compute FD covers

Overhead	All		Optimal Terminated	
Dataset/Cover	Seq	Par	Seq	Par
Non-Redundant	337.22	734.44	233.93	367.58
Reduced	28.86	9.71	30.91	5.22
Canonical	30.05	8.95	33.06	4.13
Minimal	135.01	320.43	87.79	80.22
Minimal-Reduced	54.42	82.54	40.68	20.17
Optimal			1.64	0.06

the input FD set F is *key-heavy*, that is, the difference in computing K from F over computing K from FD cover G' is larger than computing G' from F; while the latter works well in the other case when the input FD set is *key-light*. This shows in Fig. 7, which compares the runtime of the sequential relative to the parallel algorithm, in percent. Bars below 100% mean the sequential algorithm performed faster. In addition, Fig. 6 shows a detailed runtime comparison of the algorithms on each of the datasets.

In summary, the computation of mixed variants requires significant overheads compared to their corresponding FD covers, and results in sizes that are typically larger. Our algorithms can reduce these overheads when the input FD sets are heavy or light on keys. Both algorithms may be executed in parallel to obtain the result as quickly as possible. We will see in the next sections that mixed variants have a tremendous benefit over FD covers when it comes to integrity maintenance under updates.

6.5 Integrity maintenance on non-normalized tables

We will now address our third research question and show what time savings mixed variants achieve over their corresponding FD covers when integrity constraints are maintained. In this section, we will look at the original schemata of our datasets, which are not normalized and therefore represent schemata and data for analytics.

Figure 8 shows the times (in ms) for inserting records over the non-normalized schemata of our datasets. These are averaged across all FD covers, and all mixed covers, respectively. Note that the times are shown on a logarithmic scale. There were four update operations, which differ in the number of records inserted: For $i = 1, \ldots, 4$, ui inserted $i \times 10\%$ of records (after removing them first from the given dataset). In the figure, *avg-FD* refers to the average times taken across all FD covers, while *avg-mix* refers to the average times taken across all mixed covers.

The following main observations can be made. (1) For both FD covers and their mixed variants individually, times increase proportionally to the number of records inserted. (2) For each dataset and for each update operation, the time savings of the mixed variant is around one order of magnitude over their corresponding FD cover.

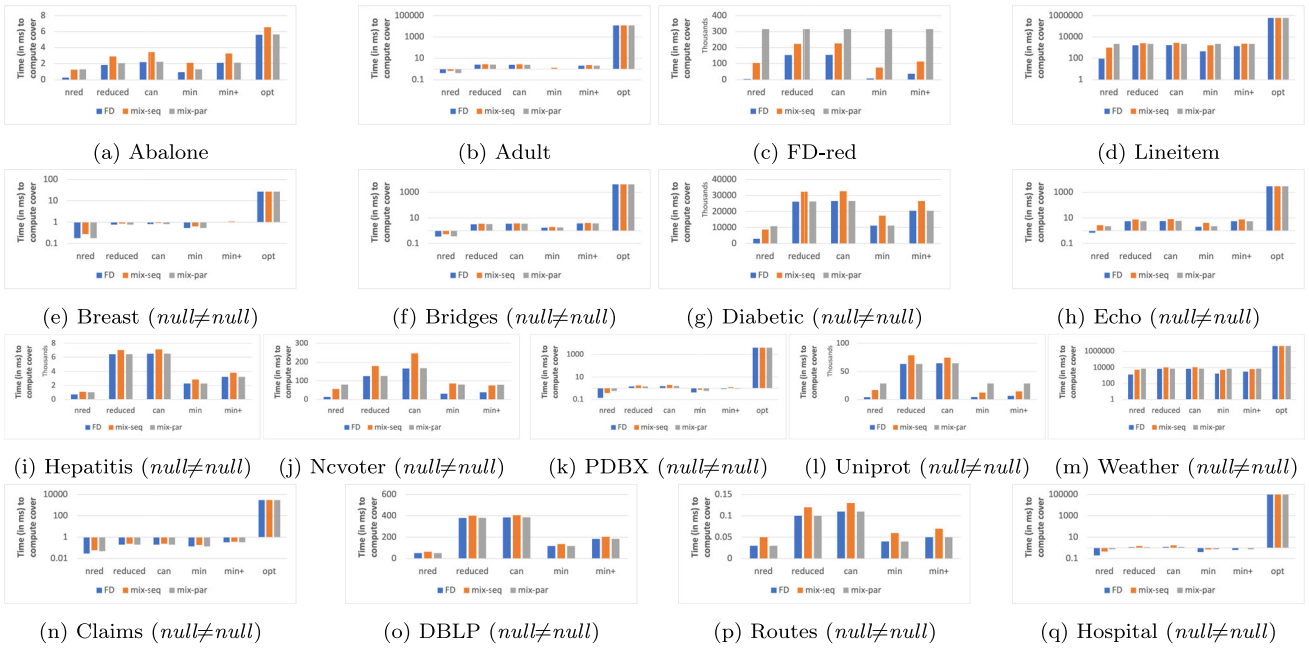
Analyzing point (2) further, Table 9 details the improvement of update performance by mixed variants over FD covers, in percent. The improvements are significant but can vary by quite a margin, that is, between approximately 12% and 98%. This is expected as the schema is not normalized. Indeed, there may still be many non-key FDs whose left-hand sides are not contained in any minimal key and whose right-hand side is not prime. Such FDs may not enjoy any speed up by the UNIQUE indices introduced by minimal keys.

However, the time savings are significant, and it should be stressed that these savings occur whenever updates are made. Due to demands from organizations to make real-time decisions, using active data warehouses or cloud architectures, updates also occur more frequently in analytical settings. Hence, computing more advanced notions of covers pays off, in particular as business rules change very rarely compared to the frequency of updates.

Figure 8 and Table 9 include results for the *bridges* data set with FDs mined under the `null = null` interpretation. While there may be differences between those FDs compared to the `null <> null` interpretation, their impact on update maintenance is typically very similar. The similarities are even stronger after normalization has taken place, as demonstrated next.

6.6 Integrity maintenance on normalized tables

We will now address a transactional setting where integrity is maintained on schemata resulting from decomposition into



(a) Abalone (b) Adult (c) FD-red (d) Lineitem

(e) Breast (*null≠null*) (f) Bridges (*null≠null*) (g) Diabetic (*null≠null*) (h) Echo (*null≠null*)

(i) Hepatitis (*null≠null*) (j) Ncvoter (*null≠null*) (k) PDBX (*null≠null*) (l) Uniprot (*null≠null*) (m) Weather (*null≠null*)

(n) Claims (*null≠null*) (o) DBLP (*null≠null*) (p) Routes (*null≠null*) (q) Hospital (*null≠null*)

Fig. 6 Times for computing covers by datasets

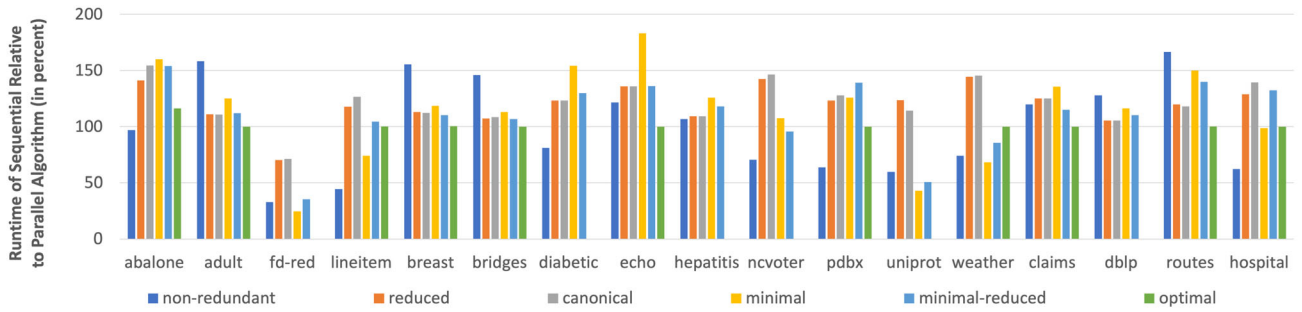


Fig. 7 Runtime comparison of sequential and parallel algorithms (sequential faster when percentage below 100)

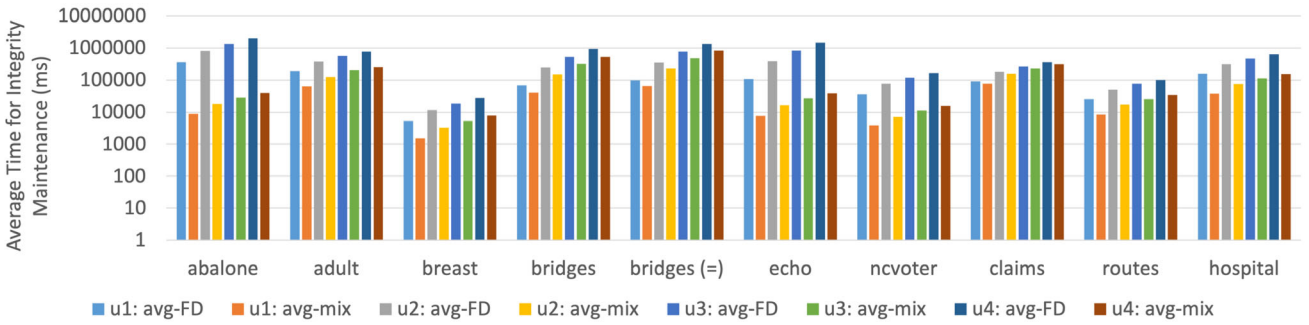


Fig. 8 Non-normalized schemata: average time in ms for integrity maintenance across FD and across mixed covers

Table 9 Non-normalized schemata: average boost of update performance by mixed over FD Covers

Measure	Average FD		Average mixed		Average update boost (%)			
Dataset	No	Size	No	Size	u1	u2	u3	u4
Abalone	42.4	252.3	(29,25)	(129, 132.6)	97.5	97.8	97.9	98.0
Adult	46.3	296.4	(2,42.6)	(20, 270.4)	66.7	66.3	63.8	66.4
Echo	76.6	388.3	(39, 46.6)	(111, 188.1)	92.9	95.7	96.8	97.4
Bridges	53.6	268.4	(3, 47.9)	(5, 232.1)	39.7	40.2	40.5	43.0
Bridges (=)	68	369.29	(5, 61)	(12, 326)	33.7	34.8	36.7	38.2
Breast	40.6	192.9	(2, 40.1)	(10,190.7)	71.2	72.1	71.4	71.4
ncvoter	178.7	1033.1	(113, 138.4)	(399, 629.6)	90.6	90.4	90.5	90.6
Claims	15.7	87.9	(1,13.4)	(1,74.7)	11.9	11.9	11.9	11.9
Routes	7.4	27.7	(2,6.3)	(7,22.1)	66.6	66.7	66.7	66.6
Hospital	21.9	89.3	(12,19.3)	(35,77.6)	73.5	73.7	73.8	73.9

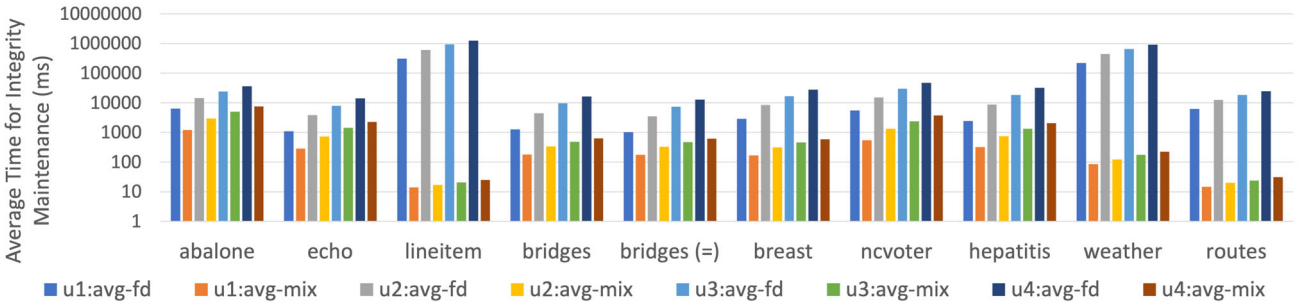


Fig. 9 Normalized schemata: average time in ms for integrity maintenance across FD and across mixed covers

Table 10 Normalized schemata: average boost of update performance by mixed over FD Covers

Measure	Average size		Average FD		Average mixed		Average update boost (%)			
Dataset	#Schema	#Records	No	Size	No	Size	u1	u2	u3	u4
Abalone	4	4176	5	26.5	(3.5,2)	(16,9.8)	96.4	97.0	97.3	97.6
Echo	7	126	2	8.17	(2, 7.2)	(1, 3.5)	84.4	91.4	94.0	95.5
Lineitem	2	6,001,214	3.5	21.7	(3,1)	(16.5,5.5)	99.9	99.9	99.9	99.9
Bridges	7	97	2.29	11.29	(2.1, 9.3)	(1, 4.1)	82.0	90.4	93.5	95.1
Bridges (=)	10	102	2.2	12.43	(2, 9.6)	(1.2, 5.6)	81.9	90.3	93.4	95.0
Breast	2	688	2	9	(2,8)	(1,4)	94.1	96.3	97.2	97.8
ncvoter	10	991	3.1	13.6	(2.3,9.1)	(1.4,5)	92.3	93.6	94.1	94.5
Hepatitis	10	144	2.7	17.7	(2.1,13)	(1.6,9.8)	89.0	93.4	94.9	95.7
Weather	2	262,684	4.5	33.5	(3,21)	(2.5,16.5)	99.9	99.9	99.9	99.9
Routes	1	67,599	2	7	(2,1)	(8,2)	99.7	99.8	99.8	99.9

lossless, dependency-preserving 3NF. For that purpose, we have computed lossless, dependency-preserving decompositions into 3NF [32], which are in BCNF whenever a lossless, dependency-preserving decomposition into BCNF exists.

Table 2 lists how many relation schemata each 3NF decomposition returned and the percentage of those in BCNF. Only 4 out of 17 datasets have a decomposition into BCNF, but only a small percentage of relation schemata is not in BCNF. However, these schemata constitute the bottleneck for integrity maintenance.

Figure 9 shows the average time in ms for maintaining integrity under insertions of records across FD and mixed covers on 3NF decompositions of schemata for our datasets. As some of the decompositions have many schemata, we limited our experiments to at most 10 sub-schemata, which we selected randomly. The update operations $u1, \ldots, u4$ are defined as before, but are now executed on the records projected to sub-schemata of the decomposition. Times are shown on a logarithmic scale.

Table 11 Numbers and sizes of integrity workloads for TPC-H experiments

Cover Workload	Optimal FD F		Mixed optimal (K, G)	
	$\|\|F\|\|$	$\|F\|$	$(\|\|K\|\|, \|\|G\|\|)$	$(\|K\|, \|G\|)$
25%	177	46	(109,53)	(43,14)
50%	205	52	(129,72)	(48,18)
75%	268	67	(165,97)	(58,25)
100%	281	70	(175,105)	(60,27)

Our main observations are similar to the case of non-normalized schemata. However, due to the normalization effort we are able to actually process updates on some of the larger datasets in the experiments (such as weather and lineitem). The improvement of time savings ranges from 3–5 orders of magnitude in these cases.

Table 10 details the improvement of update performance by mixed variants over FD covers, in percent. For each dataset, we list the number *#sch* of schemata in 3NF (but not in BCNF) analyzed and their average number *#records* of records, average numbers and sizes for FD and mixed covers on the sub-schemata as before. It should be stressed that 3NF decompositions are purposefully shifting as much of the semantics of FDs into keys, simply because non-key FDs cause data redundancy and keys prohibit them. As a consequence, a significant proportion of the original FDs can be enforced by minimal keys, which results in tremendous improvements for update efficiency. It is evident that normalization into 3NF leads to robust and significant speed ups for integrity maintenance, ranging between 81% and 99.99% here. Note that the validation of many non-key FDs can still benefit from UNIQUE indices as the left-hand sides of these FDs may be prefixes of these indices [23].

In conclusion, mixed covers and normalization work well together: Mixed covers speed up integrity maintenance and this is done robustly at orders of magnitude when schemata are in 3NF. Vice versa, 3NF schemata require mixed covers to benefit from UNIQUE indices resulting from minimal keys.

6.7 Impact on TPC-H benchmark

On each table of the TPC-H benchmark, we mined the set of FDs that hold on it. Their union F' consists 1038 FDs of size 9291. Its mixed optimal cover consists of 422 non-key FDs of size 631, and 2166 minimal keys of size 4088. We removed non-sensible FDs manually, for example those FDs including column *comments* on their LHS, and then ordered the remaining FDs starting with those we perceived most sensible (those that constitute primary key dependencies) to those least sensible. The resulting FD sets represent 100% of the constraint workload. We then took 25%, 50%, 75% of those workloads, starting from the most sensible ones.

Table 12 Average performance improvement (in percent) of query, refresh and insert opertations for optimal mixed over optimal FD covers under different constraint workloads CW and Scaling Factor SF

CW	Query	Refresh	Insert
(a) *SF=0.5*			
25%	20.8	99.9	99.9
50%	21.8	98.8	98.4
75%	19.1	98.8	98.4
100%	21.1	98.9	98.5
(b) *SF=1*			
25%	20.2	99.9	99.9
50%	20.1	99.2	99.0
75%	21.3	99.3	99.1
100%	22.0	99.3	99.1
(c) *SF=2*			
25%	22.2	99.9	100.0
50%	22.2	99.5	99.5
75%	22.1	99.5	99.4
100%	21.9	99.5	99.4
(d) *SF=3*			
25%	23.1	99.9	100.0
50%	22.5	99.6	99.5
75%	22.7	99.6	99.5
100%	22.1	99.6	99.5

Table 13 Total time savings (in seconds) of query, refresh and insert opertations for optimal mixed over optimal FD covers under different constraint workloads *CW* and Scaling Factor *SF*

CW	Query	Refresh	Insert
(a) *SF=0.5*			
25%	748	1,035	1,202
50%	738	4,877	8,298
75%	729	5,172	8,744
100%	731	5,395	9,938
(b) *SF=1*			
25%	1,450	2,029	2,346
50%	1,460	14,975	40,956
75%	1,446	16,054	44,121
100%	1,507	16,996	47,163
(c) *SF=2*			
25%	3,247	4,039	4,676
50%	3,337	42,348	137,524
75%	3,293	43,849	132,884
100%	3,301	45,206	140,754
(d) *SF=3*			
25%	6,004	6,005	7,349
50%	6,130	70,675	227,555
75%	6,033	71,914	238,026
100%	6,151	73,236	253,254

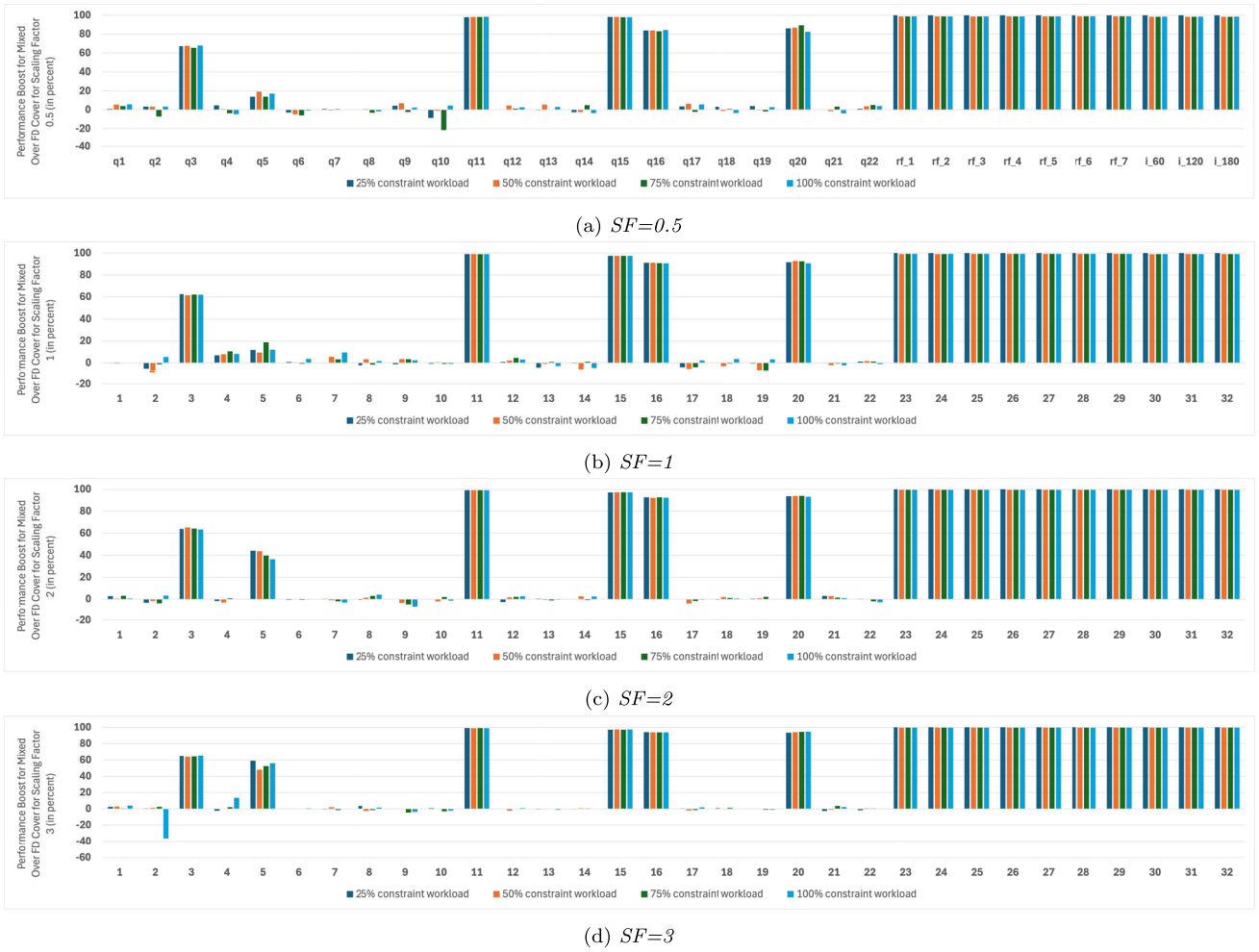


Fig. 10 Performance improvement of optimal mixed over optimal FD cover under different constraint workloads and scaling factors

Subsequently, computed an optimal FD cover and a mixed optimal cover for each of the sets. The numbers and sizes of the optimal and mixed optimal covers are summarized in Table 11.

FDs were implemented by triggers, and keys were implemented as UNIQUE constraints. For each of the constraint workloads, and each of the two covers, we ran the entire TPC-H benchmark including 22 query, 7 refresh, and 3 insert operations.

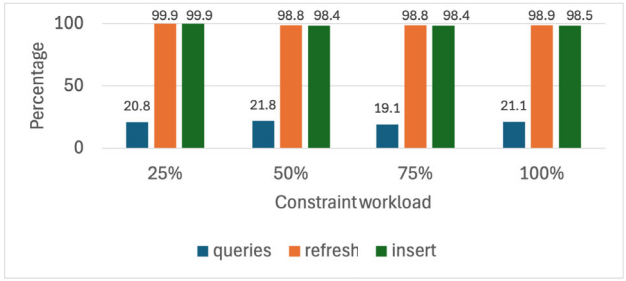
Figure 10 illustrates the performance improvement for each operation of the TPC-H benchmark gained by the use of mixed optimal covers in place of optimal FD covers, under each workload of constraints we considered. This is done for four different scaling facors: 0.5, 1, 2, and 3. Approximately half of the queries benefit significantly, while the use of mixed covers is essential for efficient integrity maintenance during refresh and insert operations. This is consistent through all constraint workloads and scaling factors considered.

Tables 12 and 13 quantify these observations in more detail.

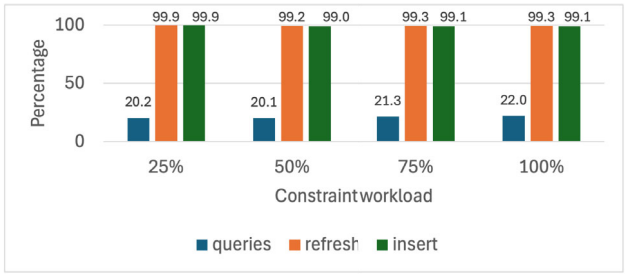
Table 12 shows the average performance improvement in percent over all 22 queries, over all seven refresh operations, and over all insert operations, under each of the four workloads for each of the four scaling factors considered, when optimal mixed covers are used instead of optimal FD covers. It is evident the average performance improvement in percent is very robust under the different constraint workloads and scaling factors considered. This is further illustrated in Fig. 11.

In addition, Table 13 lists the total time savings in *ms* when optimal mixed over optimal FD covers are used, again broken down into different constraint workloads and scaling factors. While the total time savings are robust under different constraint workloads, they grow with the underlying scaling factor, that is, the data volume. This is to be expected and further illustrated in Fig. 12.

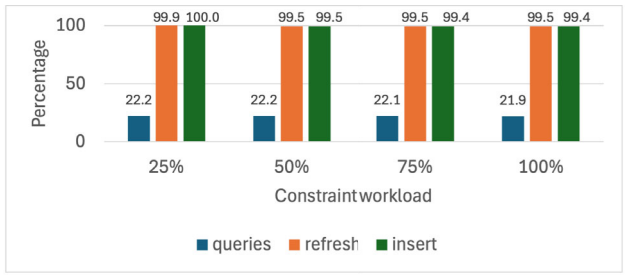
In terms of the improvement in percent by using mixed over FD covers, we obtain 21.6% for queries, 99.4% for refresh and 99.3% for insert operation when we average over all constraint workloads and all scaling factors.



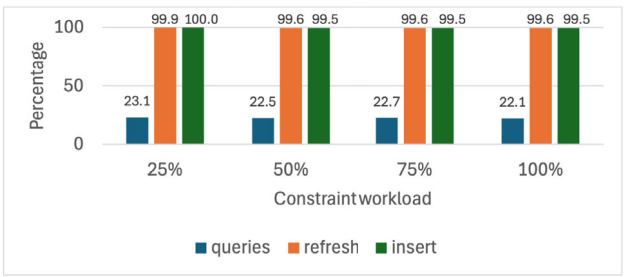
(a) Scale factor 0.5



(b) Scale factor 1

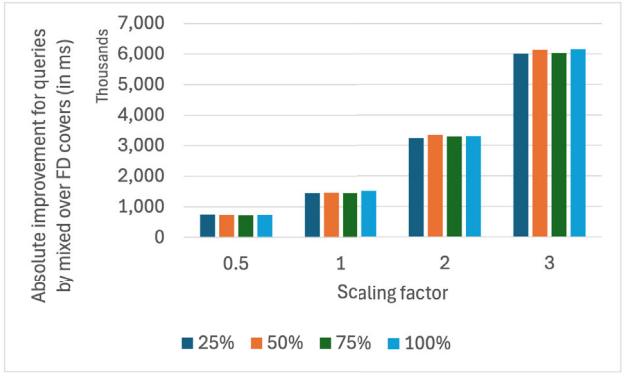


(c) Scale factor 2

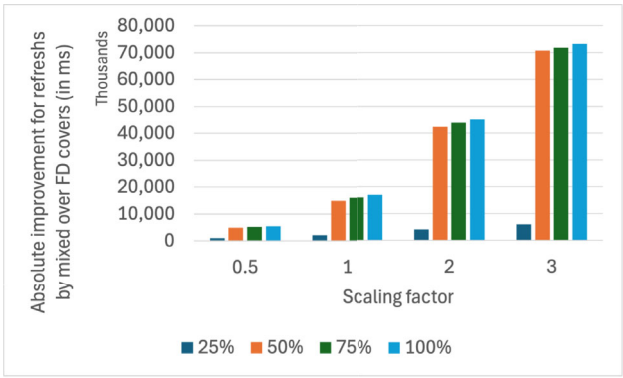


(d) Scale factor 3

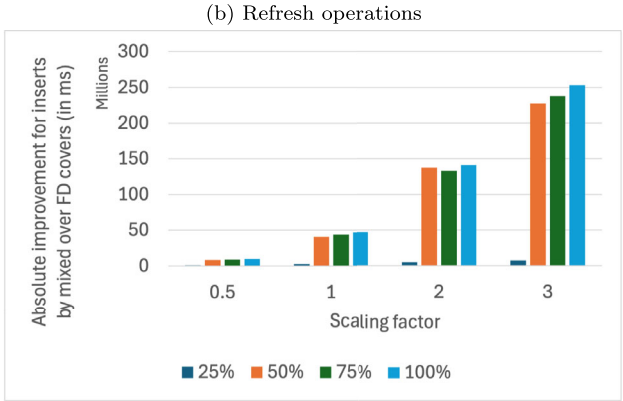
Fig. 11 Percentage of improvement for mixed over FD covers under different constraint workloads (x-axis) and scale factors



(a) Queries



(b) Refresh operations



(c) Insert operations

Fig. 12 Absolute improvement of queries, refreshs, and inserts by mixed over FD covers under different scaling factors (x-axis) and constraint workloads

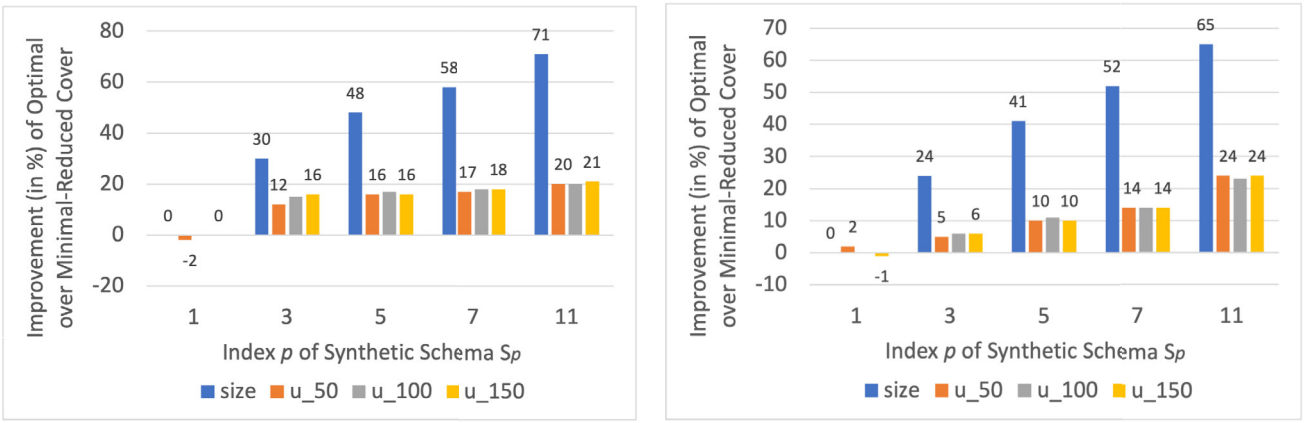
Regarding query evaluation times by using mixed over FD covers, we obtain savings of 12.28, 24.43, 54.91, and 101.33 min for scaling factors 0.5, 1, 2, and 3, respectively, each averaged over the 4 constraint workloads.

In terms of running refresh operations by using mixed over FD covers, we obtain savings of 1.14, 3.48, 9.41, and 15.4 h for scaling factors 0.5, 1, 2, and 3, respectively, each averaged over the 4 constraint workloads.

In terms of running insert operations by using mixed over FD covers, we obtain savings of 1.96, 9.35, 28.88, and 50.43 h for scaling factors 0.5, 1, 2, and 3, respectively, each averaged over the 4 constraint workloads.

6.8 Minimal-reduced vs optimal covers

Finally, we address our last research question and experimentally compare the two strongest notions: minimal-reduced and optimal covers, both for FD and mixed covers. While we have established their relationships in Theorem 1 and



(a) FD Covers (b) Mixed Covers

Fig. 13 Improvement of optimal over minimal-reduced covers in size and performance

Theorem 2, we would like to illustrate their difference in performance regarding integrity maintenance.

For our experiments, we use the schemata S_p for constraint sets F_p, G_p, F_p^m and G_p^m from the proofs of Theorems 1 and 2, for $p = 1, 3, 5, 7, 11$.

For each p, we computed Armstrong relations for each of the four constraint sets, and obtained three datasets by forming disjoint unions with 100k, 200k, and 300k copies of the relations. These three datasets satisfy the given constraint set and violate all FDs not implied by them. Hence, they form perfect sample data. We then completed five runs of three update operations where we inserted 50 (operation u_50), 100 (u_100), and 150 (u_150) records (that we previously removed), and averaged the times to complete the operations including integrity maintenance. We then averaged the ratio of times for minimal-reduced covers over times of optimal covers across the three datasets.

Figure 13 shows the improvement in size and update performance for each schema S_1, S_3, S_5, S_7 and S_{11}. In particular, Fig. 13a shows that of FD covers, and Fig. 13b for mixed covers.

The improvement in size is in line with Theorems 1 and 2, reaching 71% and 65% for mixed and FD covers for u_150. The improvement is growing

- from about 15% (6%) for mixed (FD) covers on u_50
- to around 20% (24%) on u_150.

So, not only is there a sizable difference between the two covers (for both FD and mixed covers), but also a significant difference in update performance.

What about the performance difference between FD and mixed covers that are minimal-reduced? For each p and update operation, mixed covers perform around 4 orders of magnitude faster than their FD covers, reaching a robust improvement by 99.99%.

6.9 Discussion

Optimal covers cannot improve the number of FDs in minimal-reduced covers, but the former may have fewer attributes than the latter. This improvement transcends to mixed covers, and to the operational level for the performance of update operations. The potentially significant differences were showcased on real-world like examples in our proofs and experiments. In this sense, the significant increase in complexity of computing optimal over minimal-reduced covers can be seen as overhead for getting to know the difference in size and performance. When inserting a new record, one must compare its values to those of all existing records across all attributes in the given cover. Hence, even minor improvements in cover size may lead to better performance in the validation process. In practice, minimal-reduced covers will most likely do a good job. Nevertheless, only the computation of optimal covers will provide assurance of having exhausted all opportunities for minimizing update overheads. Mixed covers take the minimization even further by first shifting as much of the application semantics as possible into minimal keys and their UNIQUE indices, which maximize the opportunities for efficient processing of updates and queries, and then applying standard notions of FD covers to non-key FDs.

7 Conclusion and future work

Starting from a simple question how FD covers can be used for integrity maintenance, we proposed our notion of mixed variants. Surprisingly, this simple extension bridges classical work on FD covers with that in database normalization

and integrity maintenance. Our results show that mixed variants provide the right notion for maintaining data integrity on 3NF schemata and elsewhere. We showed that their relationships known from previous work already hold on schemata in 3NF, and the same relationships apply to their mixed variants. We further illustrated that—while the numbers and sizes typically exceed that of their FD covers—mixed variants achieve orders of magnitude better update performance, on both normalized schemata in transactional settings and non-normalized schemata typical for analytical settings. We also highlighted that optimal covers have significantly less overheads for integrity maintenance than minimal-reduced covers, for both cases FD and mixed covers. Hence, computing mixed covers, in particular optimal ones if possible, is worth their time. Our sequential and parallel algorithms offer valuable alternatives in reducing the time to compute mixed covers. Finally, we quantified the significant reduction of query evaluation time, and the necessity of using mixed covers for refresh and insert operations on the TPC-H benchmark, in particular that total time savings grow with the underlying data volume. The work shows how simple changes of classical database notions can transform concepts from database theory into best practice.

Future work should look at appropriate notions of covers for more expressive constraints, such as variants of FDs [14, 38], denial constraints [36], order [42] or join dependencies [2]. Indeed, it is surprising that notions of covers have not even been studied for very common variants of FDs, such as approximate FDs that can handle inconsistency in the data. For that purpose, one may need to start with axiomatizations to obtain complete insight into their implication problem. This is necessary to understand which approximate FDs or which of their attributes are redundant. Similar arguments extend to approximate variants of other constraint classes, such as keys. While FDs have been extended from relational to most other data models, such as Web [47] or graph models [15, 40, 41], covers have not been studied yet. Naturally, our results lend themselves for extensions to more expressive data models, too. Practical work would introduce native support for specifying and maintaining FDs by relational database systems. This is required for schemata that are in 3NF but not BCNF.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

References

1. Abedjan, Z., Golab, L., Naumann, F., Papenbrock, T.: Data Profiling. Synthesis Lectures on Data Management. Morgan & Claypool Publishers (2018)
2. Aho, A.V., Beeri, C., Ullman, J.D.: The theory of joins in relational databases. ACM Trans. Database Syst. **4**(3), 297–314 (1979)
3. Atzeni, P., Morfuni, N.M.: Functional dependencies and constraints on null values in database relations. Inf. Control **70**(1), 1–31 (1986)
4. Beeri, C., Bernstein, P.A.: Computational problems related to the design of normal form relational schemas. ACM Trans. Database Syst. **4**(1), 30–59 (1979)
5. Beeri, C., Dowd, M., Fagin, R., Statman, R.: On the structure of Armstrong relations for functional dependencies. J. ACM **31**(1), 30–46 (1984)
6. Berti-Équille, L., Harmouch, H., Naumann, F., Novelli, N., Thirumuruganathan, S.: Discovery of genuine functional dependencies from relational data with missing values. Proc. VLDB Endow. **11**(8), 880–892 (2018)
7. Biskup, J., Dayal, U., Bernstein, P. A.: Synthesizing independent database schemas. In: SIGMOD, pages 143–151, (1979)
8. Codd, E. F.: Further normalization of the data base relational model. *Research Report / RJ / IBM / San Jose, California*, RJ909, (1971)
9. Codd, E. F.: Recent investigations in relational data base systems. In: Information Processing, Proceedings of the 6th IFIP Congress 1974, Stockholm, Sweden, August 5–10, 1974, pages 1017–1021, (1974)
10. Eich, M., Fender, P., Moerkotte, G.: Faster plan generation through consideration of functional dependencies and keys. Proc. VLDB Endow. **9**(10), 756–767 (2016)
11. Eich, M., Fender, P., Moerkotte, G.: Efficient generation of query plans containing group-by, join, and groupjoin. VLDB J. **27**(5), 617–641 (2018)
12. Fagin, R.: A normal form for relational databases that is based on domains and keys. ACM Trans. Database Syst. **6**(3), 387–415 (1981)
13. Fagin, R., Vardi, M. Y.: The theory of data dependenciesL: an overview. In: *Automata, Languages and Programming, 11th Colloquium, Antwerp, Belgium, July 16-20, 1984, Proceedings*, pages 1–22, (1984)
14. Fan, W., Geerts, F., Jia, X., Kementsietsidis, A.: Conditional functional dependencies for capturing data inconsistencies. ACM Trans. Database Syst. **33**(2), 6:1-6:48 (2008)
15. Fan, W., Wu, Y., Xu, J.: Functional dependencies for graphs. In: SIGMOD, pages 1843–1857, (2016)
16. Ge, C., Ilyas, I.F., Kerschbaum, F.: Secure multi-party functional dependency discovery. Proc. VLDB Endow. **13**(2), 184–196 (2019)
17. Greco, S., Molinaro, C., Spezzano, F.: Incomplete Data and Data Dependencies in Relational Databases. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, (2012)
18. Hartmann, S., Kirchberg, M., Link, S.: Design by example for SQL table definitions with functional dependencies. VLDB J. **21**(1), 121–144 (2012)
19. Hartmann, S., Link, S.: The implication problem of data dependencies over SQL table definitions: axiomatic, algorithmic and logical characterizations. ACM Trans. Database Syst. **37**(2), 13:1-13:40 (2012)
20. Jain, T. K., Kushwaha, D. S., Misra, A. K.: Optimization of the Quine-McCluskey method for the minimization of the boolean

expressions. In: Fourth international conference on autonomic and autonomous systems (ICAS'08), pages 165–168. IEEE, (2008)
21. Koehler, H., Link, S.: Entity/relationship profiling. In: 40th IEEE international conference on data engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024, pages 5393–5396, (2024)
22. Kossmann, J., Papenbrock, T., Naumann, F.: Data dependencies for query optimization: a survey. VLDB J. **31**(1), 1–22 (2022)
23. Lahdenmäki, T., Leach, M.: Relational Database Index Design and the Optimizers: DB2. SQL Server, et al. Wiley, Oracle (2005)
24. Langeveldt, W., Link, S.: Empirical evidence for the usefulness of Armstrong relations in the acquisition of meaningful functional dependencies. Inf. Syst. **35**(3), 352–374 (2010)
25. Levene, M., Loizou, G.: A Guided Tour of Relational Databases and Beyond. Springer (1999)
26. Link, S., Wei, Z.: 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, pages 1169–1181, (2021)
27. Liu, X., Wang, S., Sun, M., Pan, S., Li, G., Jha, S., Yan, C., Yang, J., Lu, S., Cheung, A.: Leveraging application data constraints to optimize database-backed web applications. Proc. VLDB Endow. **16**(6), 1208–1221 (2023)
28. Lucchesi, C.L., Osborn, S.L.: Candidate keys for relations. J. Comput. Syst. Sci. **17**(2), 270–279 (1978)
29. Maier, D.: Minimum covers in relational database model. J. ACM **27**(4), 664–674 (1980)
30. Maier, D.: The Theory of Relational Databases. Computer Science Press (1983)
31. Mannila, H., Räihä, K.: On the relationship of minimum and optimum covers for a set of functional dependencies. Acta Inform. **20**, 143–158 (1983)
32. Osborn, S.L.: Testing for existence of a covering Boyce-Codd normal form. Inf. Process. Lett. **8**(1), 11–14 (1979)
33. Papenbrock, T., Ehrlich, J., Marten, J., Neubert, T., Rudolph, J., Schönberg, M., Zwiener, J., Naumann, F.: Functional dependency discovery: an experimental evaluation of seven algorithms. Proc. VLDB Endow. **8**(10), 1082–1093 (2015)
34. Papenbrock, T., Naumann, F.: A hybrid approach to functional dependency discovery. In: Proceedings of the 2016 International conference on management of data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016, pages 821–833, (2016)
35. Pena, E.H.M., de Almeida, E.C., Naumann, F.: Discovery of approximate (and exact) denial constraints. Proc. VLDB Endow. **13**(3), 266–278 (2019)
36. Pena, E.H.M., Porto, F., Naumann, F.: Fast algorithms for denial constraint discovery. Proc. VLDB Endow. **16**(4), 684–696 (2022)
37. Peng, X., Xiao, Z.: Optimal covers in the relational database model. Acta Inf. **53**(5), 459–468 (2016)
38. Qahtan, A.A., Tang, N., Ouzzani, M., Cao, Y., Stonebraker, M.: Pattern functional dependencies for data cleaning. Proc. VLDB Endow. **13**(5), 684–697 (2020)
39. Saxena, H., Golab, L., Ilyas, I.F.: Distributed implementations of dependency discovery algorithms. Proc. VLDB Endow. **12**(11), 1624–1636 (2019)
40. Skavantzos, P., Link, S.: Normalizing property graphs. Proc. VLDB Endow. **16**(11), 3031–3043 (2023)
41. Skavantzos, P., Link, S.: Entity/relationship graphs: Principled design, modeling, and data integrity management of graph databases. Proc. ACM Manag. Data **3**(1), 40:1-40:26 (2025)
42. Szlichta, J., Godfrey, P., Golab, L., Kargar, M., Srivastava, D.: Effective and complete discovery of order dependencies via set-based axiomatization. Proc. VLDB Endow. **10**(7), 721–732 (2017)
43. Thalheim, B.: Dependencies in Relational Databases. Teubner (1991)
44. Wei, Z., Link, S.: Discovery and ranking of functional dependencies. In: ICDE, pages 1526–1537, (2019)
45. Wei, Z., Link, S.: Embedded functional dependencies and data-completeness tailored database design. ACM Trans. Database Syst. **46**(2), 7:1-7:46 (2021)
46. Wei, Z., Link, S.: Towards the efficient discovery of meaningful functional dependencies. Inf. Syst. **116**, 102224 (2023)
47. Yu, C., Jagadish, H. V.: Efficient discovery of XML data redundancies. In: VLDB, pages 103–114, (2006)
48. Zaniolo, C.: Database relations with null values. J. Comput. Syst. Sci. **28**(1), 142–166 (1984)
49. Zhang, Z., Chen, W., Link, S.: Composite object normal forms: Parameterizing Boyce-Codd normal form by the number of minimal keys. Proc. ACM Manag. Data **1**(1), 13:1-13:25 (2023)
50. Zhang, Z., Link, S.: Mixed covers of keys and functional dependencies for maintaining the integrity of data under updates. Proc. VLDB Endow. **17**(7), 1578–1590 (2024)