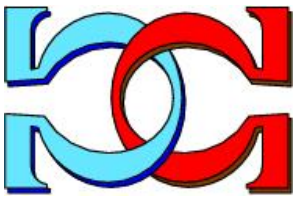
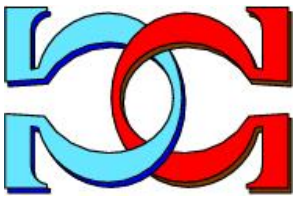
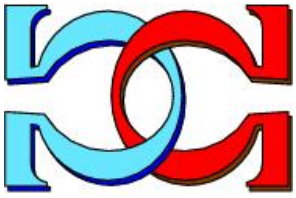


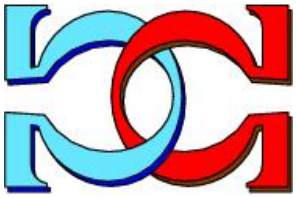
CDMTCS
Research
Report
Series



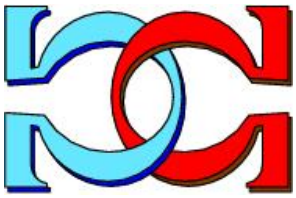
Relational Database Schema Design for Uncertain Data



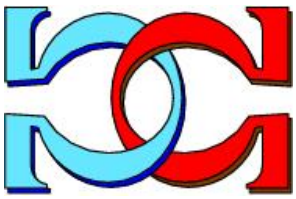
Sebastian Link
University of Auckland,
Auckland, New Zealand



Henri Prade
IRIT, CNRS and
Université de Toulouse III,
Toulouse, France



CDMTCS-469
21 August 2014



Centre for Discrete Mathematics and
Theoretical Computer Science

Relational Database Schema Design for Uncertain Data

Sebastian Link
The University of Auckland, Private Bag 92019, New Zealand
s.link@auckland.ac.nz

Henri Prade
IRIT, CNRS and Université de Toulouse III, France
prade@irit.fr

August 21, 2014

Abstract

We investigate the impact of uncertainty on relational database schema design. Uncertainty is modeled qualitatively by assigning to tuples a degree of possibility with which they occur in a relation, and assigning to functional dependencies a degree of certainty which reflects to which tuples they apply. A design theory is developed for possibilistic functional dependencies, including efficient axiomatic and algorithmic characterizations of their implication problem. Naturally, the possibility degrees of tuples result in a scale of different degrees of data redundancy, caused by functional dependencies that hold with the corresponding degrees of certainty. Scaled versions of the classical syntactic Boyce-Codd and Third Normal Forms are established and semantically justified in terms of avoiding data redundancy of different degrees. Classical decomposition and synthesis techniques are scaled as well. Therefore, possibilistic functional dependencies do not just enable designers to control the level of data integrity targeted but also to balance the classical trade-off between query and update efficiency. All algorithms are implemented in a Web-based graphical user interface that is linked to a high-performance computing cluster on which detailed experiments have been run. These do not just confirm the efficiency of our framework, but also provide original insight into classical relational database schema design.

Keywords: Data redundancy, Functional dependency, Normal form, Possibility theory, Relational database, Schema design, Uncertainty

1 Introduction

Relational databases were developed for applications in which data occurs with full certainty, such as accounting, inventory, and payroll. Modern applications such as infor-

mation extraction, big data, data integration and cleaning, require techniques that can process uncertain data. Research on uncertain data has been prolific, yet two trends can be observed: Query processing is the dominant focus point, and uncertainty is mostly modeled quantitatively in terms of probabilistic data. Here, the impact of uncertainty on database schema design is investigated qualitatively, targeting the efficient processing of frequent queries and updates.

In classical schema design update inefficiencies are avoided by removing potential data redundancy caused by functional dependencies. Intuitively speaking, if data is uncertain, then so is any redundancy that results from this data. The more possible it is for data redundancy to occur, the more functional dependencies can cause this redundancy, and the harder the normalization effort will be to remove that redundancy. In other words, the removal of redundancy from less possible data only requires normalization with respect to a smaller number of functional dependencies. This, however, is great news as data that are less possible are intuitively subject to more updates. Therefore, most of the frequent updates can be supported efficiently with less normalization effort. In turn, less normalization results in better query efficiency. It is the goal of our article to establish a practical and precise framework that enables database designers to take full advantage of this intuitive impact of uncertainty.

For that purpose we model uncertainty by assigning to each tuple a degree of possibility by which it is perceived to occur. The possibility degree comes from any given finite scale of linearly ordered values. The framework meets well the intuition and ability of people to reason qualitatively, and not quantitatively. After all, humans like to classify items into a few categories and are uncomfortable with assigning exact values such as probabilities. The framework results in a nested chain of possible worlds, each of which is a classical relation. The smallest world contains only tuples that are fully possible while the largest world contains all tuples, excluding those which are impossible to occur. Moreover, only the smallest world is regarded as the part of the database which is certain. The certainty by which a classical functional dependency holds is derived from the possibility degree of the smallest world in which the functional dependency is violated. On the extreme ends we have functional dependencies that are fully certain as they even hold in the largest world, and we have functional dependencies that are not certain at all as they do not even hold in the smallest world. Our contributions are as follows:

1. We formalize uncertainty by assigning degrees of possibilities to tuples in a database. The approach is well-founded as it results in a possibility distribution over possible worlds that form a linear chain of relations.

2. We define possibilistic functional dependencies (PFDs) as classical functional dependencies (FDs) with a degree of certainty, derived from the possibility degree of the smallest possible world in which it is violated.

3. We establish a full design theory for PFDs, including axiomatic and algorithmic characterizations of their associated implication problem, as well as algorithms for computing covers and the maximum certainty degrees by which PFDs are implied.

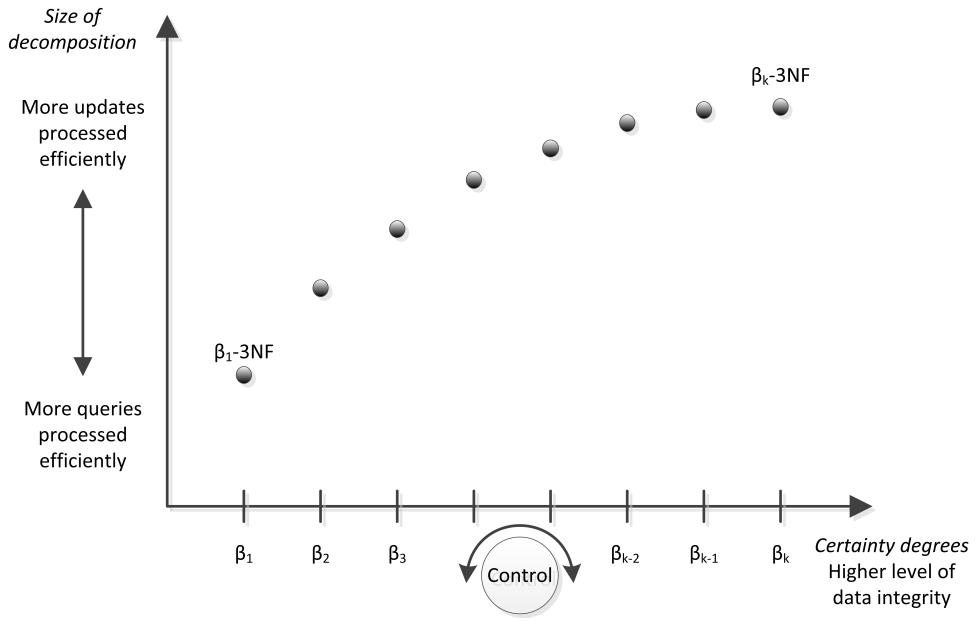


Figure 1: Exploring certainty degrees as a mechanism to control levels of data integrity, query efficiency, and update efficiency

The possibilistic framework allows us to establish strong links between PFDs and classical FDs, and thereby show the efficiency of our characterizations.

4. We apply the design theory to establish a new normalization framework, including scaled versions of Boyce-Codd (BCNF) and Third normal forms (3NF), their semantic justification in terms of dependency-preservation and removal of data redundancy, as well as normalization algorithms. Our techniques provide database designers with a full scale of normalized schemata, from which better-informed selections of the final schema can be made. We envision two main use-case scenarios: i) An organization that is primarily focused on data integrity may choose the FDs that apply to the target degree of certainty. In this case, our algorithms compute the normal forms that exactly meet that target. ii) For an organization that is primarily focused on efficiently processing its workload in terms of queries and updates, a database designer can use our techniques to derive the normal forms that best match the workload. In that case, it is also clear what level of data integrity is achieved. Therefore, our findings show that certainty degrees of FDs provide an efficient mechanism to not only control the degree of data integrity targeted, but also to derive designs that match the workload of the target database by balancing the trade-off between query and update efficiency. This is illustrated in Figure 1, showing how the selection of different certainty degrees β_i results in different decompositions, for example, β_i-3NF for $i = 1, \ldots, k$. In particular, our findings can be used to also justify classically de-normalized schema in terms of permitting data redundancy which is caused by FDs whose degree of certainty is smaller than that targeted.

5. We present a web-based graphical user interface in which implementations of all our algorithms can be accessed. The GUI also offers access to a high-performance computing cluster to run experiments with our algorithms.

6. Extensive experiments with our algorithms confirm their efficiency in practice, and provide new insight into classical normalization trade-offs. For example, we provide first empirical evidence that there almost always is a fair chance that 3NF synthesis results in an optimal decomposition, defined as a lossless, dependency-preserving BCNF decomposition, while there is hardly ever a reasonable chance that the BCNF decomposition algorithm results in a dependency-preserving decomposition. To the best of our knowledge, this is the first empirical evidence that explains why practitioners prefer 3NF synthesis to BCNF decomposition.

Organization. A running example is introduced in Section 2, on which we motivate our framework and illustrate concepts and findings. Related work is discussed in Section 3 and further motivates the need for our findings. Our framework to model uncertainty on the tuple level is introduced in Section 4. The semantics of PFDs is defined in Section 5, and the design theory for PFDs is established in Section 6. Scaled versions of BCNF and 3NF are defined in Section 7, and their semantic justifications are derived. Normalization algorithms are established in Section 8. The GUI with access to all our algorithms is introduced in Section 9, and the results of our experiments are discussed in Section 10. Section 11 concludes and comments on future work.

2 Motivating Example

As a simple example we consider a toy application in which an employee extracts information from web-sites about possible weekly project meetings in her company. Attributes of interest are *Project*, storing projects with unique names, *Time*, storing the weekday and start time of the meeting, *Manager*, storing the managers of the project that attend the meeting, and *Room*, storing the unique name of a room. The employee classifies the possibility with which tuples occur in the relation according to the web-site from which the information originates. Tuples gathered from the official project meeting's web-site are assigned possibility degree α_1, indicating that they are fully possible, tuples from a project manager's web-site are assigned α_2, tuples from a project member's web-site get degree α_3, and tuples that originate from rumors are assigned degree α_4. Implicitly, any other tuple has degree α_5, indicating that it is impossible to occur. The classification could have also resulted from a different interpretation, say already held meetings are classified as α_1, confirmed meetings are classified as α_2, requested meetings as α_3, planned meetings as α_4, and all other meetings as α_5. The point is that the employee has chosen 5 different degrees of possibility $\alpha_1 > \alpha_2 > \alpha_3 > \alpha_4 > \alpha_5$ to assign qualitatively different levels of uncertainty to tuples, where α_1 denotes the top and α_5 the bottom degree, respectively. Of course, the degrees can also be interpreted numerically, e.g. $1 > 0.75 > 0.5 > 0.25 > 0$. Table 1 shows a possibilistic relation, that is, a classical relation where a possibility degree is assigned to each tuple.

Table 1: Possibilistic Relation

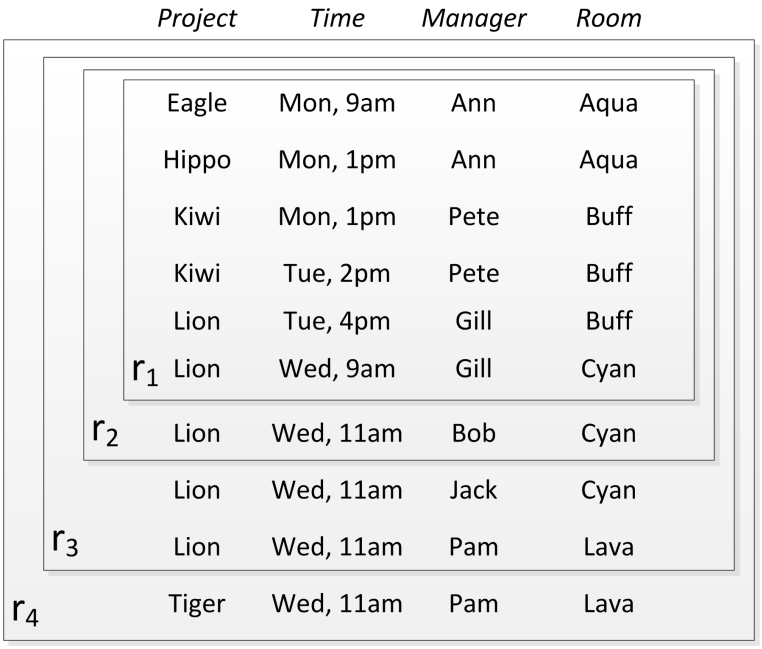
Project	*Time*	*Manager*	*Room*	*Poss. degree*
Eagle	Mon, 9am	Ann	Aqua	α_1
Hippo	Mon, 1pm	Ann	Aqua	α_1
Kiwi	Mon, 1pm	Pete	Buff	α_1
Kiwi	Tue, 2pm	Pete	Buff	α_1
Lion	Tue, 4pm	Gill	Buff	α_1
Lion	Wed, 9am	Gill	Cyan	α_1
Lion	Wed, 11am	Bob	Cyan	α_2
Lion	Wed, 11am	Jack	Cyan	α_3
Lion	Wed, 11am	Pam	Lava	α_3
Tiger	Wed, 11am	Pam	Lava	α_4

Naturally, the assignment of possibility degrees results in a linearly ordered chain of possible worlds: For $i = 1, \ldots, 4$, the relation r_i consists of tuples with possibility degree α_i or higher, i.e. α_j with $j \leq i$. The possibility degree of world r_i is α_i. In particular, fully possible tuples occur in every possible world, and are thus also fully certain to occur. The possible worlds of the possibilistic relation in Table 1 are illustrated in Figure 2.

An arbitrary FD is either satisfied by the largest possible world or there is a smallest possible world in which it is violated. For example, the FD *Manager*, *Time* $\rightarrow$ *Room* is satisfied by the world r_4, and thus holds in every possible world. Consequently, it is assigned the highest degree of certainty, denoted by β_1. The smallest relation that violates *Room*, *Time* $\rightarrow$ *Project* is r_4, that is, the FD is assigned the second highest degree of certainty, β_2. The smallest relation that violates *Project* $\rightarrow$ *Manager* is r_2, which means that the FD holds with certainty degree β_4. The FD *Manager, Room* $\rightarrow$ *Time* is violated even by the smallest possible world r_1, and is thus assigned the bottom certainty degree $\beta_5 = \beta_{k+1}$. Hence, the possibility degree α_i of the smallest possible world r_i in which the FD is violated, determines the certainty degree β_{k+2-i} with which the FD holds.

Clearly, the higher the possibility of a possible world the less tuples it contains and the more FDs may hold on it, and therefore, the more data occurrences may be redundant. Therefore, whenever we want to eliminate redundant data occurrences in more possible data, then we must expect the normalization effort to grow. Redundancy in the smallest possible world r_1, for example, could be caused by FDs that hold with any certainty degree, except the bottom degree, while redundancy in the largest possible world r_4 can only be caused by FDs with the top certainty degree. In our running example, the second occurrence of *Gill* is redundant with respect to the FD *Project* $\rightarrow$ *Manager* that holds with degree β_4. Indeed, any replacement of this data occurrence by a different value will result in a certainty degree of *Project* $\rightarrow$ *Manager* that is smaller than β_4, with which it is expected to hold. Similarly, the occurrence of *Lava* in the last tuple is redundant with respect to the FD *Manager*, *Time* $\rightarrow$ *Room* that holds with degree β_1. It is natural to think that less possible data is more likely to be updated than more possible data, since less possible data have indeed a lower possibility to be true. So positively speaking, redundancy in less possible data can be eliminated with less normalization effort. In fact, to eliminate all redundancy in tuples with possibility degree α_4 it is sufficient to

Figure 2: Worlds of Possibilistic Relation in Table 1



normalize with respect to FDs that hold with certainty degree β_1. However, to eliminate all redundancy in tuples with possibility degree α_1 it is required to normalize with respect to FDs that hold with any certainty degree $\beta_1, \beta_2, \beta_3$ and β_4. Therefore, the impact we expect uncertainty to have on schema design is that the more frequent updates are likely to be processed efficiently with less normalization effort. This also means that more queries can be processed efficiently, as less joins are required. Therefore, uncertainty results in a greater variety of normalized schema designs from which database designers can choose the best match for the target workload in terms of updates and queries. In other words, uncertainty can be understood as an effective mechanism to better control the classical trade-off between update and query efficiency.

3 Related Work

Probabilistic databases have received much interest in recent years [34] due to the need to deal with uncertain data. Constraints are a key challenge for probabilistic databases: "When the data is uncertain, constraints can be used to increase the quality of the data, and hence they are an important tool in managing data with uncertainties" [15]. Suciu et al. emphasize that "the main use of probabilities is to record the degree of uncertainty in the data and to rank the outputs to a query; in some applications, the exact output probabilities matter less to the user than the ranking of the outputs" [34]. This strongly suggests that a qualitative approach to uncertainty, such as possibility theory [18], can avoid the high computational complexities in obtaining probabilities and processing them, while guaranteeing the same qualitative outcome.

Research on probabilistic databases has naturally focused on query processing, e.g. [6, 25, 26], and resulted in several probabilistic DBMS, including MYSTIQ [11, 14], MayBMS [1, 24], and Trio [7, 32]. Common to many is the desire to extend trusted relational technology to handle uncertainty. This desire is also inherent in our approach: We show how to exploit the well-founded relational normalization framework to design database schemata for uncertain data.

An orthogonal article discusses the origins of our definition of possibilistic FDs in possibility theory and contains an up-to-date review of functional dependencies from various frameworks for uncertainty [30]. That article proves the decomposition theorem, which we exploit in this article for normalization purposes. It also contains an extension of Fagin's well-known equivalence between the implication problem of FDs and that of Horn clauses in Boolean propositional logic [20], to an equivalence between the implication problem of possibilistic FDs and that of Horn clauses in Boolean possibilistic logic. Design theory, normal forms, normalization, redundancy, implementation and experiments are not studied elsewhere, but contributions of the current article[1].

The only paper that considers schema design for uncertain databases is [33]. Das Sarma, Ullman, and Widom develop an "FD theory for data models whose basic construct for uncertainty is *alternatives*" [33]. Their work is therefore fundamentally different from our approach.

Finally, we acknowledge the findings for relational schema design, which enabled us to develop our approach towards schema design for uncertain data. FDs are already mentioned in the seminal paper by Codd [12] and constitute one of the most prolific concepts in databases. Armstrong axiomatized FDs [3], and linear-time algorithms to decide their associated implication problem are known [16]. These results provide the foundation for relational schema design, including 3NF [4, 8, 10], BCNF [13, 20, 31] and their semantic justification in terms of dependency-preservation and elimination of data redundancy [2, 27, 36]. Deciding whether a given schema is in 3NF is NP-complete, while the same decision problem for BCNF is in P-time, but deciding whether a projection of a given schema is in BCNF is coNP-complete [4]. Tsou and Fischer have established an algorithm to compute a lossless join decomposition into BCNF that requires only polynomial time [35]. Achievements of relational database design are discussed by different authors, e.g. [5, 9, 19]. Our experimental section will reveal original insight into the trade-off between classical 3NF synthesis and BCNF decomposition approaches.

4 Uncertain Databases

This section introduces our possibilistic data model for tuple uncertainty, on which we will build our schema design framework in subsequent sections.

A relation schema, usually denoted by R, is a finite non-empty set of *attributes*. Each attribute $A \in R$ has a *domain* $dom(A)$ of values. A *tuple* t over R is an element of the Cartesian product $\prod_{A \in R} dom(A)$ of the attributes' domains. For $X \subseteq R$ we denote by $t(X)$ the *projection* of t on X. A *relation* over R is a finite set r of tuples over R. As

[1]The research has also been filed as a patent application[29]

a running example we use the relation schema MEETING with attributes *Project*, *Time*, *Manager*, *Room* from Section 2.

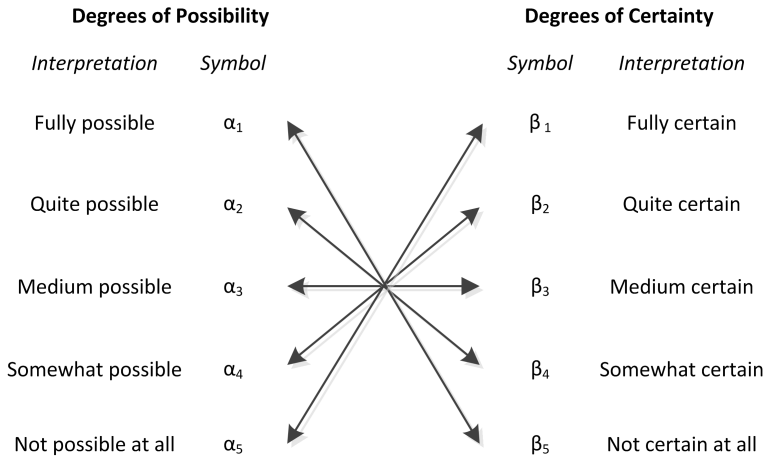
Tuples either belong or do not belong to a relation, there is no room for uncertainty. For example, we cannot express that we have less confidence that Bob attends a meeting of project Lion on Wednesday at 11am in room Cyan than we have confidence that Gill attends a meeting of project Lion on Wednesday at 11am in room Cyan. Effective database support for new applications, including data exchange, extraction, and integration, requires us to accommodate uncertainty in databases, but also to make the most out of this additional information.

We define uncertain relations as relations where each tuple is associated with some confidence. The confidence of a tuple expresses up to which degree of possibility a tuple occurs in a relation. Formally, we model the confidence as a *scale of possibility*, i.e., a finite strict linear order $\mathcal{S} = (S, <)$ with $k+1$ elements where k is some positive integer, which we denote by $\alpha_1 > \cdots > \alpha_k > \alpha_{k+1}$, and whose elements $\alpha_i \in S$ we call *possibility degrees*. The top possibility degree α_1 is reserved for tuples that are 'fully possible' to occur in a relation, while the bottom possibility degree α_{k+1} is reserved for tuples that are 'not possible at all', that is 'impossible', to occur in a relation. Humans like to use simple scales in everyday life, for instance to communicate, compare, or rank. Simple usually means to classify items qualitatively, rather than quantitatively by putting a precise value on it. Note that classical relations use a scale with two elements, i.e. where $k = 1$. As a running example, we will use a scale of five possibility degrees, ranging from *fully possible* (α_1), to *rather possible* (α_2), to *medium possible* (α_3), to *somewhat possible* (α_4), and finally to *not possible at all* (α_5).

Formally speaking, a *possibilistic relation schema* $(R, \mathcal{S})$ consists of a relation schema R and a possibility scale $\mathcal{S}$. A *possibilistic relation* over $(R, \mathcal{S})$ consists of a relation r over R, together with a function $Poss_r$ that maps each tuple $t \in r$ to a possibility degree $Poss_r(t)$ in the possibility scale $\mathcal{S}$. For example, Table 1 shows a possibilistic relation over $(\text{MEETING}, \mathcal{S} = \{\alpha_1, \ldots, \alpha_5\})$.

Possibilistic relations enjoys a well-founded semantics in terms of possible worlds. In fact, a possibilistic relation gives rise to a possibility distribution over possible worlds of relations. For $i = 1, \ldots, k$ let r_i denote the relation that consists of all tuples in r that have a possibility degree of at least α_i, i.e., $r_i = \{t \in r \mid Poss_r(t) \geq \alpha_i\}$. The linear order of the possibility degrees results in a (reversed) linear order of possible worlds of relations. Indeed, we have $r_1 \subseteq r_2 \subseteq \cdots \subseteq r_k$. The possibility distribution π_r for this linear chain of possible worlds is defined by $\pi_r(r_i) = \alpha_i$. Note that r_{k+1} is not considered to be a possible world, since its possibility $\pi(r_{k+1}) = \alpha_{k+1}$ means 'not possible at all'. Vice versa, the possibility $Poss_r(t)$ of a tuple $t \in r$ is the possibility of the smallest possible world in which t occurs, i.e., the maximum possibility $\max\{\alpha_i \mid t \in r_i\}$ of a world to which t belongs. If $t \notin r_k$, then $Poss_r(t) = \alpha_{k+1}$. The top possibility degree α_1 takes on a distinguished role: every tuple that is 'fully possible' occurs in every possible world - and is therefore - 'fully certain'. This formally confirms our intuition that uncertain relations subsume relations (of fully certain tuples) as a special case. Figure 2 shows the possible worlds $r_1 \subsetneq r_2 \subsetneq r_3 \subsetneq r_4$ that result from the possibilistic relation of Table 1.

Figure 3: Example for Possibility/Certainty Degrees



5 Possibilistic FDs

Based on our framework from the previous section we now introduce possibilistic FDs as classical FDs together with a degree of certainty. In the same way classical FDs are fundamental to classical database design, PFDs will play a fundamental role to schema design for uncertain data.

Recall that a functional dependency $X \to Y$ over relation schema R is satisfied by a relation r over R if and only if every pair of tuples in r that have matching values on all the attributes in X have also matching values on all the attributes in Y. For example, the FD *Manager, Room* $\to$ *Time* is not satisfied by any relation $r_1, \ldots, r_4$. The FD *Project* $\to$ *Manager* is satisfied by relation r_1, but not by relation r_2 and therefore not by relations r_3 and r_4. The FD *Project, Time* $\to$ *Manager* is satisfied by relations r_1 and r_2, but not by relation r_3 and therefore not by relation r_4. The FD *Time, Room* $\to$ *Project* is satisfied by relations r_1, r_2, and r_3, but not by r_4. Finally, the FD *Manager, Time* $\to$ *Room* is satisfied by all relations $r_1, \ldots, r_4$.

Naturally, the possibility degrees of tuples that define an uncertain relation also define degrees of certainty with which FDs hold in the uncertain relation. Intuitively, since *Manager, Time* $\to$ *Room* is satisfied in every possible world, it is fully certain to hold in r. As *Time, Room* $\to$ *Project* is only violated in a somewhat possible world r_4, it is quite certain to hold in r. Since *Project, Time* $\to$ *Manager* is violated in a medium possible world r_3 (but not in a quite possible world), it is medium certain to hold in r. As *Project* $\to$ *Manager* is violated in a quite possible world r_2 (but not in a fully possible world), it is somewhat certain to hold in r. Finally, as *Manager, Room* $\to$ *Time* is violated in the fully possible world r_1, it is not certain at all to hold in r.

In summary, the certainty with which an FD holds in an uncertain relation corresponds to the possibility of the smallest possible world in which the FD is violated. Therefore, similar to a scale $\mathcal{S}$ of possibility degrees for tuples we use a scale $\mathcal{S}^T$ of certainty degrees for FDs. We commonly use subscripted versions of the Greek letter β

to denote certainty degrees associated with FDs. Formally, the correspondence between possibility degrees in $\mathcal{S}$ and the certainty degrees in $\mathcal{S}^T$ can be defined by the mapping $\alpha_i \mapsto \beta_{k+2-i}$ for $i = 1, \ldots, k+1$. Figure 3 extends the scale of possibility degrees of our running example by a scale of certainty degrees, according to the mapping. Formally then, the certainty $C_r(X \to Y)$ with which the FD $X \to Y$ holds in the uncertain relation r is the top degree β_1 whenever $X \to Y$ is satisfied by r_k, or otherwise the minimum amongst the certainty degrees β_{k+2-i} that correspond to possible worlds r_i in which $X \to Y$ is violated, i.e., $C_r(X \to Y)$

$$= \begin{cases} \beta_1 & , \textit{ if } r_k \textit{ satisfies } X \to Y \\ \min\{\beta_{k+2-i} \mid \not\models_{r_i} X \to Y\} & , \textit{ otherwise} \end{cases} .$$

In other words: an FD is either fully certain as it holds on the largest possible world, or its degree corresponds to the smallest possible world in which it is violated. We can now define the syntax and semantics of PFDs.

Definition 1 *A* possibilistic functional dependency *over a possibilistic relation schema* $(R, \mathcal{S})$ *is an expression* $(X \to Y, \beta)$ *where* $X, Y \subseteq R$ *and* $\beta \in S^T$. *A possibilistic relation* $(r, Poss_r)$ *over* $(R, \mathcal{S})$ *is said to* satisfy *the possibilistic functional dependency* $(X \to Y, \beta)$ *if and only if* $C_r(X \to Y) \geq \beta$.

For examples, the possibilistic relation $(r, Poss_r)$ of our running example satisfies the following possibilistic functional dependencies:

- $(Manager, Time \to Room, \beta_1)$
- $(Room, Time \to Project, \beta_2)$
- $(Project, Time \to Manager, \beta_3)$
- $(Project \to Manager, \beta_4)$
- $(Manager, Room \to Time, \beta_5)$.

For another example, it violates the PFD

$$(Project, Time \to Manager, \beta_2)$$

since $C_r(Project, Time \to Manager) = \beta_3 < \beta_2$.

PFDs form a class of integrity constraints tailored to uncertain data. Indeed, a PFD $(X \to Y, \beta_i)$ separates semantically meaningful from meaningless possibilistic relations by allowing violations of the FD $X \to Y$ only by tuples with a possibility degree α_j where $j \leq k + 1 - i$. For $i = 1, \ldots, k$, the certainty degree β_i of $(X \to Y, \beta_i)$ means that the classical FD $X \to Y$ must hold in the possible world r_{k+1-i}. This constitutes a conveniently flexible mechanism to enforce the targeted level of integrity effectively.

As motivation for future sections it is worth pointing out the impact of PFDs on data redundancy. As PFDs with different certainty degree do not apply to the same worlds, they can also cause different degrees of data redundancy. In fact, data redundancy can only occur in worlds to which a given PFD applies. This observation will save us normalization effort when eliminating data redundancy from less possible worlds, on which only more certain FDs hold.

6 Scaled Design Theory

Classical normalization algorithms such as BCNF decomposition and 3NF synthesis are founded on the theory of classical FDs. Consequently, we establish a design theory of PFDs. Starting from a strong correspondence between the implication problem of PFDs and that of classical FDs, we identify an axiomatization for PFDs that is reminiscent of Armstrong's axioms for FDs, and show that the associated implication problem can be decided in time linear in the input. As applications we discuss an algorithm that infers the highest certainty degree with which a given FD is implied by a given set of PFDs, as well as the computation of non-redundant and canonical covers.

6.1 The Magic of β-Cuts

First, we establish a strong correspondence between the implication of PFDs and FDs. Let $\Sigma \cup \{\varphi\}$ denote a set of PFDs over a possibilistic relation schema $(R, \mathcal{S})$. We say that Σ *implies* φ, denoted by $\Sigma \models \varphi$, if every possibilistic relation $(r, Poss_r)$ over $(R, \mathcal{S})$ that satisfies every PFD in Σ also satisfies φ.

Example 1 *Let Σ consist of the following four PFDs $(Manager, Time \rightarrow Room, \beta_1)$, $(Room, Time \rightarrow Project, \beta_2)$, $(Project, Time \rightarrow Manager, \beta_3)$, $(Project \rightarrow Manager, \beta_4)$ over $(\textsc{Meeting}, \{\alpha_1, \ldots, \alpha_5\})$. Further, let φ denote the PFD $(Room, Time \rightarrow Manager, \beta_2)$. Then Σ does not imply φ as the following possibilistic relation witnesses:*

Project	*Time*	*Manager*	*Room*	*Poss. degree*
Lion	*Wed, 3pm*	*Gill*	*Cyan*	α_1
Lion	*Wed, 3pm*	*Robert*	*Cyan*	α_3

.

For a set Σ of PFDs on some possibilistic relation schema $(R, \mathcal{S})$ and certainty degree $\beta \in \mathcal{S}^T$ where $\beta > \beta_{k+1}$, let

$$\Sigma_\beta = \{X \rightarrow Y \mid (X \rightarrow Y, \beta') \in \Sigma \text{ and } \beta' \geq \beta\}$$

be the *β-cut* of Σ. The major strength of our framework is engraved in the following result.

Theorem 1 *Let $\Sigma \cup \{(X \rightarrow Y, \beta)\}$ be a set of PFDs over a possibilistic relation schema $(R, \mathcal{S})$ where $\beta > \beta_{k+1}$. Then $\Sigma \models (X \rightarrow Y, \beta)$ if and only if $\Sigma_\beta \models X \rightarrow Y$.*

Proof Suppose $(r, Poss_r)$ is some possibilistic relation over $(R, \mathcal{S})$ that satisfies Σ, but violates $(X \rightarrow Y, \beta)$. In particular, $C_r(X \rightarrow Y) < \beta$ implies that there is some relation r_i that violates $X \rightarrow Y$ and where

$$\beta_{k+2-i} < \beta. \tag{1}$$

Let $U \rightarrow V \in \Sigma_\beta$, where $(U \rightarrow V, \beta') \in \Sigma$. Since r satisfies $(U \rightarrow V, \beta') \in \Sigma$ we have

$$C_r(U \rightarrow V) \geq \beta' \geq \beta. \tag{2}$$

If r_i violated $U \to V$, then

$$\begin{array}{rcll} \beta & > & \beta_{k+2-i} & \text{by (1)} \\ & \geq & C_r(U \to V) & \text{by Definition of } C_r \\ & \geq & \beta & \text{by (2)} \end{array}$$

a contradiction. Hence, the relation r_i satisfies Σ_β and violates $X \to Y$.

Let r' denote some relation that satisfies Σ_β and violates $X \to Y$. W.l.o.G. we assume that $r' = \{t, t'\}$ consists of only two tuples. If that is not the case, then it is well-known that there is a sub-relation of r' with two tuples that satisfies Σ_β and violates $X \to Y$. Let r be the possibilistic relation over $(R, \mathcal{S})$ that consists of the relation r' and where $Poss_{r'}(t) = \alpha_1$ and $Poss_{r'}(t') = \alpha_i$, such that $\beta_{k+1-i} = \beta$. Then r violates $(X \to Y, \beta)$ since $C_r(X \to Y) = \beta_{k+2-i}$, as $r_i = r'$ is the smallest relation that violates $X \to Y$, and $\beta_{k+2-i} < \beta_{k+1-i} = \beta$. For $(U \to V, \beta') \in \Sigma$ we distinguish two cases. If r_i satisfies $U \to V$, then $C_r(U \to V) = \beta_1 \geq \beta$. If r_i violates $U \to V$, then $U \to V \notin \Sigma_\beta$, i.e., $\beta' < \beta = \beta_{k+1-i}$. Therefore, $\beta' \leq \beta_{k+2-i} = C_r(U \to V)$ as $r_i = r'$ is the smallest relation that violates $U \to V$. We conclude that $C_r(U \to V) \geq \beta'$. Consequently, $(r, Poss_r)$ is a possibilistic relation that satisfies Σ and violates $(X \to Y, \beta)$. ∎

Example 2 *Let Σ and φ be as in Example 1, in particular Σ does not imply φ. Theorem 1 says that Σ_{β_2} does not imply Room, Time → Manager. Indeed, the relation*

Project	*Time*	*Manager*	*Room*
Lion	*Wed, 3pm*	*Gill*	*Cyan*
Lion	*Wed, 3pm*	*Robert*	*Cyan*

satisfies the two classical FDs Manager, Time → Room and Room, Time → Project that form Σ_{β_2}, and violates the FD Room, Time → Manager. Note how this relation is the possible world r_3 of the possibilistic relation from Example 1.

6.2 Scaled Armstrong Axioms

The *semantic closure* $\Sigma^* = \{\varphi \mid \Sigma \models \varphi\}$ contains all dependencies implied by the given set Σ. In order to determine the semantic closure, one can utilize a syntactic approach by applying *inference rules* of the form

$$\frac{\text{premise}}{\text{conclusion}} \text{ condition},$$

where rules without a premise are called *axioms*. For a set $\mathfrak{R}$ of inference rules let $\Sigma \vdash_\mathfrak{R} \varphi$ denote the *inference* of φ from Σ by $\mathfrak{R}$. That is, there is some sequence $\sigma_1, \ldots, \sigma_n$ such that $\sigma_n = \varphi$ and every σ_i is an element of Σ or is the conclusion that results from an application of an inference rule in $\mathfrak{R}$ to some premises in $\{\sigma_1, \ldots, \sigma_{i-1}\}$. Let $\Sigma^+_\mathfrak{R} = \{\varphi \mid \Sigma \vdash_\mathfrak{R} \varphi\}$ denote the *syntactic closure* of Σ under inferences by $\mathfrak{R}$. $\mathfrak{R}$ is *sound* (*complete*) if for every possibilistic relation schema $(R, \mathcal{S})$ and for every set Σ we have $\Sigma^+_\mathfrak{R} \subseteq \Sigma^*$ ($\Sigma^* \subseteq \Sigma^+_\mathfrak{R}$). The (finite) set $\mathfrak{R}$ is said to be a (finite) *axiomatization* if $\mathfrak{R}$ is both sound and complete.

$$\frac{}{XY \to Y}\ (\text{reflexivity, } \mathcal{R}') \qquad \frac{X \to Y}{X \to XY}\ (\text{extension, } \mathcal{E}')$$

$$\frac{X \to Y \quad Y \to Z}{X \to Z}\ (\text{transitivity, } \mathcal{T}')$$

Table 2: Armstrong axioms $\mathfrak{A} = \{\mathcal{R}', \mathcal{E}', \mathcal{T}'\}$ of FDs

$$\frac{}{(XY \to Y, \beta)}\ (\text{reflexivity, } \mathcal{R}) \qquad \frac{(X \to Y, \beta)}{(X \to XY, \beta)}\ (\text{extension, } \mathcal{E})$$

$$\frac{(X \to Y, \beta) \quad (Y \to Z, \beta)}{(X \to Z, \beta)}\ (\text{transitivity, } \mathcal{T})$$

$$\frac{}{(X \to Y, \beta_{k+1})}\ (\text{bottom, } \mathcal{B}) \qquad \frac{(X \to Y, \beta)}{(X \to Y, \beta')}\ \beta' < \beta\ (\text{weakening, } \mathcal{W})$$

Table 3: Axiomatization $\mathfrak{P} = \{\mathcal{R}, \mathcal{E}, \mathcal{T}, \mathcal{B}, \mathcal{W}\}$ of PFDs

One of the early and well-known results from relational database theory are Armstrong's axioms which Armstrong showed to be sound and complete for the implication of FDs [3]. The axioms are shown in Table 2.

Possibilistic FDs enjoy a similar axiomatization, summarized in Table 3, as we show now. In these rules we implicitly assume that all attribute sets X, Y, Z are subsets of the arbitrarily given relation schema R, and the certainty degrees β and β' are elements of the arbitrarily given certainty scale $\mathcal{S}^T$. In particular, β_{k+1} denotes the bottom certainty degree. The weakening axiom $\mathcal{W}$ in Table 3 strictly conforms with the rules of possibilistic logic [17]. Note that system $\mathfrak{P}$ is equivalent to Armstrong's axioms for FDs, if the scale $\mathcal{S}^T$ consists of just two certainty degrees.

Theorem 2 *The set $\mathfrak{P} = \{\mathcal{R}, \mathcal{E}, \mathcal{T}, \mathcal{B}, \mathcal{W}\}$ of inference rules forms a finite axiomatization for the implication of possibilistic functional dependencies.*

Proof The proofs of soundness are straightforward, keeping in mind the soundness of Armstrong's axioms for FDs and Theorem 1. For the completeness proof, we take full advantage of Theorem 1 and the fact that the Armstrong axioms are sound and complete for the implication of FDs. Let $(R, \mathcal{S})$ be an arbitrary possibilistisc relation schema with $|\mathcal{S}| = k + 1$, and $\Sigma \cup \{(X \to Y, \beta)\}$ an arbitrary set of PFDs over the schema, such that $\Sigma \models (X \to Y, \beta)$ holds. We need to show that $\Sigma \vdash_{\mathfrak{P}} (X \to Y, \beta)$ holds, too.

We distinguish two cases. If $\beta = \beta_{k+1}$, then $\Sigma \models (X \to Y, \beta)$ means that $\Sigma \vdash_{\mathfrak{P}} (X \to Y, \beta)$ holds by a single application of the bottom rule $\mathcal{B}$. For the remainder of the proof we therefore assume that $\beta < \beta_{k+1}$. From $\Sigma \models (X \to Y, \beta)$ we conclude $\Sigma_\beta \models X \to Y$ by Theorem 1. Since the Armstrong axioms $\mathfrak{A}$ are complete for the implication of FDs, we conclude that $\Sigma_\beta \vdash_{\mathfrak{A}} X \to Y$ holds. Now, for the FD set Σ_β we define $\Sigma_\beta^\beta = \{(X \to Y, \beta) \mid X \to Y \in \Sigma_\beta\}$. Therefore, the inference of $X \to Y$ from Σ_β using the Armstrong axioms can be easily turned into an inference of $(X \to Y, \beta)$ from Σ_β^β by $\mathfrak{P}$, simply by adding the certainty degree β to each FD that occurs in the inference. Therefore, whenever an Armstrong axiom $\mathcal{R}'$, $\mathcal{E}'$ or $\mathcal{T}'$ is applied, one can now apply the corresponding rule $\mathcal{R}$, $\mathcal{E}$, or $\mathcal{T}$, respectively. It follows that $\Sigma_\beta^\beta \vdash_{\mathfrak{P}} (X \to Y, \beta)$ holds. Finally, the definition of Σ_β^β ensures that every PFD in Σ_β^β can be inferred from a PFD in Σ by a single application of the weakening rule $\mathcal{W}$. Hence, $\Sigma_\beta^\beta \vdash_{\mathfrak{P}} (X \to Y, \beta)$ means, in particular, $\Sigma \vdash_{\mathfrak{P}} (X \to Y, \beta)$. This completes the proof. ∎

Alternatively, we could have restricted the definition of PFDs to certainty degrees that are different from the bottom degree. In that case, the bottom axiom $\mathfrak{B}$ is not required. Here, we have decided to include the bottom degree to equip every FD with some certainty degree.

Example 3 *Let Σ and φ be as in Example 1, and $\varphi' = (Room, Time \to Manager, \beta_3)$. We show an inference of φ' from Σ by $\mathfrak{P}$. Attribute names are abbreviated by their first letters.*

$$\dfrac{\dfrac{\dfrac{(RT \to P, \beta_2)}{\mathcal{W} : (RT \to P, \beta_3)} \quad \dfrac{}{\mathcal{R} : (RTP \to PT, \beta_3)}}{\mathcal{T} : (RT \to PT, \beta_3)} \quad (PT \to M, \beta_3)}{\mathcal{T} : (RT \to M, \beta_3)}$$

Of course, φ cannot be inferred from Σ by $\mathfrak{P}$, as Example 1 and the soundness of $\mathfrak{P}$ show.

6.3 Scaled Decision Algorithm

In practice it is often not necessary to compute the full set Σ^* of integrity constraints implied by a given set Σ. In contrast, rather frequent is the problem whether for a given set $\Sigma \cup \{\varphi\}$ of integrity constraints, Σ implies φ.

PROBLEM:	IMPLICATION
INPUT:	Relation schema R, Scale $\mathcal{S}$ with $k+1$ possibility degrees, Set $\Sigma \cup \{\varphi\}$ of PFDs over $(R, \mathcal{S})$
OUTPUT:	Yes, if $\Sigma \models \varphi$, and No, otherwise

Of course, one could compute Σ^* and check whether $\varphi \in \Sigma^*$, but this is hardly efficient nor does it make effective use of the input φ. In the case of PFDs, we can again exploit

Theorem 1 to derive an algorithm that decides the implication problem in time linear in the input. Given a PFD set $\Sigma \cup \{(X \to Y, \beta)\}$ we return *true* if $\beta = \beta_{k+1}$, otherwise it is sufficient to check whether $\Sigma_\beta \models X \to Y$. The latter test can be done in linear time by computing the attribute set closure $X^+_{\Sigma_\beta} = \{A \in R \mid \Sigma \vdash_{\mathfrak{A}} X \to A\}$ of X with respect to Σ_β. Algorithm 1 computes the attribute set closure in time $\mathcal{O}(||\Sigma_\beta|| + |X|)$. We use $||\Sigma||$ to denote the total number of attribute occurrences in Σ, and $|X|$ to denote the cardinality of X, independently of whether Σ is a set of PFDs or FDs.

Algorithm 1 Closure Computation

```
1: procedure CLOSURE(R, X, Σ_β)
2:     Closure ← X;
3:     FDList ← List of X → Y ∈ Σ_β;
4:     repeat
5:         OldClosure ← Closure;
6:         Remove Closure from LHS of FDs in FDList;
7:         for all ∅ → Y in FDList do
8:             Closure ← Closure ∪ Y;
9:             FDList ← FDList − {∅ → Y};
10:        end for
11:    until Closure=OldClosure or FDList=[ ]
12:    return(Closure);
13: end procedure
```

Therefore, we obtain the following algorithmic characterization of the implication problem for PFDs.

Theorem 3 *The implication problem $\Sigma \models \varphi$ of possibilistic FDs can be decided in time $\mathcal{O}(||\Sigma \cup \{\varphi\}||)$.* ∎

Example 4 *Let Σ and φ be as in Example 1, and $\varphi' = (Room, Time \to Manager, \beta_3)$. It follows that φ is not implied by Σ as* Closure(Meeting,*RT*,Σ_{β_2})=*RTP. Further, φ' is implied by Σ as* Closure(Meeting,*RT*,Σ_{β_3})=*RTPM.*

6.4 Certainty inference

From the perspective of uncertainty, the following problem is interesting: Given a set Σ of PFDs over schema $(R, \mathcal{S})$ and an FD $X \to Y$ over R, what is maximum certainty degree $\beta \in \mathcal{S}^T$ such that $\Sigma \models (X \to Y, \beta)$ holds?

PROBLEM: INFERENCE	
INPUT:	Relation schema R, Scale $\mathcal{S}$ with $k+1$ possibility degrees, Set Σ of PFDs over $(R, \mathcal{S})$, FD $X \to Y$ over R
OUTPUT:	$\max\{\beta \in \mathcal{S}^T \mid \Sigma \models (X \to Y, \beta)\}$

A simple approach to solve the inference problem is to compute attribute set closures of X incrementally by adding FDs from Σ_{β_1}, $\Sigma_{\beta_2} - \Sigma_{\beta_1}$, ... , $\Sigma_{\beta_{i+1}} - \Sigma_{\beta_i}$, until $Y \subseteq X^+_{\Sigma_{\beta_i}}$ for the smallest i, or $Y \not\subseteq X^+_{\Sigma_{\beta_k}}$. In the first case, we return β_i, in the latter case β_{k+1}. The algorithm should only be applied to indices i for which a PFD $(U \to V, \beta_i) \in \Sigma$ exists.

Theorem 4 *The inference problem for PFDs with input $\Sigma \cup \{X \to Y\}$ can be computed in time $\mathcal{O}(||\Sigma|| + |X \cup Y|)$.*

Example 5 *Let Σ be as in Example 1. Then the maximum certainty degree β for which the FD $(RT \to M, \beta)$ is implied by Σ is β_3.*

6.5 Computation of Covers

From the perspective of integrity constraints, the notion of a cover embodies a minimal amount of resources necessary to enforce the business rules that have been identified to be semantically meaningful for an application domain. Standard notions of covers apply to PFDs as well. A set Θ of PFDs is a *cover* of a set Σ of PFDs, if $\Theta^+_{\mathfrak{P}} = \Sigma^+_{\mathfrak{P}}$. A cover Θ is said to be *non-redundant*, if for every PFD $\sigma \in \Theta$, $\sigma \notin (\Theta - \{\sigma\})^+_{\mathfrak{P}}$. Θ is said to be L-reduced, if for every $(X \to Y, \beta) \in \Sigma$ and for all $Z \subset X$, $(Z \to Y, \beta) \notin \Theta^+_{\mathfrak{P}}$. A cover Θ is said to be *canonical*, if Θ is non-redundant, L-reduced, and there are no two different $(X \to Y, \beta), (X \to Z, \beta) \in \Theta$. Exploiting Algorithm 1, the following two algorithms compute non-redundant and canonical covers, respectively. Without loss of generality, we assume that the input Σ to both algorithms consists exclusively of PFDs $(X \to Y, \beta)$ where $Y = A$ denotes a singleton attribute of the underlying relation schema.

Algorithm 2 Non-redundant Cover

```
1: procedure NR-Cover(R, Σ)
2:     for all σ = (X → A, β) ∈ Σ do
3:         if A ∈ Closure(R, X, (Σ − {σ})_β) then
4:             Σ ← Σ − {σ};
5:         end if
6:     end for
7:     return(Σ);
8: end procedure
```

While non-redundant covers eliminate redundant PFDs, canonical covers also eliminate redundant attributes. In the following algorithm it is important to eliminate redundant attributes first, before eliminating redundant PFDs. This sequence of steps is already fixed in the classical case.

Theorem 5 *On input (R, Σ), Algorithm 2 computes a non-redundant cover of Σ in time $\mathcal{O}(||\Sigma||^2)$, and Algorithm 3 computes a canonical cover of Σ in time $\mathcal{O}(|R| \times ||\Sigma||^2)$.*

Algorithm 3 Canonical Cover

```
1: procedure CAN-COVER(R, Σ)
2:     for all σ = (X → A, β) ∈ Σ do
3:         Z ← X;
4:         for all B ∈ Z do
5:             if A ∈ CLOSURE(R, Z − {B}, Σ_β) then
6:                 Z ← Z − {B};
7:             end if
8:         end for
9:         Σ ← (Σ − {σ}) ∪ {(Z → A, β)};
10:    end for
11:    Σ ← NR-COVER(R, Σ);
12:    while (X → Y, β), (X → Z, β) ∈ Σ do
13:        Σ ← Σ − {(X → Y, β), (X → Z, β)};
14:        Σ ← Σ ∪ {(X → YZ, β)};
15:    end while
16:    return(Σ);
17: end procedure
```

Example 6 *Consider the relation schema* MEETING *with the following set* Σ *of PFDs:* $\sigma_1 = (Manager, Time \rightarrow Room, \beta_1)$, $\sigma_2 = (Manager, Time, Room \rightarrow Project, \beta_1)$, *and* $\sigma_3 = (Manager, Time \rightarrow Project, \beta_2)$. *A non-redundant cover of* Σ *is the set* $\Theta = \{\sigma_1, \sigma_2\}$, *while a canonical cover of* Σ *is the set*

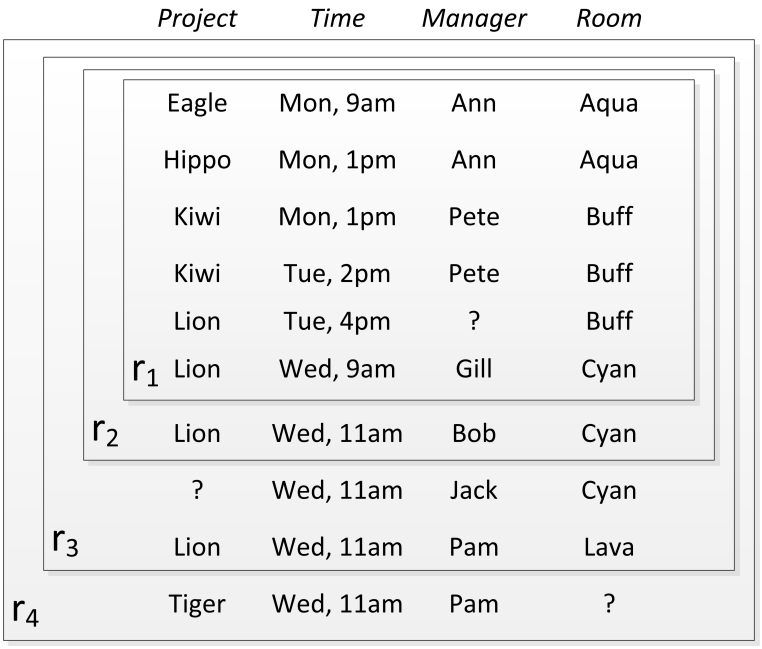
$$\Theta' = \{(Manager, Time \rightarrow Room, Project, \beta_1)\}.$$

7 Scaled Normal Forms

In classical relational databases, Boyce-Codd normal form (BCNF) syntactically characterizes relation schemata that are guaranteed to be free of data redundancy in any relations over the schema, in terms of functional dependencies [36]. Third normal form (3NF) syntactically characterizes relation schemata that are guaranteed to have the least amount of data redundancy in their relations amongst all schemata on which all functional dependencies can be enforced locally [27]. In relational databases, all tuples are fully certain and all semantically meaningful functional dependencies apply to all data. Thus, data redundancy is treated uniformly for all data, and the elimination of all data redundancy requires normalization with respect to all functional dependencies.

In possibilistic relations, different data can occur with different degrees of possibility, and different possibilistic FDs may apply to different data. Therefore, data redundancy may occur with different degrees of possibility. Intuitively speaking, the smaller the degree of possibility for which data redundancy is to be eliminated, the smaller the normalization effort will be. In this section we will introduce notions of data redundancy that are tailored to the possibility degree of tuples in which they occur. This results in a whole range of semantic normal forms by which data redundancy of growing degrees of

Figure 4: Illustration of Data Redundancy



possibility are eliminated. We then characterize each of these semantic normal forms by a corresponding syntactic normal form, and establish strong correspondences with BCNF in relational databases. Additionally, we tailor 3NF to possibilistic databases.

7.1 Scaled Data Redundancy

We motivate our scaled notions of data redundancy on our running example. Suppose we are given the possibilistic relation schema MEETING with the set Σ of PFDs from Example 1. Suppose we know that the possibilistic relation $(r, Poss_r)$ in Figure 4 satisfies Σ, but we cannot see all of its data value occurrences. Since r satisfies Σ, the question mark in r_4 cannot stand for any data value different from *Lava* - as any other data value would violate the PFD $(Time, Manager \rightarrow Room, \beta_1)$. Therefore, the data value occurrence of *Lava* in the last tuple is redundant in the sense that it can be inferred from the other data value occurrences and the given PFDs. More specifically, the data value is α_4-redundant as it occurs in the possible world r_4 and can therefore only be caused by PFDs that hold with certainty degree β_1. The question mark in r_3 can only be replaced by the redundant data value *Lion*, caused by the PFD $(Time, Room \rightarrow Project, \beta_2)$. The data value is α_3-redundant, and could be caused by any PFD in Σ with certainty degree β_1 or β_2. Finally, the question mark in r_1 stands for the redundant data value occurrence of *Gill*, caused by the PFD $(Project \rightarrow Manager, \beta_4)$. This occurrence is α_1-redundant, and could be caused by any PFD in Σ - all of which apply to tuples with possibility degree α_1.

We will now introduce different notions of data redundancy that are tailored towards

the uncertainty of tuples. For this, we exploit the classical proposal by Vincent [36]. Let R denote a relation schema, A an attribute of R, t a tuple over R, and Σ a set of FDs over R. A *replacement* of $t(A)$ is a tuple $\bar{t}$ over R that satisfies the following conditions: i) for all $\bar{A} \in R - \{A\}$ we have $\bar{t}(\bar{A}) = t(\bar{A})$, and ii) $\bar{t}(A) \neq t(A)$. For a relation r over R that satisfies Σ and $t \in r$, the data value occurrence $t(A)$ in r is *redundant* with respect to Σ if and only if for every replacement $\bar{t}$ of $t(A)$, $\bar{r} := (r - \{t\}) \cup \{\bar{t}\}$ violates some FD in Σ. A relation schema R is in Redundancy Free normal form (RFNF) with respect to a set Σ of FDs if and only if there are no relation r over R that satisfies Σ, tuple $t \in r$, and attribute $A \in R$ such that the data value occurrence $t(A)$ is redundant with respect to Σ [36].

Definition 2 *Let $(R, \mathcal{S})$ denote a possibilistic relation schema, Σ a set of PFDs over $(R, \mathcal{S})$, $A \in R$ an attribute, $(r, Poss_r)$ a possibilistic relation over $(R, \mathcal{S})$ that satisfies Σ, and t a tuple in r_i. The data value occurrence $t(A)$ is* α_i-redundant *if and only if $t(A)$ is redundant with respect to $\Sigma_{\alpha_i} = \{X \to Y \mid (X \to Y, \beta) \in \Sigma \text{ and } \beta \geq \beta_{k+1-i}\}$.* ∎

This definition meets the intuition of data redundancy we had derived from our motivating example. In particular, the occurrences of *Lava*, *Lion*, and *Gill* are α_4-, α_3- and α_1-redundant, respectively. Importantly, α_i-redundant data value occurrences can only be caused by PFDs $(X \to Y, \beta)$ that apply to the world of the occurrence, that is, where $\beta \geq \beta_{k+1-i}$. Hence, α_1-redundancy can be caused by PFDs with any certainty degree in $\beta_1, \ldots, \beta_k$, while α_k-redundancy can only be caused by PFDs with certainty degree β_1. Naturally, we have now arrived at the definition of the following semantic normal forms.

Definition 3 *A possibilistic relation schema $(R, \mathcal{S})$ is in α_i-Redundancy-Free Normal Form (α_i-RFNF) with respect to a set Σ of PFDs over $(R, \mathcal{S})$ if and only if there do not exist a possibilistic relation $(r, Poss_r)$ over $(R, \mathcal{S})$ that satisfies Σ, an attribute $A \in R$, and a tuple $t \in r_i$ such that $t(A)$ is α_i-redundant.* ∎

(MEETING, $\mathcal{S}$) is not in α_4-RFNF, α_3-RFNF, nor α_1-RFNF, but it is in α_2-RFNF with respect to Σ. The negative results follow immediately from Figure 4 above, but the satisfaction of the α_2-RFNF condition is not obvious. The next result shows that α_i-RFNF characterizes possibilistic schemata that permit exactly those possibilistic relations whose possible world r_i is guaranteed to be free from data redundancy caused by the classical FDs that apply to it.

Theorem 6 *$(R, \mathcal{S})$ is in α_i-RFNF with respect to Σ if and only if R is in RFNF with respect to Σ_{α_i}.*

Proof We show first the following: if $(R, \mathcal{S})$ is not in α_i-RFNF with respect to Σ, then R is in not in RFNF with respect to Σ_{α_i}. According to our hypothesis, there is some possibilistic relation $(r, Poss_r)$ over $(R, \mathcal{S})$ that satisfies Σ, an attribute $A \in R$, and a tuple $t \in r_i$ such that $t(A)$ is redundant with respect to Σ_{α_i}. In particular, it follows that r_i satisfies Σ_{α_i} since r satisfies Σ. Hence, R is not in RFNF with respect to Σ_{α_i}.

We now show: if R is in not in RFNF with respect to Σ_{α_i}, then $(R, \mathcal{S})$ is not in α_i-RFNF with respect to Σ. According to our hypothesis, there is some relation r_i over

R that satisfies Σ_{α_i}, and some $t \in r_i$ and $A \in R$ such that $t(A)$ is redundant with respect to Σ_{α_i}. In particular, r_i must contain some tuple $t_1 \neq t$. We now extend the relation r_i to a possibilistic relation $(r, Poss_r)$ by defining $Poss_r(t_1) = \alpha_1$ and $Poss_r(t') = \alpha_i$ for all $t' \in r_i - \{t_1\}$. We show that r satisfies Σ. If $(U \to V, \beta) \in \Sigma$ is in Σ_{α_i}, then $C_r(U \to V) = \beta_1 \geq \beta$. If $(U \to V, \beta) \notin \Sigma_{\alpha_i}$, then $\beta < \beta_{k+1-i}$. If r_i satisfies $U \to V$, then $C_r(U \to V) = \beta_1 \geq \beta$. If r_i violates $U \to V$, then $C_r(U \to V) = \beta_{k+2-i} \geq \beta$. As $t(A)$ is α_i-redundant we have shown that $(R, \mathcal{S})$ is not in α_i-RFNF with respect to Σ. ∎

7.2 Scaled BCNF

Our goal is now to characterize α-Redundancy Free normal form, which is a semantic normal form, purely syntactically. This is achieved by scaling the classical BCNF condition. Recall that a relation schema R is in Boyce-Codd normal form (BCNF) with respect to a set Σ of FDs over R if and only if for all $X \to Y \in \Sigma^+_{\mathfrak{A}}$ where $Y \not\subseteq X$, we have $X \to R \in \Sigma^+_{\mathfrak{A}}$. Here, $\Sigma^+_{\mathfrak{R}}$ denotes the syntactic closure of Σ with respect to the set $\mathfrak{R}$ of inference rules. We will also show that the definition of scaled BCNF is independent of the representation of the PFDs and can be checked efficiently. While α-RFNF is defined in terms of the semantics in the possible world with possibility degree α, scaled BCNF is defined in terms of the syntax of the given PFDs with their certainty degrees.

Definition 4 *A possibilistic relation schema $(R, \mathcal{S})$ is in* β-Boyce-Codd Normal Form *with respect to a set Σ of PFDs over $(R, \mathcal{S})$ if and only if for every PFD $(X \to Y, \beta) \in \Sigma^+_{\mathfrak{P}}$ where $Y \not\subseteq X$, we have $(X \to R, \beta) \in \Sigma^+_{\mathfrak{P}}$.* ∎

Note that the property of being in β-BCNF with respect to Σ is independent of the representation of Σ. That is, for any cover Σ' of Σ, $(R, \mathcal{S})$ is in β-BCNF with respect to Σ if and only if $(R, \mathcal{S})$ is in β-BCNF with respect to Σ'. The same way α-RFNF with respect to a PFD set Σ could be characterized by RFNF with respect to the FD set Σ_α, β-BCNF with respect to a PFD set Σ can be characterized by BCNF with respect to the FD set Σ_β.

Theorem 7 *$(R, \mathcal{S})$ is in β-BCNF with respect to a set Σ if and only if R is in BCNF with respect to Σ_β.*

Proof By definition, $(R, \mathcal{S})$ is in β-BCNF with respect to a set Σ if and only if for every PFD $(X \to Y, \beta) \in \Sigma^+_{\mathfrak{P}}$ and $Y \not\subseteq X$ we have $(X \to R, \beta) \in \Sigma^+_{\mathfrak{P}}$. Due to Theorem 2 the latter condition is equivalent to saying that for every PFD $(X \to Y, \beta)$ that is implied by Σ and $Y \not\subseteq X$, the PFD $(X \to R, \beta)$ is implied by Σ, too. According to Theorem 1, this statement is equivalent to saying that for every FD $X \to Y$ that is implied by Σ_β and $Y \not\subseteq X$, the FD $X \to R$ is implied by Σ_β, too. Due to the completeness of the Armstrong axioms for the implication of FDs, this statements is equivalent to saying that for every FD $X \to Y \in (\Sigma_\beta)^+_{\mathfrak{A}}$ where $Y \not\subseteq X$ we have $X \to R \in (\Sigma_\beta)^+_{\mathfrak{A}}$, too. The latter statement, however, is equivalent to saying that R is in BCNF with respect to Σ_β. ∎

We are now in a position to characterize the semantic α_i-RFNF by the syntactic β_{k+1-i}-BCNF.

Figure 5: Relationships between Normal Forms

$$
\begin{array}{cccc}
\Sigma_{\beta_4}\text{-}BCNF & \Sigma_{\beta_3}\text{-}BCNF & \Sigma_{\beta_2}\text{-}BCNF & \Sigma_{\beta_1}\text{-}BCNF \\
\Updownarrow & \Updownarrow & \Updownarrow & \Updownarrow \\
\beta_4\text{-}BCNF & \beta_3\text{-}BCNF & \beta_2\text{-}BCNF & \beta_1\text{-}BCNF \\
\Updownarrow & \Updownarrow & \Updownarrow & \Updownarrow \\
\alpha_1\text{-}RFNF & \alpha_2\text{-}RFNF & \alpha_3\text{-}RFNF & \alpha_4\text{-}RFNF \\
\Updownarrow & \Updownarrow & \Updownarrow & \Updownarrow \\
\Sigma_{\alpha_1}\text{-}RFNF & \Sigma_{\alpha_2}\text{-}RFNF & \Sigma_{\alpha_3}\text{-}RFNF & \Sigma_{\alpha_4}\text{-}RFNF
\end{array}
$$

Theorem 8 *For all $i = 1, \ldots, k$, $(R, \mathcal{S})$ with $|\mathcal{S}| = k + 1$ is in α_i-RFNF with respect to Σ if and only if $(R, \mathcal{S})$ is in β_{k+1-i}-BCNF with respect to Σ.*

Proof By Theorem 6, $(R, \mathcal{S})$ is in α_i-RFNF with respect to Σ if and only if R is in RFNF with respect to Σ_{α_i}. Since $\Sigma_{\alpha_i} = \Sigma_{\beta_{k+1-i}}$, and since RFNF and BCNF coincide in relational databases, the latter statement is equivalent to saying that R is in BCNF with respect to $\Sigma_{\beta_{k+1-i}}$. However, the last statement is equivalent to saying that $(R, \mathcal{S})$ is in β_{k+1-i}-BCNF with respect to Σ, by Theorem 7. ∎

Figure 5 shows the correspondences between the syntactic and semantic normal forms, and their relationships to classical normal forms.

Due to the cover-insensitivity of the β-BCNF condition, one may wonder about the efficiency of checking whether a given possibilistic relation schema $(R, \mathcal{S})$ is in β-BCNF with respect to a set Σ. Indeed, as in the classical case it suffices to check some PFDs in Σ instead of checking all PFDs in $\Sigma^+_{\mathfrak{P}}$.

Theorem 9 *A possibilistic relation schema $(R, \mathcal{S})$ is in β-BCNF with respect to a set Σ of PFDs over $(R, \mathcal{S})$ if and only if for every PFD $(X \to Y, \beta') \in \Sigma$ where $\beta' \geq \beta$ and $Y \not\subseteq X$ we have $(X \to R, \beta) \in \Sigma^+_{\mathfrak{P}}$.*

Proof By Theorem 7, $(R, \mathcal{S})$ is in β-*BCNF* with respect to Σ if and only if R is in BCNF with respect to Σ_β. However, the latter condition is well-known to be equivalent to checking for every FD $X \to Y \in \Sigma_\beta$ where $Y \not\subseteq X$ that $X \to R \in (\Sigma_\beta)^+_{\mathfrak{A}}$. The condition that $X \to Y \in \Sigma_\beta$ is equivalent to saying that $(X \to Y, \beta') \in \Sigma$ for $\beta' \geq \beta$. Lastly, the condition $X \to R \in (\Sigma_\beta)^+_{\mathfrak{A}}$ is equivalent to saying that $(X \to R, \beta') \in \Sigma^+_{\mathfrak{P}}$ for some $\beta' \geq \beta$, according to Theorem 2 and Theorem 1.

Example 7 *Let* (MEETING, $\mathcal{S}$) *and Σ be as in Example 1. Using Theorem 9 we can observe that the schema is neither in β_1-, nor β_2-, nor β_4-BCNF with respect to Σ, but it is in β_3-BCNF with respect to Σ. By Theorem 8 we conclude that the schema is neither in α_4-, nor α_3-, nor α_1-RFNF with respect to Σ, but it is in α_2-RFNF with respect to Σ. By Theorem 7 it follows that* MEETING *is neither in BCNF with respect to Σ_{β_1}, nor Σ_{β_2}, nor Σ_{β_4}, but it is in BCNF with respect to Σ_{β_3}. Finally, by Theorem 6, it follows that* MEETING *is neither in RFNF with respect to Σ_{α_4}, nor Σ_{α_3}, nor Σ_{α_1}, but it is in RFNF with respect to Σ_{α_2}.*

7.3 Scaled Third Normal Form

Similar to the β-BCNF we are now introducing the β-Third normal form (3NF). The goal of this normal form is similar to its classical counterpart: 3NF ensures that all functional dependencies can be enforced locally, without the need of joining relations to check for consistency of updates. We will return to this point in the following section.

Recall the 3NF condition from relational databases: A relation schema R is in *3NF* with respect to a given set Σ of FDs if and only if for every FD $X \to A \in \Sigma_{\mathfrak{A}}^+$ where $A \notin X$, we have $X \to R \in \Sigma_{\mathfrak{A}}^+$ or A is a prime attribute. An attribute A is *prime* if and only if it occurs in some minimal key with respect to Σ. An attribute subset X of R is a *key* of R with respect to Σ if and only if $X \to R \in \Sigma_{\mathfrak{A}}^+$. A key X of R is *minimal* with respect to Σ if and only if every proper attribute subset Y of X is not a key of R with respect to Σ.

We will now introduce analogous concepts for possibilistic databases. Given a possibilistic relation schema $(R, \mathcal{S})$, a certainty degree $\beta \in \mathcal{S}^T$, and a set Σ of PFDs over $(R, \mathcal{S})$, an attribute subset X of R is said to be a *β-key* of R with respect to Σ if and only if $(X \to R, \beta) \in \Sigma_{\mathfrak{P}}^+$. A β-key X of R with respect to Σ is a *β-minimal key* if and only if every proper attribute subset Y of X is not a β-key of R with respect to Σ. An attribute $A \in R$ is said to be *β-prime* if and only if it is contained in some β-minimal key X of R with respect to Σ.

Definition 5 *A possibilistic relation schema $(R, \mathcal{S})$ is in β-Third Normal Form (3NF) with respect to a set Σ of PFDs over $(R, \mathcal{S})$ if and only if for every PFD $(X \to A, \beta) \in \Sigma_{\mathfrak{P}}^+$ where $A \notin X$, we have $(X \to R, \beta) \in \Sigma_{\mathfrak{P}}^+$ or A is a β-prime attribute.* ∎

Theorem 7 characterized β-BCNF with respect to the PFD set Σ in terms of classical BCNF with respect to the β-cut Σ_β. An analogous result holds for 3NF.

Theorem 10 *$(R, \mathcal{S})$ is in β-3NF with respect to a set Σ if and only if R is in 3NF with respect to Σ_β.*

Proof Theorem 1 guarantees that $(X \to Y, \beta) \in \Sigma_{\mathfrak{P}}^+$ if and only if $X \to Y \in (\Sigma_\beta)_{\mathfrak{A}}^+$. In particular, X is a (minimal) β-key of R with respect to Σ if and only if X is a (minimal) key of R with respect to Σ_β. ∎

Again, due to the cover-insensitivity of the β-3NF condition, one may wonder about the "efficiency" of checking whether a given possibilistic relation schema $(R, \mathcal{S})$ is in β-3NF with respect to a set Σ. Indeed, as in the classical case it suffices to check some PFDs in Σ instead of checking all PFDs in $\Sigma_{\mathfrak{P}}^+$.

Theorem 11 *A possibilistic relation schema $(R, \mathcal{S})$ is in β-3NF with respect to a set Σ of PFDs over $(R, \mathcal{S})$ if and only if for every PFD $(X \to Y, \beta') \in \Sigma$ where $\beta' \geq \beta$ and $Y \not\subseteq X$, we have $(X \to R, \beta) \in \Sigma_{\mathfrak{P}}^+$ or every attribute $A \in Y - X$ is a β-prime attribute.*

Proof A relation schema R is in 3NF with respect to an FD set Σ_β if and only if for every FD $X \to Y \in \Sigma_\beta$ where $Y \not\subseteq X$, we have $X \to R \in (\Sigma_\beta)_{\mathfrak{A}}^+$ or every attribute $A \in Y - X$ is prime. The theorem now follows from Theorem 1. ∎

Figure 6: Applying $Manager, Time \rightarrow Room$ to decompose world r_4 of Figure 2

Project	*Time*	*Manager*
Eagle	Mon, 9am	Ann
Hippo	Mon, 1pm	Ann
Kiwi	Mon, 1pm	Pete
Kiwi	Tue, 2pm	Pete
Lion	Tue, 4pm	Gill
Lion	Wed, 9am	Gill
Lion	Wed, 11am	Bob
Lion	Wed, 11am	Jack
Lion	Wed, 11am	Pam
Tiger	Wed, 11am	Pam

Time	*Manager*	*Room*
Mon, 9am	Ann	Aqua
Mon, 1pm	Ann	Aqua
Mon, 1pm	Pete	Buff
Tue, 2pm	Pete	Buff
Tue, 4pm	Gill	Buff
Wed, 9am	Gill	Cyan
Wed, 11am	Bob	Cyan
Wed, 11am	Jack	Cyan
Wed, 11am	Pam	Lava

Example 8 *Let* $(\text{MEETING}, \mathcal{S})$ *and* Σ *be as in Example 1. Using Theorem 11 we can observe that the schema is in neither* β_1*- nor* β_2*-3NF with respect to* Σ*, but it is in* β_3*-3NF and in* β_4*-3NF with respect to* Σ*. Finally, by Theorem 10, it follows that* MEETING *is neither in 3NF with respect to* Σ_{β_1} *nor* Σ_{β_2}*, but it is in 3NF with respect to* Σ_{β_3} *and in 3NF with respect to* Σ_{β_4}*.* ∎

8 Scaled Normalization

In this section we establish algorithmic means to design relational database schemata for applications with uncertain data. For that purpose, we normalize a given possibilistic relation schemata $(R, \mathcal{S})$ with respect to the given set Σ of PFDs. Our strategy is to fix a certainty degree $\beta \in \mathcal{S}^T$ that determines which possible worlds we normalize with respect to which FDs. We pursue BCNF decompositions to obtain lossless decompositions free of any data redundancy but where some FDs may require validation by joining some relations. We also pursue 3NF-synthesis to obtain lossless decompositions on which all FDs can be validated without joining any relations (i.e. a dependency-preserving decomposition), and where the level of data redundancy is minimal with respect to any decompositions that are dependency-preserving. Applying our strategy to different certainty degrees provides organizations with a whole scale of normalized database schemata, each targeted at different degrees of data integrity, the efficiency of different updates, and the efficiency of different queries.

8.1 Scaled BCNF Decomposition

We recall basic terminology from relational databases. A *decomposition* of relation schema R is a set $\mathcal{D} = \{R_1, \ldots, R_n\}$ of relation schemata such that $R_1 \cup \cdots \cup R_n = R$. For $R_j \subseteq R$ and FD set Σ over R, $\Sigma[R_j] = \{X \to Y \mid X \to Y \in \Sigma^+_{\mathfrak{A}} \text{ and } X, Y \subseteq R_j\}$ denotes the *projection* of Σ onto R_j. A decomposition $\mathcal{D}$ of a relation schema R with FD set Σ is called *lossless* if and only if every relation r over R that satisfies Σ is the join of its projections on the elements of $\mathcal{D}$, that is, $r = \bowtie_{R_j \in \mathcal{D}} r[R_j]$. Here, $r[R_j] = \{t(R_j) \mid t \in r\}$. A *BCNF* decomposition of a relation schema R with FD set Σ is a decomposition $\mathcal{D}$ of R where every $R_j \in \mathcal{D}$ is in BCNF with respect to $\Sigma[R_j]$. Theorem 7 motivates Definition 6.

Definition 6 *An α_{k+1-i}-lossless BCNF decomposition of a possibilistic relation schema $(R, \{\alpha_1, \ldots, \alpha_{k+1}\})$ with respect to the PFD set Σ is a lossless BCNF decomposition of R with respect to Σ_{β_i}.* ∎

Instrumental to Definition 6 is the following decomposition theorem, proven in [30]. It generalizes the classical decomposition theorem for FDs [31], which is covered by Theorem 12 as the special case of having available just one possible world.

Theorem 12 *Let $(X \to Y, \beta_i)$ be a PFD that satisfies the possibilistic relation $(r, Poss_r)$ over the possibilistic relation schema $(R, \mathcal{S})$. Then*

$$r_{k+1-i} = r_{k+1-i}[XY] \bowtie r_{k+1-i}[X(R-Y)],$$

that is, the possible world r_{k+1-i} of r is the lossless join of its projection on XY and $X(R-Y)$. ∎

Therefore, an α_{k+1-i}-lossless BCNF decomposition with respect to a PFD set Σ can simply be obtained by performing a classical lossless BCNF-decomposition with respect to the β_i-cut Σ_{β_i} of Σ. This suggests a simple lossless BCNF decomposition strategy.

PROBLEM:	Scaled BCNF Decomposition
INPUT:	Possibilistic Relation Schema $(R, \mathcal{S})$ Set Σ of PFDs over $(R, \mathcal{S})$ Certainty degree $\beta_i \in \mathcal{S}^T$
OUTPUT:	α_{k+1-i}-lossless BCNF decomposition of $(R, \mathcal{S})$ with respect to Σ
METHOD:	Perform a lossless BCNF decomposition of R with respect to Σ_{β_i}

We illustrate the decomposition process on our running example.

Example 9 *Let* (MEETING, $\mathcal{S}$) *and Σ be as in Example 1. As* (MEETING, $\mathcal{S}$) *is not in β_1-BCNF with respect to Σ, we perform a α_4-lossless BCNF decomposition. The result consists of $R_1 = \{Project, Manager, Time\}$ with projected FD set*

$$\Sigma_{\beta_1}[R_1] = \emptyset,$$

Figure 7: Applying $Room, Time \rightarrow Project$ to decompose world r_3 of Figure 2

Project	*Time*	*Room*
Eagle	Mon, 9am	Aqua
Hippo	Mon, 1pm	Aqua
Kiwi	Mon, 1pm	Buff
Kiwi	Tue, 2pm	Buff
Lion	Tue, 4pm	Buff
Lion	Wed, 9am	Cyan
Lion	Wed, 11am	Cyan
Lion	Wed, 11am	Lava

Time	*Manager*	*Room*
Mon, 9am	Ann	Aqua
Mon, 1pm	Ann	Aqua
Mon, 1pm	Pete	Buff
Tue, 2pm	Pete	Buff
Tue, 4pm	Gill	Buff
Wed, 9am	Gill	Cyan
Wed, 11am	Bob	Cyan
Wed, 11am	Jack	Cyan
Wed, 11am	Pam	Lava

and $R_2 = \{Manager, Room, Time\}$ *with projected FD set*

$$\Sigma_{\beta_1}[R_2] = \{Manager, Time \rightarrow Room\}.$$

Every FD in Σ_{β_1} *is implied by* $\Sigma_{\beta_1}[R_1] \cup \Sigma_{\beta_1}[R_2]$. *Since the PFDs in* Σ *apply to worlds* r_1, r_2, r_3 *and* r_4 *from Figure 2, this decomposition may be applied to the world* r_4 *and results in the projections illustrated in Figure 6.* ∎

Example 10 *Let* $(\text{MEETING}, \mathcal{S})$ *and* Σ *be as in Example 1. As* $(\text{MEETING}, \mathcal{S})$ *is not in* β_2*-BCNF with respect to* Σ*, we perform an* α_3*-lossless BCNF decomposition. The result consists of* $R_1 = \{Project, Room, Time\}$ *with projected FD set*

$$\Sigma_{\beta_2}[R_1] = \{Room, Time \rightarrow Project\},$$

and $R_2 = \{Manager, Room, Time\}$ *with projected FD set*

$$\Sigma_{\beta_2}[R_2] = \{Manager, Time \rightarrow Room\}.$$

Note that every FD in Σ_{β_2} *is implied by* $\Sigma_{\beta_2}[R_1] \cup \Sigma_{\beta_2}[R_2]$. *Since the PFDs in* Σ *apply to worlds* r_1, r_2 *and* r_3 *from Figure 2, this decomposition may be applied to the world* r_3 *and results in the projections illustrated in Figure 7.* ∎

The situation in the last two examples is rather special, in fact, one cannot hope to preserve all FDs in the BCNF decomposition process. This is illustrated with another case on our running example.

Example 11 *Let* $(\text{MEETING}, \mathcal{S})$ *and* Σ *be as in Example 1. As* $(\text{MEETING}, \mathcal{S})$ *is not in* β_4*-BCNF with respect to* Σ*, we perform an* α_1*-lossless BCNF decomposition. The result is* $R_1 = \{Manager, Project\}$ *with projected FD set*

$$\Sigma_{\beta_4}[R_1] = \{Project \rightarrow Manager\},$$

and $R_2 = \{Project, Room, Time\}$ *with projected FD set*

$$\Sigma_{\beta_4}[R_2] = \{Room,\ Time \rightarrow Project, Project,\ Time \rightarrow Room\}.$$

Note that the FD Manager, $Time \rightarrow Room$ *is not implied by* $\Sigma_{\beta_4}[R_1] \cup \Sigma_{\beta_4}[R_2]$. ∎

A decomposition $\mathcal{D}$ of relation schema R with FD set Σ is *dependency-preserving* if and only if $\Sigma^+_{\mathfrak{A}} = (\cup_{R_j \in \mathcal{D}} \Sigma[R_j])^+_{\mathfrak{A}}$.

Definition 7 *A* β-dependency-preserving *decomposition of a possibilistic relation schema* $(R, \mathcal{S})$ *with respect to the PFD set* Σ *is a dependency-preserving decomposition of* R *with respect to* Σ_β. ∎

The α_3-lossless BCNF decomposition from Example 10 is β_2-dependency-preserving, but the α_1-lossless BCNF decomposition from Example 11 is not β_4-dependency-preserving. In fact, for (MEETING, $\mathcal{S}$) and Σ as in Example 1 there is no β_4-dependency-preserving lossless BCNF decomposition of Σ. In practice, lost dependencies can only be validated by joining relations after inserts or modification. For example, to validate the FD *Manager*, $Time \rightarrow Room$ after an update, one would have to join R_1 and R_2 from Example 11. This can be prohibitively expensive.

8.2 Scaled 3NF Synthesis

3NF synthesis guarantees dependency-preservation, but cannot guarantee the elimination of all data redundancy in terms of FDs. Recently, an information-theoretic framework was developed to provide a formal justification for Third normal form by showing that it pays the smallest possible price, in terms of data redundancy, for achieving dependency-preservation [27]. For these reasons we will now equip our database schema design framework for uncertain data with an appropriate 3NF synthesis strategy.

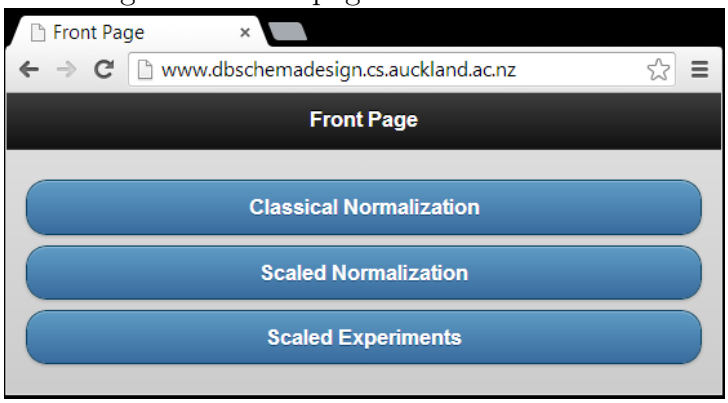
A 3NF-decomposition of a relation schema R with respect to an FD set Σ is a decomposition $\mathcal{D}$ of R where every $R_j \in \mathcal{D}$ is in 3NF with respect to $\Sigma[R_j]$. Theorem 10 motivates the following definition.

Definition 8 *A* β_i-dependency-preserving, α_{k+1-i}-lossless 3NF decomposition *of a possibilistic relation schema* $(R, \{\alpha_1, \ldots, \alpha_{k+1}\})$ *with respect to the PFD set* Σ *is a dependency-preserving, lossless 3NF decomposition of* R *with respect to* Σ_{β_i}. ∎

Due to Theorem 12 a β_i-dependency-preserving, α_{k+1-i}-lossless 3NF synthesis wrt a PFD set Σ can simply be obtained by performing a classical dependency-preserving lossless 3NF synthesis with respect to the β_i-cut Σ_{β_i} of Σ.

PROBLEM:	Scaled 3NF Synthesis
INPUT:	Possibilistic relation schema $(R, \mathcal{S})$
	Set Σ of PFDs over $(R, \mathcal{S})$
	Certainty degree $\beta_i \in \mathcal{S}^T$
OUTPUT:	β_i-dependency-preserving, α_{k+1-i}-lossless
	3NF decomposition of $(R, \mathcal{S})$ wrt Σ
METHOD:	Perform a dependency-preserving, lossless
	3NF synthesis of R with respect to Σ_{β_i}

Figure 8: Front page of Web-based GUI



We illustrate the synthesis process on our running example.

Example 12 *Let* (MEETING, $\mathcal{S}$) *and* Σ *be as in Example 1. As* (MEETING, $\mathcal{S}$) *is not in* β_2*-3NF with respect to* Σ*, we perform an* α_3*-lossless,* β_2*-dependency-preserving 3NF synthesis. The result consists of* $R_1 = \{Project, Room, Time\}$ *with projected FD set*

$$\Sigma_{\beta_2}[R_1] = \{Room, Time \rightarrow Project\},$$

and $R_2 = \{Manager, Room, Time\}$ *with projected FD set*

$$\Sigma_{\beta_2}[R_2] = \{Manager, Time \rightarrow Room\}.$$

Note that R_1 *is in BCNF with respect to* $\Sigma_{\beta_2}[R_1]$*, and* R_2 *is in BCNF with respect to* $\Sigma_{\beta_2}[R_2]$*, as we have already seen in Example 10.* ∎

The ultimate we can strive for is a dependency-preserving, lossless BCNF decomposition. There are two approaches: performing a lossless BCNF decomposition and hope that it is dependency-preserving, or performing a dependency-preserving 3NF synthesis and hope that all schemata are actually in BCNF. It is a natural and important practical question which approach has a better chance of succeeding. We will provide an answer to this question with first empirical evidence later when we present our experiments.

9 Web-based GUI

A Web-based GUI was developed to provide full access to our algorithms and experiments. The GUI is hosted at

`http://www.dbschemadesign.cs.auckland.ac.nz/`

and consists of three main parts: Classical Normalization, Scaled Normalization, and Scaled Experiments. The front-page of the GUI can be seen in Figure 8.

Under *Classical Normalization*, users can input a relation schema and a set of FDs. Several algorithms can be applied to this and additional input. Regarding BCNF decompositions, users can choose from the standard algorithm, the polynomial-time algorithm

[35], and a variation of the latter that computes the projection of original FD sets to the final output schemata only. For 3NF synthesis, users can decide whether the algorithm should be applied with respect to a canonical cover as before (Option: Unique LHS), or with respect to non-redundant, L-reduced covers in which every FD has only one attribute on the right-hand side (Option: Expanded). Users can also choose to just compute either of these covers for a given FD set, to compute the set of all minimal keys, to compute the projection of the given FD set or the attribute set closure of a further to be specified attribute subset of the given schema. Finally, users can also generate randomized input data of desired size.

Under *Scaled Normalization*, scaled versions of all of the above algorithms can be applied to a set of PFDs over a possibilistic relation schema and an optional attribute subset, as specified by a user or randomly generated by the GUI. In particular, the user can specify a scale of certainty degrees in the form of positive integers, and these can be assigned to FDs to obtain PFDs. Whenever the user specifies a certainty degree (under Scale Index), the GUI applies the scaled version of an algorithm with that certainty degree. If the user does not specify a certainty degree, the same output is returned as for the classical algorithms. In addition, users can also apply the following algorithms: possibilistic canonical covers, certainty inference, FD closure, and deciding implication. As an illustration, Figure 9 shows part of the user interface available under *Scaled Normalization*, and Figure 10 shows the input and output to the β_2-BCNF decomposition from Example 10. The latter figure also shows that the application returns several timings with the output.

Under *Scaled Experiments*, our web-based GUI provides the ability to launch experiments with our algorithms. This can be done either via the Web or in a distributed way upon the high-performance computing cluster used nationwide in New Zealand. For the first option, experiments may time out quickly and the latter option is password-protected. Parameters must be specified before experiments can be run. These include `m`, which specifies the number of runs to perform, `n`, which specifies the size of the schema to be generated and can be a range values min : max, `l`, which specifies the cardinality of the scale to be used and can be a range min : max of values, and `FD set accelerator`, a number a that determines the maximum size $a \cdot$ `n` of an FD set to be generated. Available experiments include: decompositions, FD set closures, deciding implication, inferences, and covers. Several results of already run experiments are available to view by selecting a data set whose name has the format `m_n_l_filename_time`. The generated graphs are interactive and available for download in different formats.

The GUI is also targeted to support the learning of undergraduate and postgraduate students of database courses.

10 Experiments

In this section we analyze the results of our experiments regarding the scaled versions of BCNF decompositions and 3NF syntheses, the implication and inference problems of PFDs, as well as the computation of covers. The analysis includes new insights into the traditional problems, too.

Figure 9: User Interface - Scaled Normalization

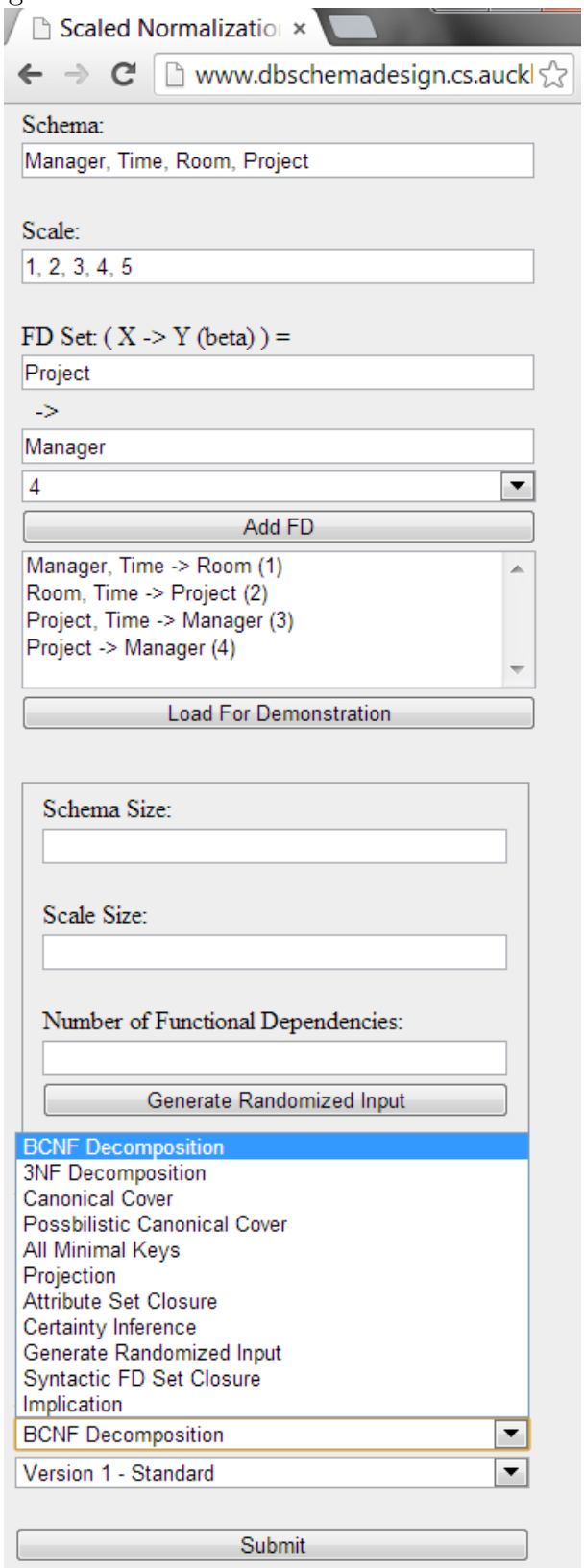


Figure 10: BCNF Example in GUI

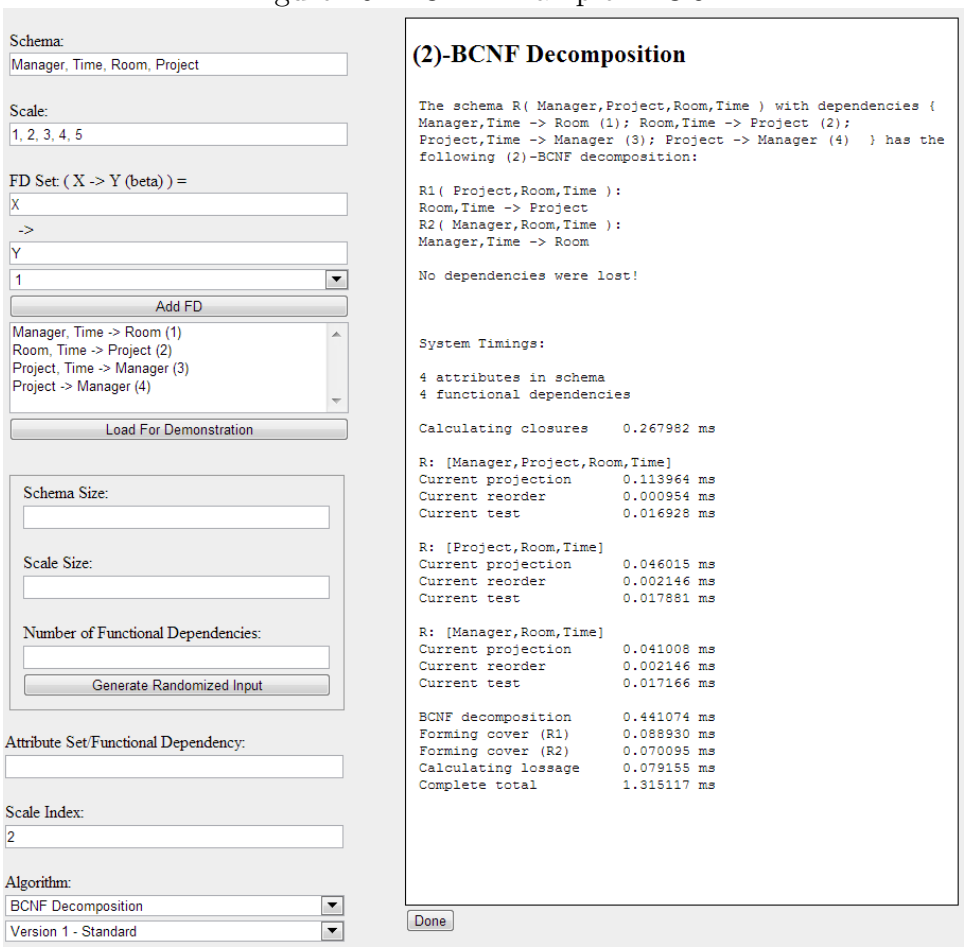
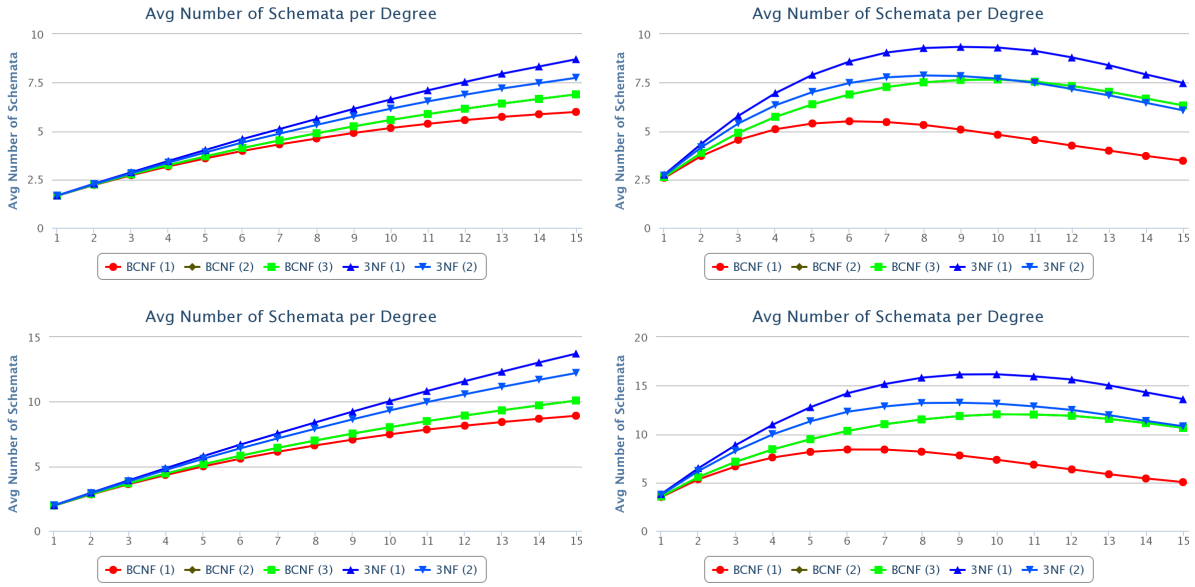


Figure 11: Average Size of Decompositions, taken over 5,000 runs each, where the Input Relation Schemata have 10 Attributes in the Top Row and 15 Attributes in the Bottom Row, and the Maximum PFD Set Size is n in the Left Column and $5n$ in the Right Column



10.1 Normalization

Our core analysis concerns the trade-offs between the elimination of data redundancy achieved by BCNF decompositions and the preservation of FDs achieved by 3NF syntheses, as well as between query and update efficiency. The simplest general measure for the query and update efficiency of a decomposition we could think of is the *number of its schemata* - to which we refer as the size of the decomposition. Intuitively, the more schemata a decomposition contains the more updates and the less queries it can support efficiently. In particular, less schemata require less joins, the main source of complexity in query evaluation. Our framework allows us to analyze the results from different angles: The normalization effort can be measured either classically in the size of the input FD set or from a scaled point of view in the given certainty degree. Our analysis should be read with both points of views in mind.

For our analysis we compare five normalization algorithms: BCNF(1) is the classical BCNF decomposition algorithm that takes exponential time and space, BCNF(2) is Tsou and Fischer's polynomial-time algorithm, BCNF(3) is the variant of BCNF(2) where the projection of original FD sets is only determined for the output schemata, 3NF(1) is the 3NF synthesis algorithm that first computes a non-redundant, L-reduced cover in which all FDs have a singleton attribute on the right-hand side, and 3NF(2) is the 3NF synthesis algorithm that first computes a canonical cover as defined previously.

Figure 11 shows the graphs of four experiments. For each of 15 certainty degrees $\beta_1, \ldots, \beta_{15}$, 5,000 sets of at most $a \cdot n$ PFDs over a relation schema with n attributes were created, and β_i-BCNF decompositions and β_i-3NF syntheses created, using our 5 algorithms. The top row shows the graphs where $n = 10$ with $a = 1$ on the left and $a = 5$

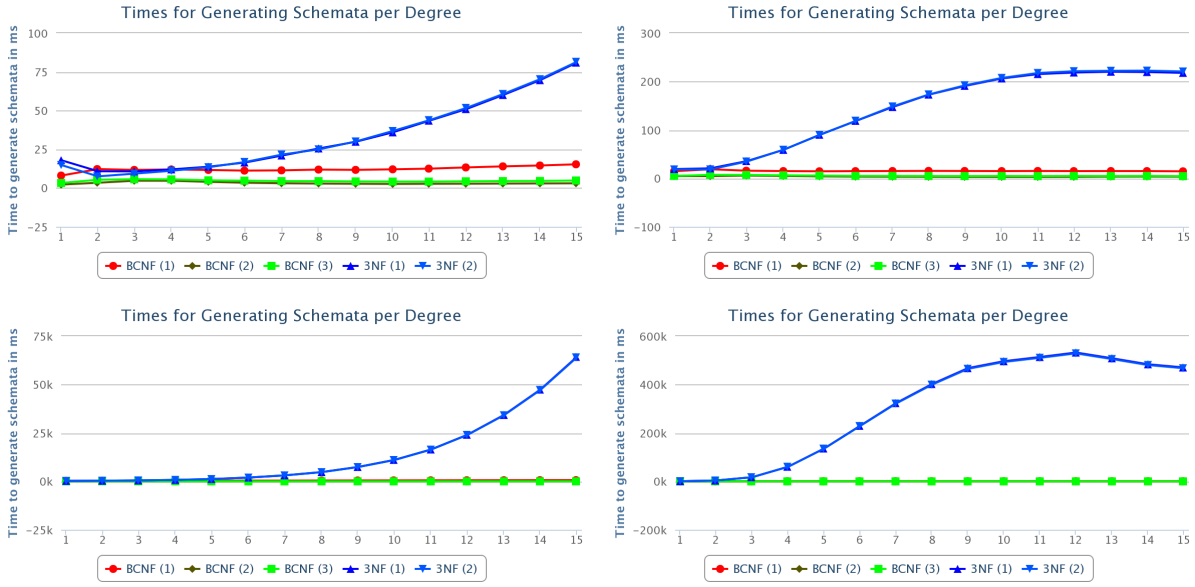
on the right. The bottom row shows the graphs where $n = 15$ with $a = 1$ on the left and $a = 5$ on the right.

The four different experiments show nearly uniform results: The lowest-sized decomposition is BCNF(1), followed by BCNF(2,3) which agree, followed by 3NF(2), and then 3NF(1). The results are rather intuitive as 3NF produces more schemata to accommodate the preservation of all functional dependencies and thereby sacrificing the elimination of some data redundancy. The graphs show the price for producing a BCNF decomposition in polynomial time, which results from not having to check the BCNF condition of intermediate results and thereby producing superfluous schemata not required by BCNF(1). It is interesting that this price is still mostly smaller than that to accommodate dependency-preservation. However, we will see later that the superfluous schemata in BCNF(2,3) are unlikely to help with dependency-preservation. Finally, 3NF(2) generates less schemata than 3NF(1) as a canonical cover contains less FDs than the other cover considered. The difference between BCNF(1) and 3NF(2) tells us how many more schemata are required to guarantee dependency-preservation. The results suggest to prefer BCNF decomposition in terms of query efficiency and, obviously, in terms of elimination of data redundancy. They do not mean by any means, to prefer BCNF decomposition in terms of update efficiency. Indeed, BCNF decompositions are only more efficient for updates of redundant data value occurrences caused by FDs preserved during the decomposition, while update inefficiencies result from having to join schemata to validate unpreserved FDs.

The four graphs further show that the bigger the size of the input relation schema, the bigger the size of the decomposition - observing the column of graphs on the left and that on the right. However, the story is different in terms of growing FD size - observing each row from left to right. When there are about as many given FDs as underlying attributes, then the average size of decompositions grows linearly. When there are significantly more given FDs than underlying attributes, then a global maximum is observable in the average size of decompositions. This is explained by the fact that there will be a point when there are sufficiently many FDs that turn attribute sets into keys, in which case further decompositions become unnecessary. Not surprisingly, this saturation point comes earlier for BCNF(1) than for any of the other algorithms considered. For larger FD sets, higher levels of data integrity achieve the same levels of query efficiency as significantly lower levels of data integrity.

Figure 12 shows the average times to generate the decompositions for our four experiments. As observed with respect to the average size of decompositions, the four different experiments show nearly uniform results with respect to the average times: BCNF(2) takes the lowest time, followed by BCNF(3), followed by BCNF(1), and followed by 3NF(1,2). These results are intuitive, too. BCNF(2) demonstrates its major strength in comparison to the other algorithms, with a maximum average time of 8ms for $a = 1$ and 13ms for $a = 5$. BCNF(3) requires slightly more time with a maximum average time of 12ms for $a = 1$ and 35ms for $a = 5$, resulting from the projections of the original FD sets to the final output schemata at the end. BCNF(1) requires more time with a maximum average time of 630ms for $a = 1$ and 627ms for $a = 5$, resulting from the projections of the original FD sets to all intermediate and final schemata. Finally, 3NF(1,2) require significantly more time with a maximum average time of 64s for $a = 1$ and $530s$ for $a = 5$.

Figure 12: Average Times to Generate Decompositions, taken over 5,000 runs each, where the Input Relation Schemata have 10 Attributes in the Top Row and 15 Attributes in the Bottom Row, and the Maximum PFD Set Size is n in the Left Column and $5n$ in the Right Column



Two main observations illustrate inherent trade-offs. For BCNF decompositions firstly, smaller-sized decompositions come at the price of requiring more time to be generated. In other words, polynomial-time BCNF decompositions result from not having to spend time on validating the BCNF condition, thereby generating additional schemata that are superfluous in terms of update efficiency but may significantly add to the inefficiency of query evaluation. For 3NF decompositions secondly, the price to pay for dependency-preservation is a higher complexity in terms of the size of the decompositions and the time taken to generate them.

The ultimate decision whether to use a 3NF decomposition that is not a BCNF decomposition or a BCNF decomposition that is not dependency-preserving is likely to require a detailed analysis which differs from case to case. Considering update efficiency one must ask whether frequent updates are slowed down more by data redundancies inherent in the given 3NF decomposition that is not a BCNF decomposition, or by the validation of FDs that are not preserved in the given BCNF decomposition? This should be balanced with an analysis which of the schemata provides a better efficiency in terms of evaluating frequent queries.

An important practical question is which of the two normalization strategies is more likely to result in a best decomposition, defined as a lossless, dependency-preserving BCNF decomposition. Therefore, we have randomly created the percentages of those lossless BCNF decompositions that are dependency-preserving and of those lossless 3NF decompositions that are in BCNF. The results are shown in Figure 13. It clearly shows that the 3NF strategy is much more likely to result in a best decomposition than the BCNF strategy. This is not too surprising as it seems intuitive that the probability

Figure 13: Percentages of 3NF Outputs in BCNF versus BCNF Outputs that are Dependency-preserving, taken over 5,000 runs each, where the Input Relation Schemata have 10 Attributes in the Top Row and 15 Attributes in the Bottom Row, and the Maximum PFD Set Size is n in the Left Column and $5n$ in the Right Column

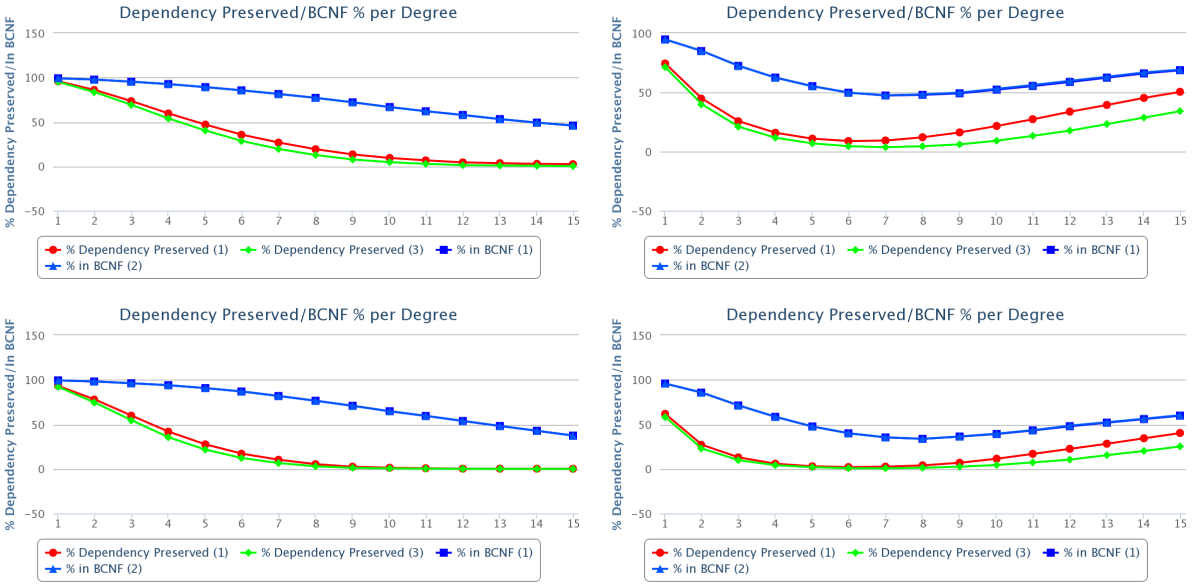
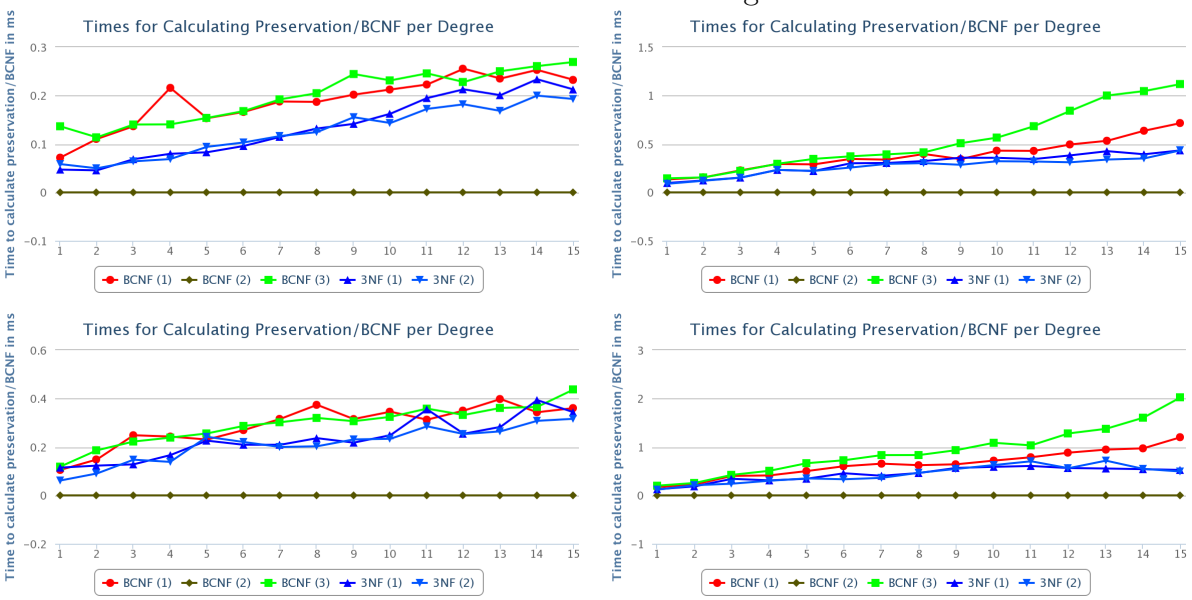


Figure 14: Times to Check for BCNF in 3NF Outputs versus Dependency-preservation in BCNF Outputs, taken over 5,000 runs each, where the Input Relation Schemata have 10 Attributes in the Top Row and 15 Attributes in the Bottom Row, and the Maximum PFD Set Size is n in the Left Column and $5n$ in the Right Column



of generating some 3NF pattern that violates the BCNF condition is lower than the probability of not preserving some FD. Nevertheless, the difference is rather substantial. Indeed, for the case where $a = 1$, the average percentage for 3NF(1,2) is 73, for BCNF(1) it is 22.5 and for BCNF(3) it is 20. For the case where $a = 5$, the average percentage for 3NF(1,2) is 53, for BCNF(1) it is 18.5 and for BCNF(3) it is 12. These observations provide first empirical explanations why 3NF is the preferred normalization strategy in practice.

Finally, Figure 14 shows that the average time to validate the BCNF condition for a given 3NF decomposition is less than the average time to validate dependency-preservation for a given BCNF decomposition. However, the most time-consuming case requires less than 2ms.

Summary. The segmentation of an FD set into PFDs using certainty degrees allows standard normalization approaches to derive a variety of relational database schemata whose final selection can be based on the targeted level of data integrity or the expected workload of queries and updates. The more certainty degrees that can be modeled effectively, the more control is gained over classical trade-offs between update and query efficiencies, as better matches to the expected workload can be found and unwanted integrity enforcement can be avoided. Indeed, excluding FDs whose certainty degree is lower than that targeted will result in database schemata that are normalized with respect to the target degree, but de-normalized in the classical sense. This explains how our framework can be viewed as unifying normalization and de-normalization. Thereby, classically de-normalized schemata can be justified in terms of the level of data integrity that is targeted.

10.2 Implication Problem

We now investigate experimentally the efficiency of deciding the implication problem for possibilistic FDs, providing confirmation of the theoretical bound and insight into the actual time it takes to arrive at the decisions.

For the experiments, we have randomly created 1,000 different instances of the implication problem for PFDs for each of the underlying relation schemata with $n = 3, \ldots, 20$ attributes. For each of these instances we have then simply removed the associated certainty degrees and decided the corresponding instances of the implication problem for FDs. For each n, we have then calculated the percentage of positive outcomes and taken the average time of deciding each of the 1,000 instances over n.

Figure 15 shows the results for PFD sets as the black line, and the results for the corresponding FD sets as the blue line. On average, the success rate of deciding implication for the corresponding FD sets doubles in comparison to the PFD sets they were derived from. In actual numbers it is an increase by 30% from an average 29% for PFD sets to an average 59.5% for the FD sets. The doubling of positive outcomes is indeed intuitive: On average, we are deciding whether the same FD is implied by twice as many given FDs. The same intuition explains the difference in times taken to decide the various instances. On average, the time for deciding implication for the corresponding FD sets doubles in comparison to the PFD sets they were derived from. In actual numbers it is an increase from an average 0.08ms for PFD sets to an average 0.17ms for the FD sets.

Figure 15: Average Outcomes and Times to Decide Implication Problems for Sets of PFDs and the Sets of FDs after Removing the Certainty Degrees, taken over 1,000 runs each, where the Input Relation Schemata have between 3 and 20 Attributes and there are 15 Certainty Degrees

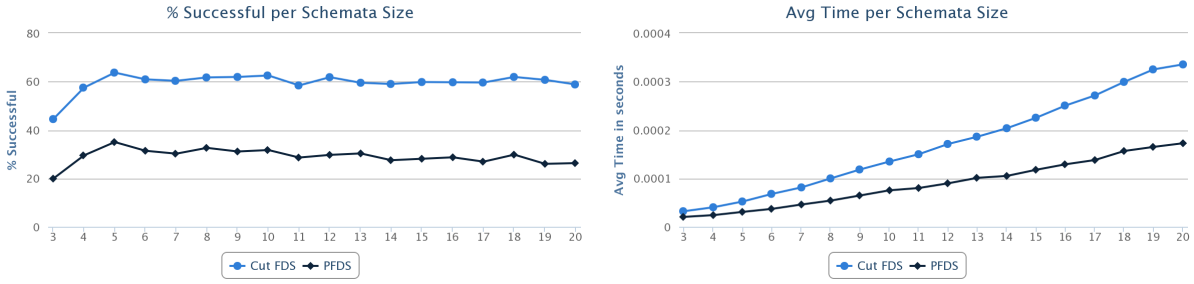


Figure 16: Average Certainty Degrees and Times to Compute Certainty Degree for Sets of PFDs, taken over 1,000 runs each, where the Input Relation Schemata have between 3 and 20 Attributes and there are 15 Certainty Degrees

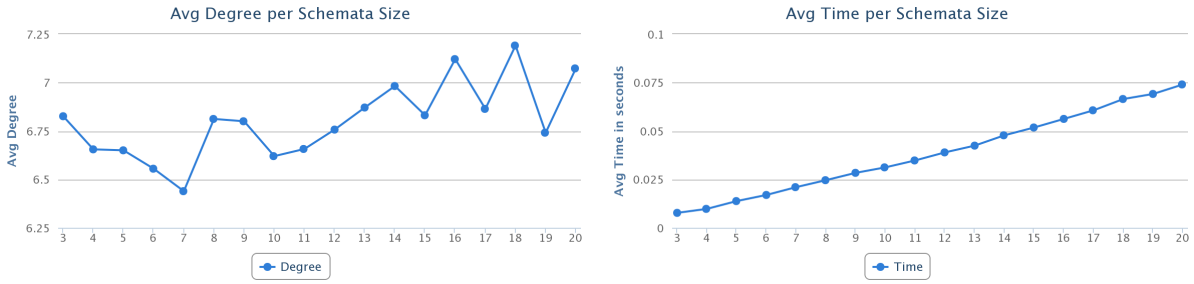
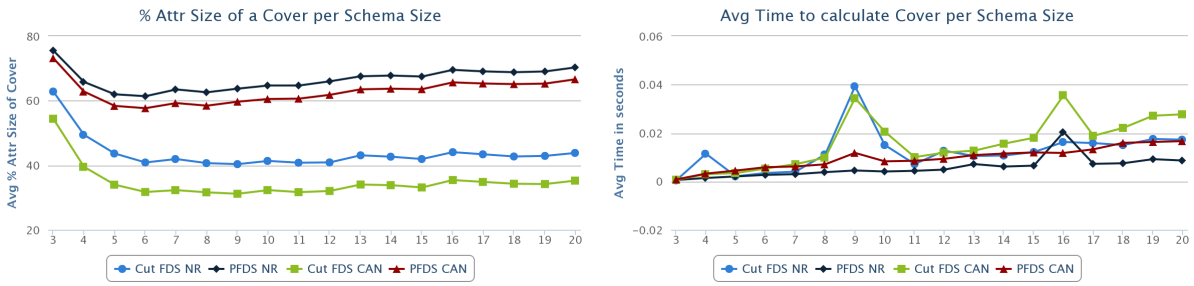


Figure 17: Average Sizes of Covers and Times for their Computation, taken over 1,000 runs each, where the Input Relation Schemata have between 3 and 20 Attributes and there are 15 Certainty Degrees



In both cases the graphs confirm the linear time complexity. In practice, the problems can both be decided very efficiently.

10.3 Inference Problem

We now investigate experimentally the efficiency of inferring the maximum certainty degree for a given FD such that it is implied by a given set of PFDs, providing confirmation of the theoretical bound and insight into the actual time it takes to arrive at the certainty degree.

For the experiments, we have randomly created 1,000 different instances of the inference problem for PFDs for each of the underlying relation schemata with $n = 3, \ldots, 20$ attributes. For each n, we have then calculated the average certainty degree and taken the average time of computing the degree for each of the 1,000 instances over n.

Figure 16 shows the results of our experiments for the inference problem. The average maximum certainty degree by which a given FD is implied by a given set of PFDs is 6.8025, taken over the averages of all $n = 3, \ldots, 20$. The average time taken to compute these degrees is 38.64ms, taken over the average times of all $n = 3, \ldots, 20$. The graph regarding the average times confirms the linear time complexity of the inference problem. In practice, the problem can be solved very efficiently.

10.4 Computing Covers

We now investigate experimentally the efficiency of computing non-redundant and canonical covers for sets of PFDs. We also compare the results to non-redundant and canonical covers for corresponding sets of FDs, obtained from the sets of PFDs by removing their certainty degrees.

For the experiments, we have randomly created 1,000 different instances of sets of up to $5n$ PFDs for each of the underlying relation schemata with $n = 3, \ldots, 20$ attributes and 15 different certainty degrees. For each n, we have then calculated the average percentage of the size of the non-redundant and canonical covers compared to their input PFD set. The size of an FD set or PFD set is the total number of attributes in it. The average time taken to compute the covers is also calculated for each n.

Figure 17 shows the results of our experiments for the computation of covers. In consistency with our results for the implication problem, covers for PFD sets are of larger size than covers for the corresponding FD sets. As before, the simple reason is that we indirectly increase the input size by removing certainty degrees, which increases the redundancy among the given FDs. In both the PFD and FD case, canonical covers result in significant additional savings in terms of the size. The additional savings in size require additional time to be computed. On average, non-redundant covers of the PFD sets result in 66.64% of the original size, computed in an average time of 5.77ms; and canonical covers of the PFD sets result in 62.85% of the original size, computed in an average time of 9.63ms. Similarly, non-redundant covers of the corresponding FD sets result in 43.73% of the original size, computed in an average time of 12.31ms; and canonical covers of the corresponding FD sets result in 34.74% of the original size, computed in an average time of 15.76ms. The results confirm our intuition about the

Table 4: Achievements of β-BCNF and β-3NF

	α_{k+1-i}-lossless	Free from α_{k+1-i} data redundancy	β_i-dependency-preserving
β_i-BCNF	✔	✔	
β_i-3NF	✔		✔

effectiveness of the different covers and the required time to compute them. In practice, the computation of non-redundant and canonical covers is very efficient, resulting in considerably savings in size. Therefore, validating covers of (P)FD sets can result in enormous time savings whenever updates are processed.

11 Conclusion and Future Work

We have established a framework to design relational database schemata for applications that require uncertain data. Each tuple is assigned some degree of possibility, and each functional dependency is assigned some degree of certainty that models to which tuples the functional dependency applies. The framework results in a customized segmentation of the well-known Boyce-Codd and Third normal forms, extending all of their properties to each segment, as shown in Table 4. Organizations can select the design that best fits their expected workload and targeted level of data integrity. The possibility and certainty degrees empower designers to focus the normalization effort on boosting the efficient processing of most frequent updates, thereby also enabling the efficient processing of more queries. Therefore, certainty degrees can be seen as a mechanism to control the levels of data integrity, query efficiency, and update efficiency, see Figure 1. The framework further unifies traditional relational normalization and de-normalization, as classically de-normalized schemata are fully normalized with respect to the functional dependencies that meet the targeted degree of certainty. Our algorithms can be publicly accessed on a web-based GUI, which also enables users to distribute experiments on high-performance computing facilities. The experiments confirm the effectiveness and efficiency of our framework. Furthermore, they provide original insight into traditional database schema design. For instance, we establish a strong empirical justification for the use of 3NF synthesis in practice - showing that it offers a much better chance of obtaining an optimal database schema than BCNF decompositions offer.

There are several problems that warrant future research. The same way relational schema designs for certain data depend on the identification of functional dependencies that are semantically meaningful for the given application, the schema designs for uncertain data depend on the identification of semantically meaningful possibilistic functional dependencies. This strongly motivates the extension of research about Armstrong relations [22, 23] to the possibilistic setting. Further normal forms should be customized to their use for uncertain data, including 4NF that is based on multivalued dependencies [21, 36] as well as Inclusion Dependency normal form [28]. For tuple-generating depen-

dencies, such as multivalued and inclusion dependencies, it is challenging to define a suitable possibilistic semantics as the removal of tuples from the database can lead to violations of some dependencies. Dedicated data management tools should be established that incorporate possibility and certainty degrees when they manage uncertain data.

Acknowledgement

This research is supported by the Marsden fund council from Government funding, administered by the Royal Society of New Zealand.

The authors acknowledge the contribution of NeSI high-performance computing facilities to the results of this research. NZ's national facilities are provided by the NZ eScience Infrastructure and funded jointly by NeSI's collaborator institutions and through the Ministry of Business, Innovation & Employment's Research Infrastructure programme (`https://www.nesi.org.nz`).

We thank Andrew Probert who implemented our algorithms, designed the GUI and executed the experiments.

References

[1] L. Antova, C. Koch, and D. Olteanu. $10^{(10^6)}$ worlds and beyond: efficient representation and processing of incomplete information. *VLDB J.*, 18(5):1021–1040, 2009.

[2] M. Arenas and L. Libkin. An information-theoretic approach to normal forms for relational and XML data. *J. ACM*, 52(2):246–283, 2005.

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

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

[5] C. Beeri, R. Fagin, and J. H. Howard. A complete axiomatization for functional and multivalued dependencies in database relations. In *SIGMOD*, pages 47–61. ACM, 1977.

[6] O. Benjelloun, A. D. Sarma, A. Y. Halevy, M. Theobald, and J. Widom. Databases with uncertainty and lineage. *VLDB J.*, 17(2):243–264, 2008.

[7] O. Benjelloun, A. D. Sarma, C. Hayworth, and J. Widom. An introduction to ULDBs and the Trio system. *IEEE Data Eng. Bull.*, 29(1):5–16, 2006.

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

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

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

[11] J. Boulos, N. N. Dalvi, B. Mandhani, S. Mathur, C. Ré, and D. Suciu. MYSTIQ: a system for finding more answers by using probabilities. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 891–893, 2005.

[12] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

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

[14] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007.

[15] N. N. Dalvi and D. Suciu. Management of probabilistic data: foundations and challenges. In *PODS*, pages 1–12, 2007.

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

[17] D. Dubois, J. Lang, and H. Prade. Automated reasoning using possibilistic logic: semantics, belief revision, and variable certainty weights. *IEEE Trans. Knowl. Data Eng.*, 6(1):64–71, 1994.

[18] D. Dubois and H. Prade. Possibility theory. In R. A. Meyers, editor, *Computational Complexity*, pages 2240–2252. Springer New York, 2012.

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

[20] R. Fagin. Functional dependencies in a relational data base and propositional logic. *IBM Journal of Research and Development*, 21(6):543–544, 1977.

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

[22] R. Fagin. Horn clauses and database dependencies. *J. ACM*, 29(4):952–985, 1982.

[23] S. Hartmann, M. Kirchberg, and S. Link. Design by example for SQL table definitions with functional dependencies. *VLDB J.*, 21(1):121–144, 2012.

[24] J. Huang, L. Antova, C. Koch, and D. Olteanu. MayBMS: a probabilistic database management system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1071–1074, 2009.

[25] A. K. Jha and D. Suciu. Probabilistic databases with markoviews. *PVLDB*, 5(11):1160–1171, 2012.

[26] C. Koch and D. Olteanu. Conditioning probabilistic databases. *PVLDB*, 1(1):313–325, 2008.

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

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

[29] S. Link and H. Prade. Database schema design generation system and method. Patent application, Number: 14 57862, France, Filed on 18 August 2014.

[30] S. Link and H. Prade. Possibilistic functional dependencies and their relationship to possibility theory. Technical Report CDMTCS-468, The University of Auckland, 2014.

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

[32] A. D. Sarma, O. Benjelloun, A. Y. Halevy, S. U. Nabar, and J. Widom. Representing uncertain data: models, properties, and algorithms. *VLDB J.*, 18(5):989–1019, 2009.

[33] A. D. Sarma, J. D. Ullman, and J. Widom. Schema design for uncertain databases. In *Proceedings of the 3rd Alberto Mendelzon International Workshop on Foundations of Data Management*, volume 450 of *CEUR Workshop Proceedings*, 2009.

[34] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

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

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